

RIVET

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

July 13, 2016, CERN



CONTENTS

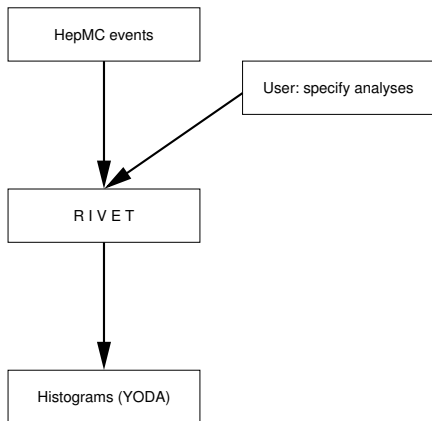
WHAT IS RIVET?

- 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)
- → allows for fair comparison with (unfolded) collider data
- Core written in robust C++, elegant and flexible Python scripts for the rest

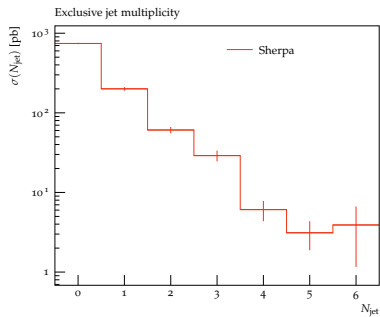
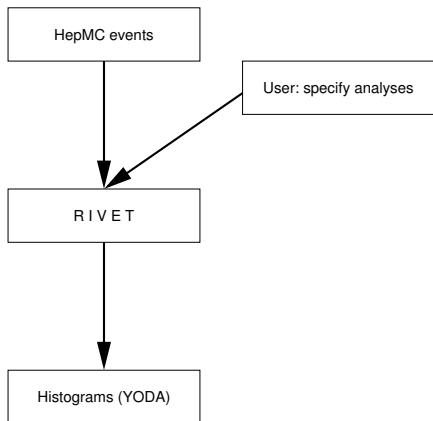
WHY USE RIVET?

- Generator independent code due to “industry standard” for simulated events (HepMC)
- Version 2.5 contains ~ 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)

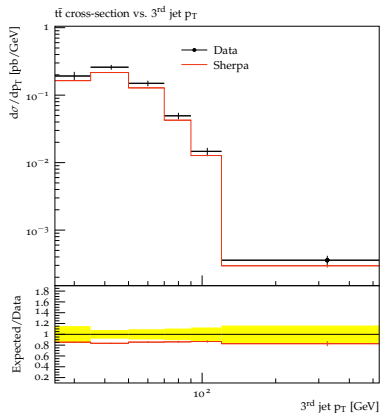
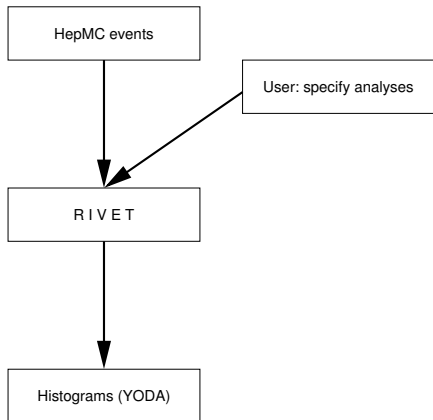
BASIC PRINCIPLE



BASIC PRINCIPLE



BASIC PRINCIPLE



EVENTS

- Several event files are available on lxplus
- 13 TeV Z+jets (muon) inclusive
- 13 TeV Z+jets (muon) with $Z_{pt} > 200$ 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](https://afs.cern.ch/work/h/hschulz/public/Events)

HANDS ON 1 — ON LXPLUS

```
1 source /afs/cern.ch/work/h/hschulz/public/setupRivet2.5.0.sh // Set up Rivet 2.5
   with Python 2.7 from LCG
3 rivet -h
5 rivet --list-analyses // List precompiled analyses shipped with Rivet
7 rivet --show-analysis CMS_2015_I1310737 // Detailed info for particular analysis
9 rivet --show-analysis MC_ZJETS_MU
  rivet --show-analysis ATLAS_2015_CONF_2015_041
11 rivet -a ATLAS_2015_CONF_2015_041 -a MC_ZJETS_MU -H Output.yoda
   INPUTFILE
13 rivet -a CMS_2015_I1310737 -H Output2.yoda --ignore-beams INPUTFILE
15 rivet-mkhtml -h
17 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  
4 }  
5  
6 void ANANAME::analyze(const Event & e) {  
7     // Executed once per event, calc observables, fill histograms here  
8 }  
9  
10 void ANANAME::finalize() {  
11     // Normalise, divide histograms here  
12 }  
13  
14 private:  
15     // Histogram pointers go here
```

HANDS ON 1 — ON LXPLUS

```
1 source /afs/cern.ch/work/h/hschulz/public/setupRivet2.5.0.sh // Set up Rivet 2.5
   with Python 2.7 from LCG
3 rivet -h
5 rivet --list-analyses // List precompiled analyses shipped with Rivet
7 rivet --show-analysis CMS_2015_I1310737 // Detailed info for particular analysis
9 rivet --show-analysis MC_ZJETS_MU
  rivet --show-analysis ATLAS_2015_CONF_2015_041
11 rivet -a ATLAS_2015_CONF_2015_041 -a MC_ZJETS_MU -H Output.yoda
   INPUTFILE
13 rivet -a CMS_2015_I1310737 -H Output2.yoda --ignore-beams INPUTFILE
15 rivet-mkhtml -h
17 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  
4 }  
5  
6 void ANANAME::analyze(const Event & e) {  
7     // Executed once per event, calc observables, fill histograms here  
8 }  
9  
10 void ANANAME::finalize() {  
11     // Normalise, divide histograms here  
12 }  
13  
14 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    _hp1d=bookProfile1D("JetPt_vs_JetEta", 50, 10, 120);  
    _c=bookCounter('passedCutXY');  
5 }  
  
7 void ANANAME::analyze(const Event & e) {  
    double weight = e.weight()  
9    _c->fill(weight);  
    _hist->fill(MET, weight);  
11    _p1d->fill(jet.pt(), jet.eta(), weight);  
    }  
13 void ANANAME::finalize() {  
    _h->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`

```
1 #include "Rivet/Projections/FastJets.hh"
3 void init() {
4     FinalState fs;
5     FastJets jetpro(fs, FastJets::ANTIKT, 0.4);
6     declare(jetpro, "Jets");
7 }
9 void ANANAME::analyze(const Event & e) {
10     const Jets& jets = apply<FastJets>(e, "Jets").jetsByPt(Cuts::pT>30*GeV);
11     foreach(const Jet& j, jets) {
12         if (j.bTagged()) { // Is there a ghost associated B-hadron
13             //
14         }
15     }
16 }
```

FINALSTATES

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

```
1 #include "Rivet/Projections/FastJets.hh"
2 #include "Rivet/Projections/IdentifiedFinalState.hh"
3 #include "Rivet/Projections/VetoedFinalState.hh"
4
5 void init() {
6     IdentifiedFinalState neutrinoFS(fs);
7     neutrinoFS.acceptNeutrinos();
8     declare(neutrinoFS, "Neutrinos");
9
10    VetoedFinalState jetinput;
11    jetinput.addVetoOnThisFinalState(neutrinoFS);
12
13    FastJets jetpro(jetinput, FastJets::ANTIKT, 0.4);
14    declare(jetpro, "Jets");
15 }
```

PLOTTING

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

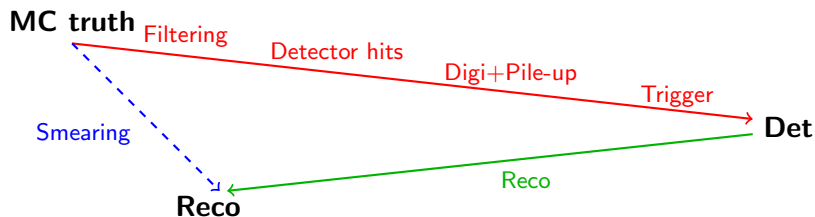
```
1 BEGIN PLOT /ATLAS_2015_I1343107/d18-x01-y01  
XLabel= $E^{\{\rm miss\}}-T$  [GeV]  
3 YLabel=Events  
XMin=150  
5 END PLOT
```


NEW IN RIVET 2.5

- Even simpler projections syntax
- Basic modelling of detector responses
- 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

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

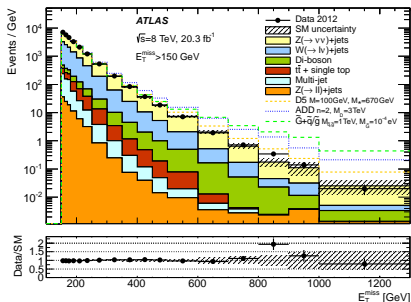
   SmearredJets sj(fj, MY_JET_SMEAR_ATLAS_RUN1,
5     [](const Jet& j){ return j.bTagged() ? 0.7*(1 - exp(-j.pT()/(10*GeV))) :
     0.01; } );
7   declare(sj, "SmearredJets");

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

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

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

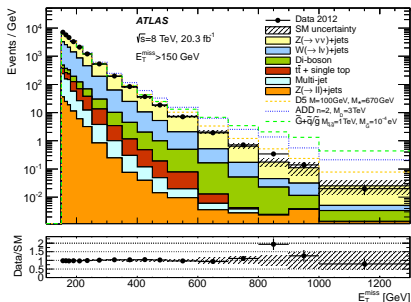
HANDS ON 2



- 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

$E_T^{\text{miss}} > 150 \text{ GeV}$

Jet quality requirements

At least one jet with $p_T > 30 \text{ GeV}$ and $|\eta| < 4.5$

Lepton and isolated track vetoes

Monojet-like selection

The leading jet with $p_T > 120 \text{ GeV}$ and $|\eta| < 2.0$

Leading jet $p_T/E_T^{\text{miss}} > 0.5$

$\Delta\phi(\text{jet}, \mathbf{p}_T^{\text{miss}}) > 1.0$

FINAL REMARKS

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

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