Rivet tutorial

ATLAS week, 16 October 2024

Martin Habedank (Glasgow) and Peng Wang (UCL)

Outline

1. **Rivet: Introduction and motivation**
2. **Rivet: Hand-on tutorial**
   a. Event generation & Rivet in a **Docker container**
   b. Event generation & Rivet in **Athena**

Loading the Docker image

For the later tutorial please load the correct Docker image (if not done already):

```
docker pull hepstore/rivet-tutorial:4.0.1
```

depending on your setup, you might have to call always docker with superuser rights:

```
sudo docker <your commands>
```

(that's bad, we know :( )

Rivet: Introduction and motivation

heavily borrowed from Andy Buckley and Chris Gütschow

Rivet origins

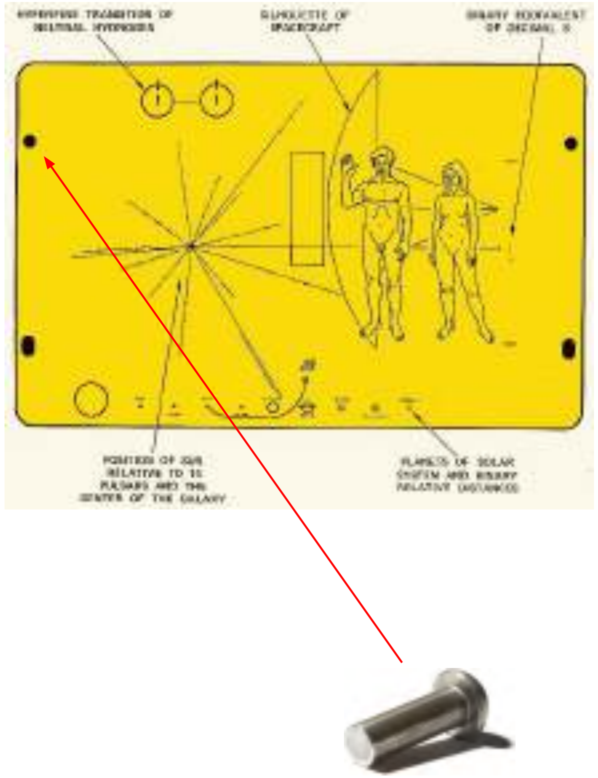
- Idea of preserving experimental analyses for MC validation: **HZTOOL** (HERA)

- Need a common language for phenomenology and experiment

➔ Tune and validate new MC event generators

- Simple/obvious idea, with surprising impact:
  - **Reproducing** key plots (or *not*) is **powerful**

  ➔ Understand physics, communicate issues, improve MCs


HYPERFINE TRANSITION OF NEUTRAL HYDROGEN
SILHOUETTE OF SPACECRAFT
BINARY EQUIVALENT OF DECIMAL 8
POSITION OF SUN RELATIVE TO 14 PULSARS AND THE CENTER OF THE GALAXY
PLANETS OF SOLAR SYSTEM AND BINARY RELATIVE DISTANCES

What is Rivet?

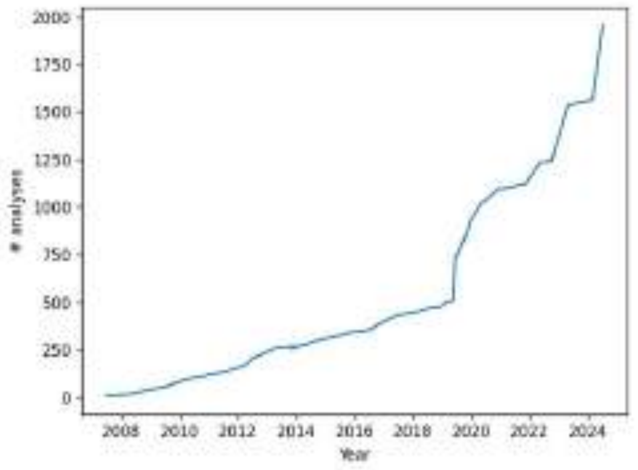
→Paper, →Webpage

- Project to **preserve analysis logic** and **foster expt-pheno collaboration**
  - The **“LHC standard” MC analysis toolkit**
  - Central to ecosystem of analysis reinterpretation tools, linking experiment to theory

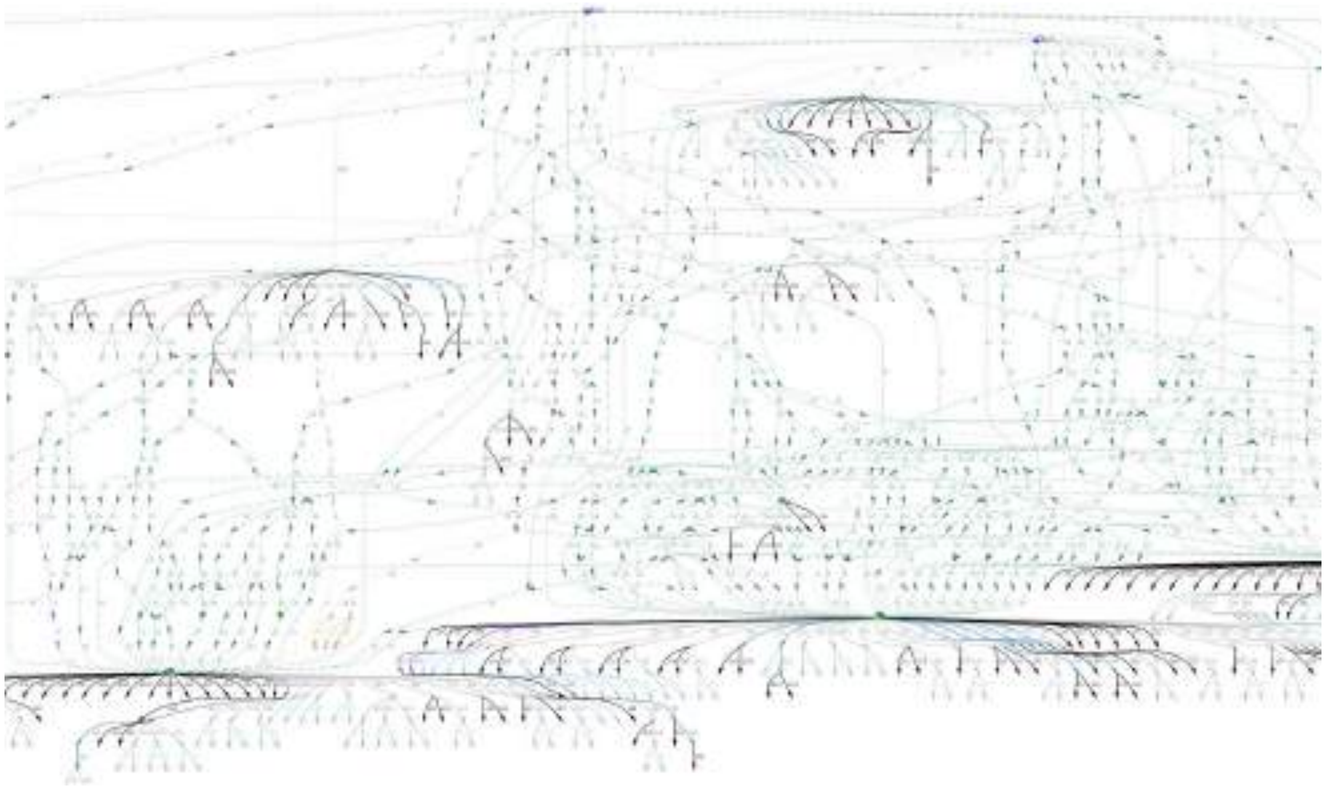
- **Vast library of measurements** of final state particles, and derived variables
  - Fiducial / generator-independence emphasis
  - Mostly collider physics, some cosmic-ray
  - Almost 2000 analyses!

- Integration with HEPData
- Transparent HepMC weight-stream handling

- But why? Event loops are trivial…

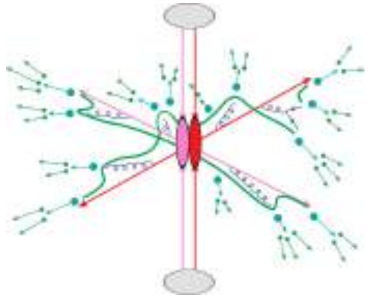


Why Rivet?



Want to avoid all physicists needing to rediscover
graph algorithms, conventions, pitfalls, physical/debug distinctions, ...

Discovered pitfalls



- **Usage of partons, bosons**, etc. directly from the event graph: **Unphysical**, depend on approximations, may not even exist!

➔ **Final state objects**

- Many analyses → lots of expensive operations are repeated

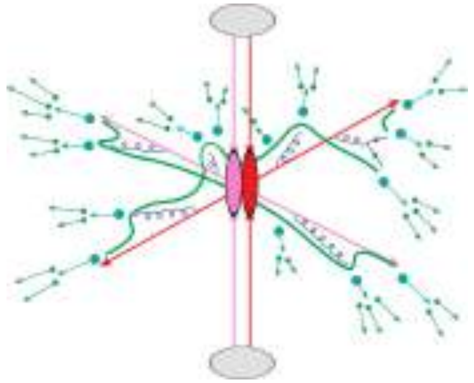
  ➔ **Share calculations**!

- **Standardisation**: boring but important
  - Event format conventions, statuses, PDG particle numbering, weights…

Rivet applications

Lightweight analysis preservation has **spurred many other exp./pheno activities**

- MC development
- Tuning
- PDF studies, EFT studies, global BSM fits…
- Heavy-ion methods
- …



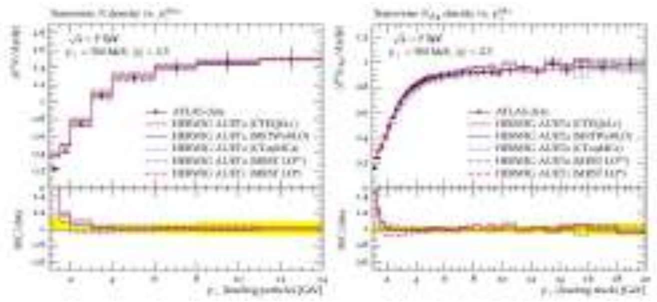
Preservation is an **accelerator for analysis impact**:
experiment-theory studies, fun collaborations!

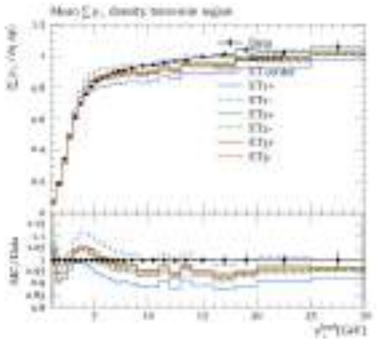
Rivet – Tuning history

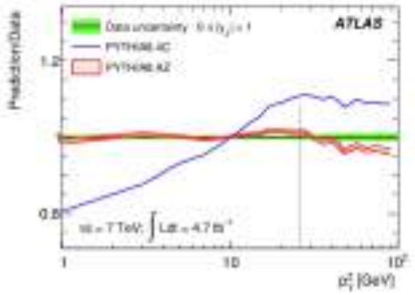
- Pre-LHC, soft QCD uncertainties were *huge*
  - 200% uncertainty on 7 TeV σ_{tot}!
- Feed in to underlying event, pile-up, etc.

➔ Tuning **essential task**:
Better tunes → Better analysis designs → Better limits, ...

➔ **Impact**: LEP+Tevatron analyses published for ~10 years suddenly used – and cited...
  - ATLAS AMBT, AUET, AZ, A14 etc. tunes + CMS
  - **Rapid responses** to preliminary data, changes of model
    - E.g. Py8 for ATLAS pile-up
  - **Model development**: matching & merging, addition of energy evolution & colour-reconnection to Herwig, ...





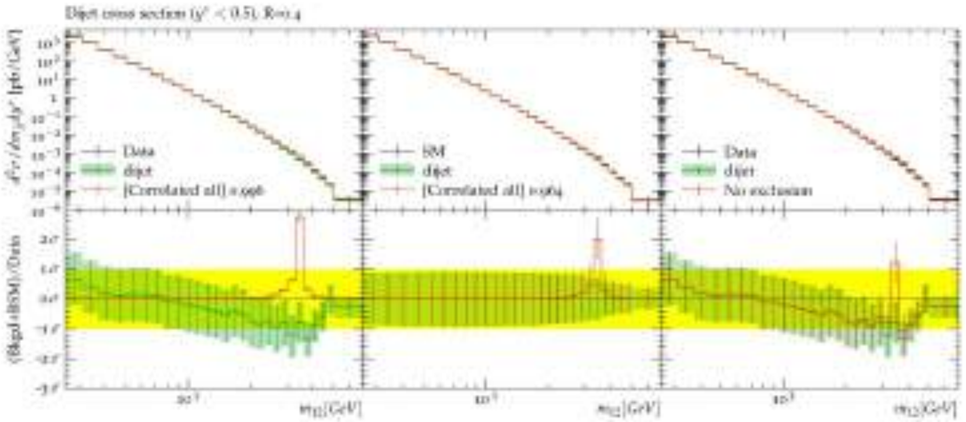


Digression: Contur toolkit

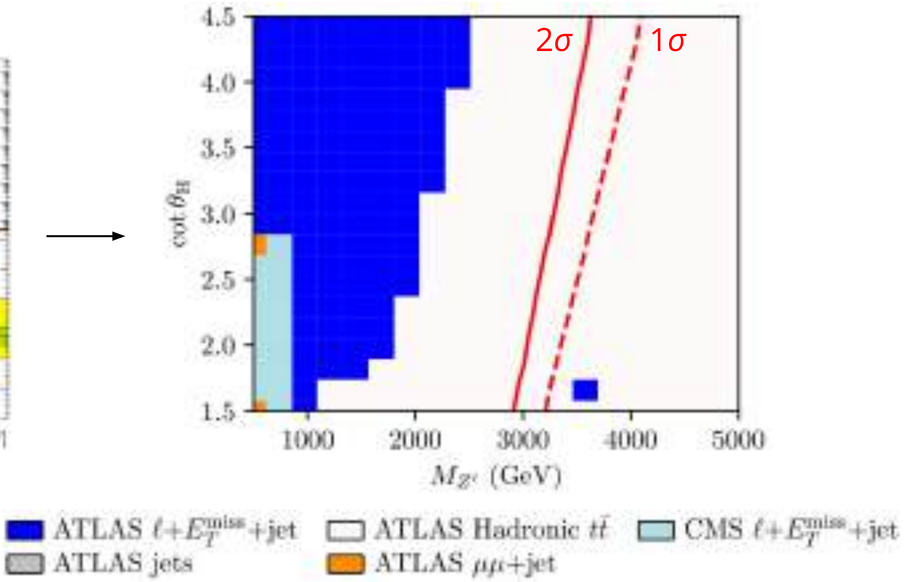
→Paper 1, →Paper 2, →Webpage

- Use Rivet for beyond-Standard-Model (**BSM**) **reinterpretation**
- **Automatic signal-injection** of BSM events onto measured Rivet cross sections
- **Extend known exclusion** reach

Example: →Contur on Top-colour model



$M_{Z'}$ = 3.6 TeV, $\cot\theta_H$ = 4.5



Rivet and BSM-search recasting I

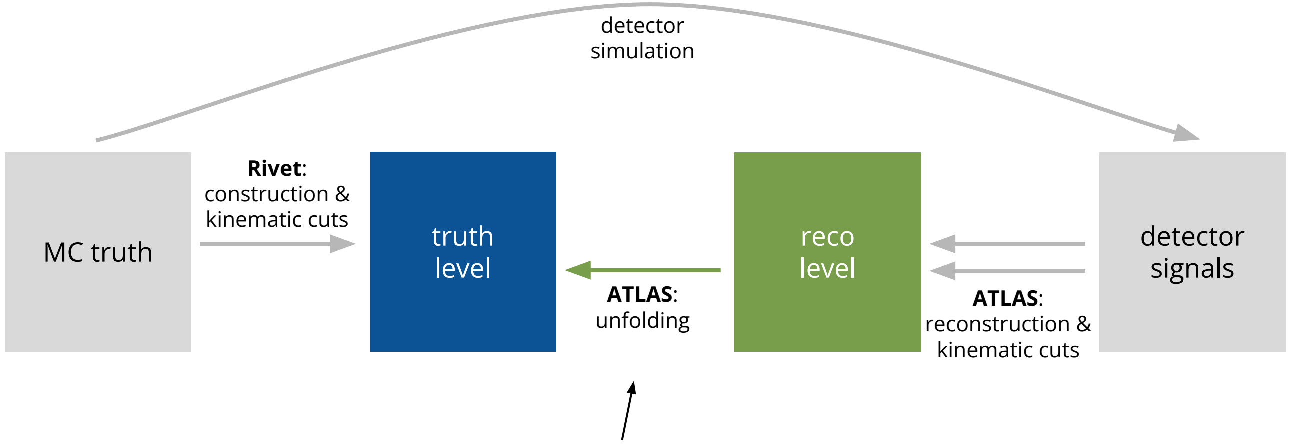
- Rivet's **main emphasis *isn't* BSM** direct searches, **but no reason not to**
  - Lots of experiment experience and support
  - Efficient scaling-up to hundreds of analyses, with distinct phase-space specific detector/efficiency functions

➔ Can we do for BSM preservation what we did for measurement analyses?

- Friendly competition, mainly from/with MadAnalysis5
  - All good tools, all geared to getting your analysis into pheno studies asap
  - But ours is best, obviously… ;-)

Rivet and BSM-search recasting II

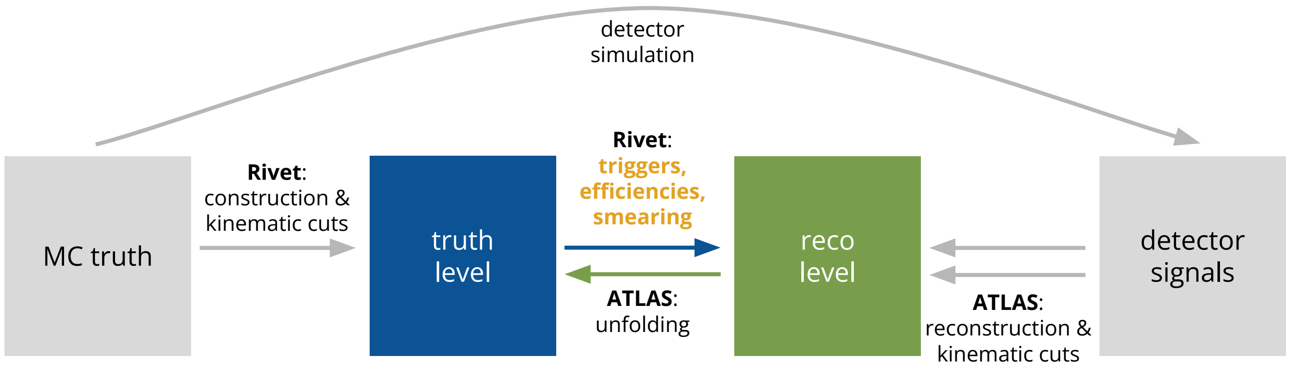
- Measurements in Rivet so far:



- Rivet helped define **unfolding targets**
- Today: "Rivet = level that is unfolded to"

Rivet and BSM-search recasting II

- Detector smearing built on Rivet's projection system — for **reco-level analyses**
  - Developed based on Gambit ColliderBit experience: No need for "full fast-sim"

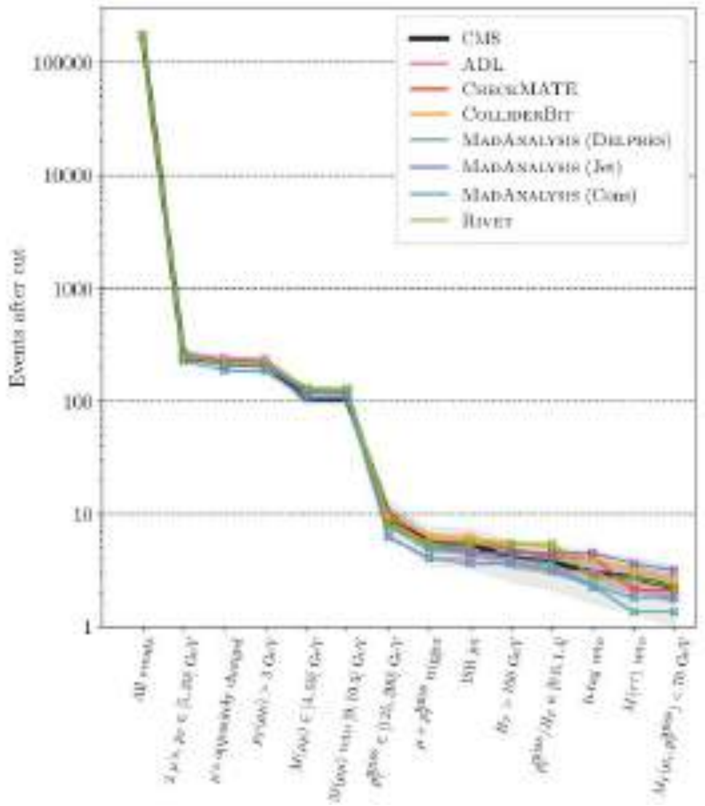

detector simulation
MC truth
Rivet: construction & kinematic cuts
truth level
Rivet: triggers, efficiencies, smearing
ATLAS: unfolding
reco level
ATLAS: reconstruction & kinematic cuts
detector signals

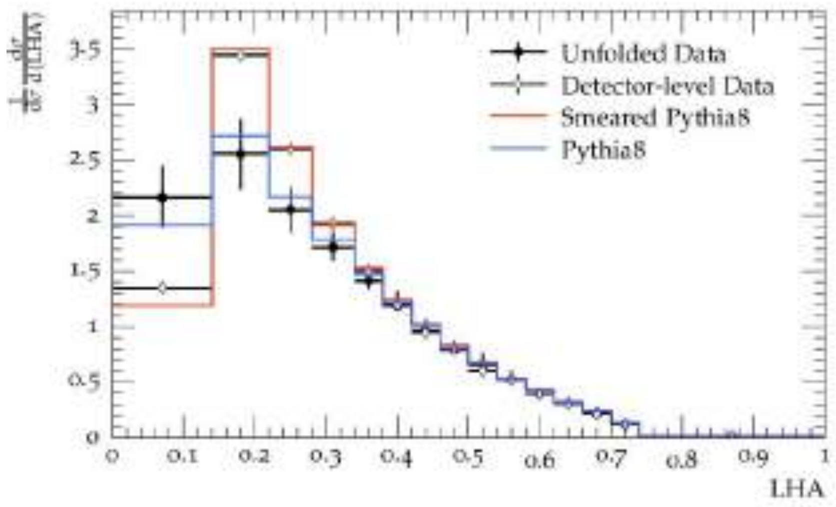
- Like Delphes, but **more flexible** & can be ***analysis-specific***

➔ MadAnalysis5 "SFS" mode

Rivet and BSM-search recasting III

- Flexibility allows e.g. “tuned” jet-substructure smearing
- Excellent performance:





→Les Houches 2019 CMS soft-lepton recasting-tools comparison

Rivet 4 – New features

- Now requires **C++17**, drops support for HepMC2 and Python2
- Adopts **YODA2** for histogramming backend
- **matplotlib-based plotting machinery** now default
- Renamings:
  - DressedLeptons → LeptonFinder
  - ZFinder → DileptonFinder
  - WFinder X
- **Thread safety**: Better support for massively parallel applications
- Interface to **HDF5** for auxiliary data files
- New **interface to ONNX Runtime** as (current) best option for ML preservation
- More in →backup

Rivet and machine learning (ML)

- Extensive recommendations in →Les Houches guide

Analysis design

- Stable, open-source tool for ML → ideally preserve in **ONNX**
  - Alternatively: Executable code with minimal dependencies
- Neural nets should be **compilable** (C++ > Python) for speed
- **Avoid over-complex** network designs without gain
  - Minimise customised layers, custom activation functions, ...
- **Avoid** features heavily depending on **detector/ reconstruction details**
  - Unavoidable? → **Detailed effiency maps** or **surrogate network** using less det-sensitive inputs

Rivet and machine learning (ML)

- Extensive recommendations in →Les Houches guide

Documentation & validation: Provide …

- **Exact definitions** of network inputs
  - Including units, ordering, conventions, …
- **Sample input features** and **output values**
  - Add plots of these (with feature importance, if possible)
- **Cut-flow info** before and after ML selection

$\}$

➔ **Validated** and **runnable published analysis code**

➔ **Rivet!**

- **Full description of physics models**
  - E.g. SLHA files, generator run cards, …

➔ More info needed to understand "black box" ML
  - ➔ Establish pipeline?

➔ **Available information >> polished information**

Rivet and machine learning (ML)

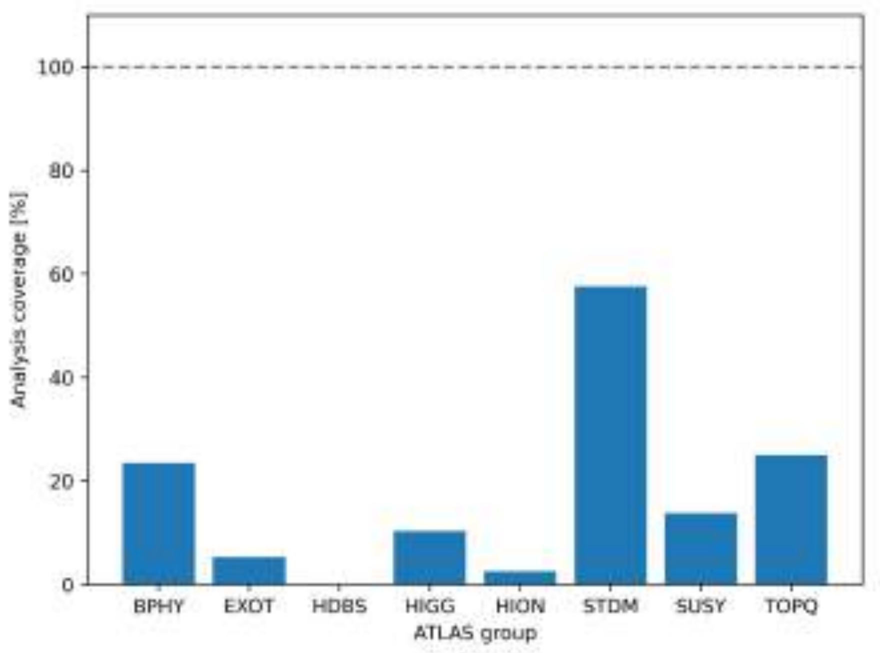
- Extensive recommendations in →Les Houches guide

Analysis mentioned here → People are using it!

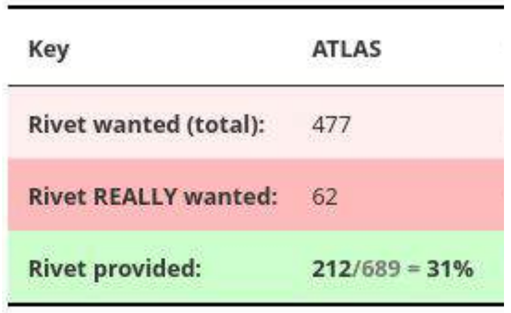
Examples

- SUSY squarks and gluinos (→ATLAS-SUSY-2018-22)
  - Given: BDTs in TMVA XML, code snippet, BDT score distributions
  - Good!
- SUSY *R*-parity violation (→ATLAS-SUSY-2019-04)
  - Given: Trained net in ONNX, using high-level variables, truth-level implementation
  - Needed: Instructions on reproducing *b*-tag score, validation distributions
- SUSY *b*-jets+E_T^{miss} (→ATLAS-SUSY-2018-30)
  - Given: Trained net in ONNX, SimpleAnalysis code with variables, NN scores, binary *b*-tagging
  - Needed: NN return scores in SimpleAnalysis
- EXOT CalRatio LLP (→ATLAS-EXOT-2019-23)
  - Given: Neural net in ONNX, C++/Python BDTs, low-level maps but 6D efficiency maps
  - Good!
- HDBS anomaly detection (→ATLAS-HDBS-2019-23)
  - Given: Post-training weights of NN in PyTorch file, Python code
  - Needed: Description of input/ output variables, description of code usage

The future of Rivet



→ATLAS Rivet analysis coverage



Key	ATLAS
Rivet wanted (total):	477
Rivet REALLY wanted:	62
Rivet provided:	212/689 = 31%

→Full Rivet analysis coverage as of 2024-09-25

➔ **And now you!**

Rivet: Hands-on tutorial

Rivet and Contur tutorials

- In the following: Step-by-step **Rivet tutorial**
- If more interested into **BSM reinterpretation**:
  Follow **Contur tutorial** $\rightarrow$here, let us know about any questions!

Environment Setup

If you have a local Rivet Installation: Skip to next page

Otherwise setup environment using Docker image:

```
docker pull hepstore/rivet-tutorial:4.0.1
docker container run -it -v $PWD:$PWD -w $PWD hepstore/rivet-tutorial:4.0.1 /bin/bash
```

You can test if environment is setup successfully by running:

```
rivet --version
```

Which should print 4.0.1

depending on your setup, you might have to call always docker with superuser rights:

```
sudo docker <your commands>
```

(that’s bad, we know :( )

Generating events with MG5

The first step will be generating events using MadGraph 5 inside this docker image.

```
//Initiate the Madgraph 5 script
/work/MG5_aMC_v3_5_5/bin/mg5_aMC
```

Then you should see the MG5 command line:

```
MG5_aMC>
```

For people new to MG5, you can get an introductory tutorial via running

```
MG5_aMC> tutorial
```

The aim of this tutorial is to show the general workflow chain and will not go into too much details of MG5 generation settings, for those interested check here for a full MG5 tutorial from MCnet School 2024.

Generating events with MG5

Run through these commands to get some hepmc files out for later use
Let’s first generate the muon distributions

```
//Generate cross-section and invariant mass distribution for two muons
MG5_aMC> generate p p > mu+ mu-
//output these into a directory
MG5_aMC> output pp_mumu_out
//launch generation
MG5_aMC> launch pp_mumu_out

//It will then ask you for settings of shower/detector/analysis etc
//Turn on the Pythia8 Shower
MG5_aMC> 1
// Then type 0 or done for the following setting questions.
// The output .hepmc file should be located at pp_mumu_out/Events/run_01/
// After everything is finished, you can quit the MG interactive mode by:
MG5_aMC> quit
```

Then repeat for e+e- generation. So in total you should get two `.hepmc.gz` files.

Getting started with Rivet

Starting a fresh Rivet routine:

```
rivet-mkanalysis MY_ROUTINE
```

This will create a `MY_ROUTINE.cc` file for routine skeleton, a `MY_ROUTINE.plot` file for plot formatting, and a `MY_ROUTINE.info` file for storing metadata about the analysis.

The main body of the rivet routine is within this `.cc` file, which consists:

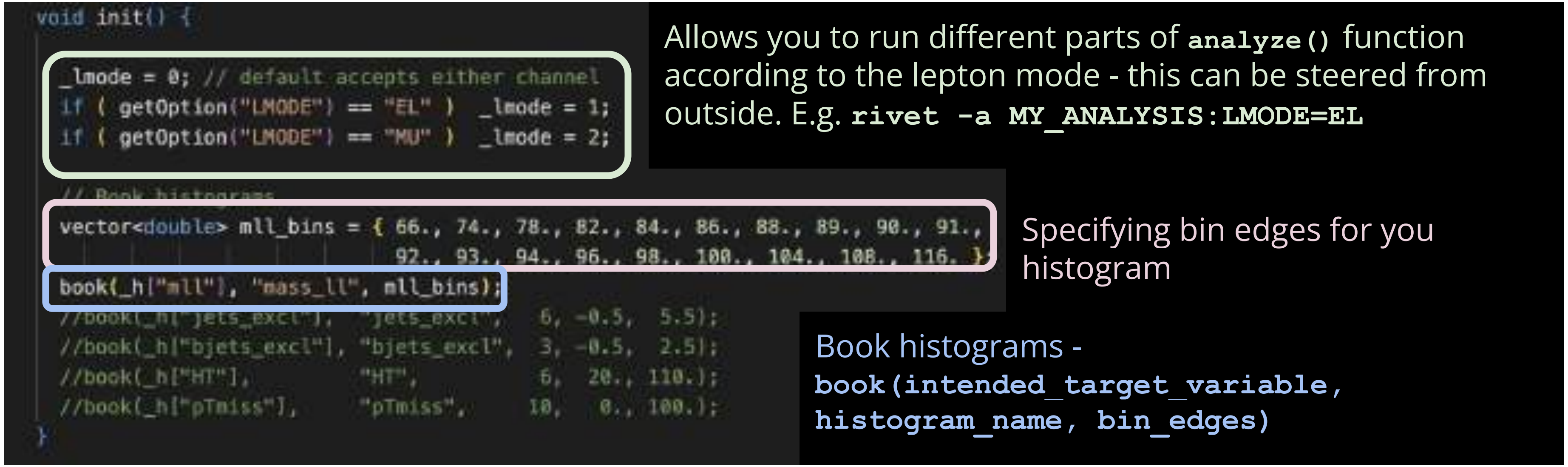
1. An `init()` method that runs onces at the beginning of the run to initialise the routine
2. An `analyze()` method that is executed for every single event
3. A `finalize()` method that is called once at the end to do post-analysis operations such as scaling the histograms

Getting started with Rivet

An example routine is prepared at the gitLab repository - copy this into your local working area

```
git clone https://gitlab.cern.ch/mhabedan/rivet-tutorial.git
```

It contains an example analysis file **`MY_ANLAYSIS.cc`**, and we'll start from there:



```
void init() {

    _lmode = 0; // default accepts either channel
    if ( getOption("LMODE") == "EL" )  _lmode = 1;
    if ( getOption("LMODE") == "MU" )  _lmode = 2;

    // Book histograms
    vector<double> mll_bins = { 66., 74., 78., 82., 84., 86., 88., 89., 90., 91.,
                                92., 93., 94., 96., 98., 100., 104., 108., 116. };
    book(_h["mll"], "mass_ll", mll_bins);
    //book(_h["jets_excl"],  "jets_excl",   6, -0.5,  5.5);
    //book(_h["bjets_excl"], "bjets_excl",  3, -0.5,  2.5);
    //book(_h["HT"],         "HT",          6,  20., 110.);
    //book(_h["pTmiss"],     "pTmiss",     10,   0., 100.);
}
```

Allows you to run different parts of **`analyze()`** function according to the lepton mode - this can be steered from outside. E.g. **`rivet -a MY_ANALYSIS:LMODE=EL`**

Specifying bin edges for you histogram

Book histograms - **`book(intended_target_variable, histogram_name, bin_edges)`**

Getting started

- The current `analyze()` function is only filling the histogram with 1 - need to be changed later.
- `finalize()` method is scaling the histogram by cross_section/Sum_of_weight, where the cross section by default will be the one extracted from HepMC file by Rivet. But this can be customized using `--cross-section`

To Run this routine, run in command line:

```
//Rivet provides a wrapper script to add the relevant compiler flags
//Rivet*.so is the canonical form of the compiled Rivet plugin

rivet-build RivetMY_ANALYSIS.so MY_ANALYSIS.cc
//Can also compile several routines into a single shared object library:
//rivet-build RivetUBER.so ROUTINE1.cc ROUTINE2.cc …

//Add the directory containing your routine to the rivet analysis path
export RIVET_ANALYSIS_PATH=$PWD:$RIVET_ANALYSIS_PATH

//Run the routine over a hepmc file
rivet -a MY_ANALYSIS input.hepmc.gz
```

Use the hepmc files you just generated or get them from →CERNBox

Projection Examples

Before starting the first exercise:

Projection: specify kinematic requirements on particles that should be considered in the calculation. This is declared in init() method and applied in analyze()

- **Example 1: FinalState** - Providing access to all final-state particles

```
#include “Rivet/Projections/FinalState.hh” //Make sure the header file is included
…
…
void init(){ // Declare the Projection here
     …
     FinalState fs;
     declare(fs, “my_first_projection”);
}

void analyze(){
     …
     const FinalState& p1 = apply<FinalState>(event, “my_first_projection”);
     const Particles& fs_particles = p1.particlesByPt();
     …
}
```

Takes event argument and apply FinalState to it, saving the result as p1

Using p1 to retrieve a set of all final-state particles ordered by transverse momentum (hard to soft)

Projection Examples

- **Example 2: FinalState** with additional cuts

```
#include “Rivet/Projections/FinalState.hh” //Make sure the header file is included
#include “Rivet/Projections/PromptFinalState.hh”
…
…
void init(){ // Declare the Projection here
     …
     // require a charged particle
     FinalState charged_tracks(Cuts::charge != 0);
     // require a charged particle + additional pseudorapidity cut
     FinalState IDtrack(Cuts::charge != 0 && Cuts::abseta < 2.5);
     // All muons with transverse momentum >20GeV
     FinalState allMuons(Cuts::abspid == PID::MUON && Cuts::pT > 20*GeV);

     // Using the previous muon cut result to get a subset of prompt muons
     PromptFinalState promptMuons(allMuons);
     declare…
}
```

Projection Examples

The **`LeptonFinder`** projection in Rivet allows you to dress the bare-lepton four-momentum with all photon four-momenta within a given cone size.

These are referred as **dressed-level leptons.**

- **Example 3: `LeptonFinder`**

```
#include “Rivet/Projections/LeptonFinder.hh”
…
void init(){ // Declare the Projection here
     …
     // First define the prompt Final State photons and electrons
     FinalState photons(Cuts::abspid == PID::PHOTON);
     PromptFinalState electrons(Cuts::abspid == PID::ELECTRON);

     // Define some cuts on electrons
     Cut electron_cut = (Cuts::pT > 7*GeV) && ( Cuts::abseta < 2.47 );

     // Define dressed electrons
     LeptonFinder dressed_electrons(electrons, photons, 0.1, electron_cut);
     declare…
}
```

$\searrow$ Cone size set to 0.1

Exercise 1

Now Practice with the Exercise - Modify the .cc file so that it:

1. **Selects exactly one same-flavour opposite-charge lepton pair, with transverse momentum at least 10 GeV and lie within 2.5 in pseudo-rapidity (`Cuts::abseta`)**

2. **Selects an electron or muon pair depending on the value of _lmode**

3. **Use selected dressed lepton pairs to reconstruct the dilepton invariant mass and pass into the prepared histogram. Helpful: → Rivet documentation**

4. **Compile the routine (also write an .info file), run over the provided HepMC events using both lepton-flavour options in the same run**

You will have a .yoda file as an output. To see the histograms from your output file, run

```
rivet-mkhtml pp_ee_File.yoda pp_mumu_File.yoda
```

Digression: YODA

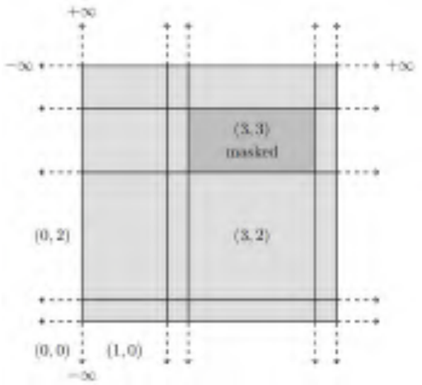
Histograms: powerful tool, often taken for granted

→ Enter **YODA**: →Paper, →Webpage

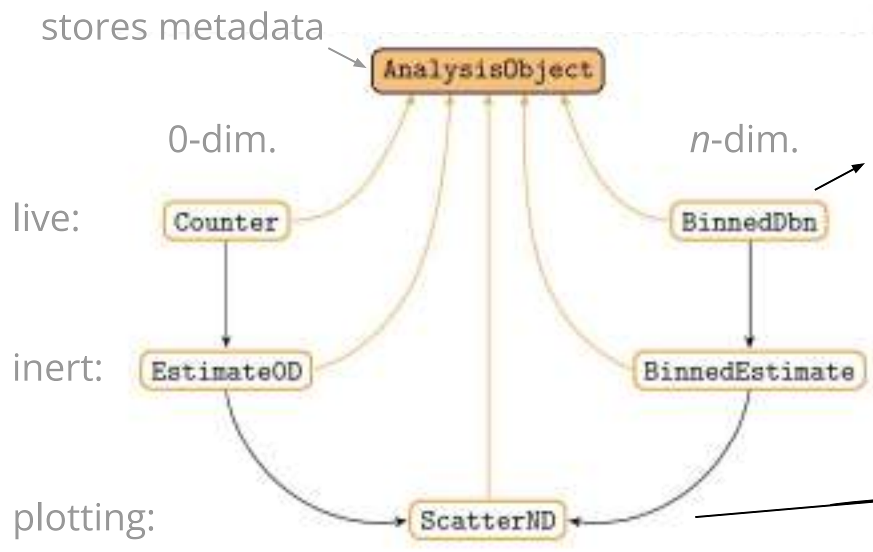
- **Lightweight and general purpose library for binned statistical data analysis**
  - Statistically sound
  - Tracks uncertainties and correlations across distributions

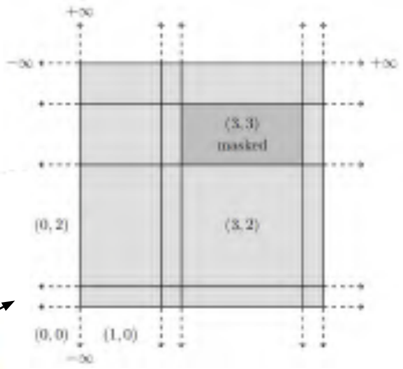
- First released in 2013, re-designed ~2023 → **v2!**
- Based on templated C++, also Python interface

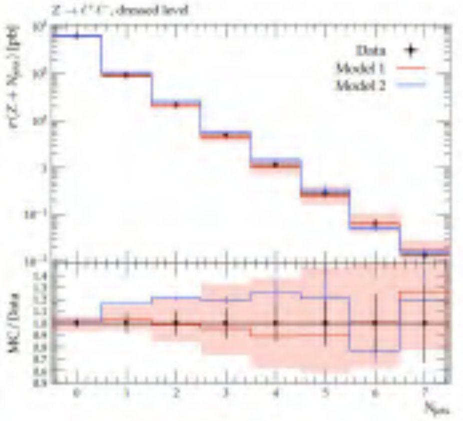
- Histogramming library for **Rivet**, also used by HEPData

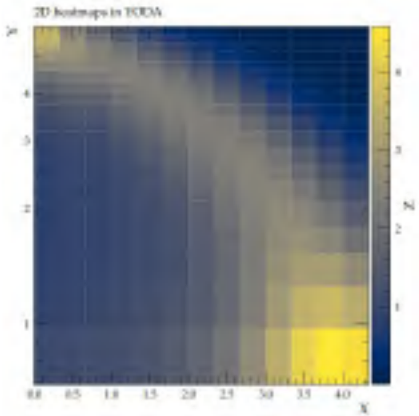


Digression: Basic principle of YODA



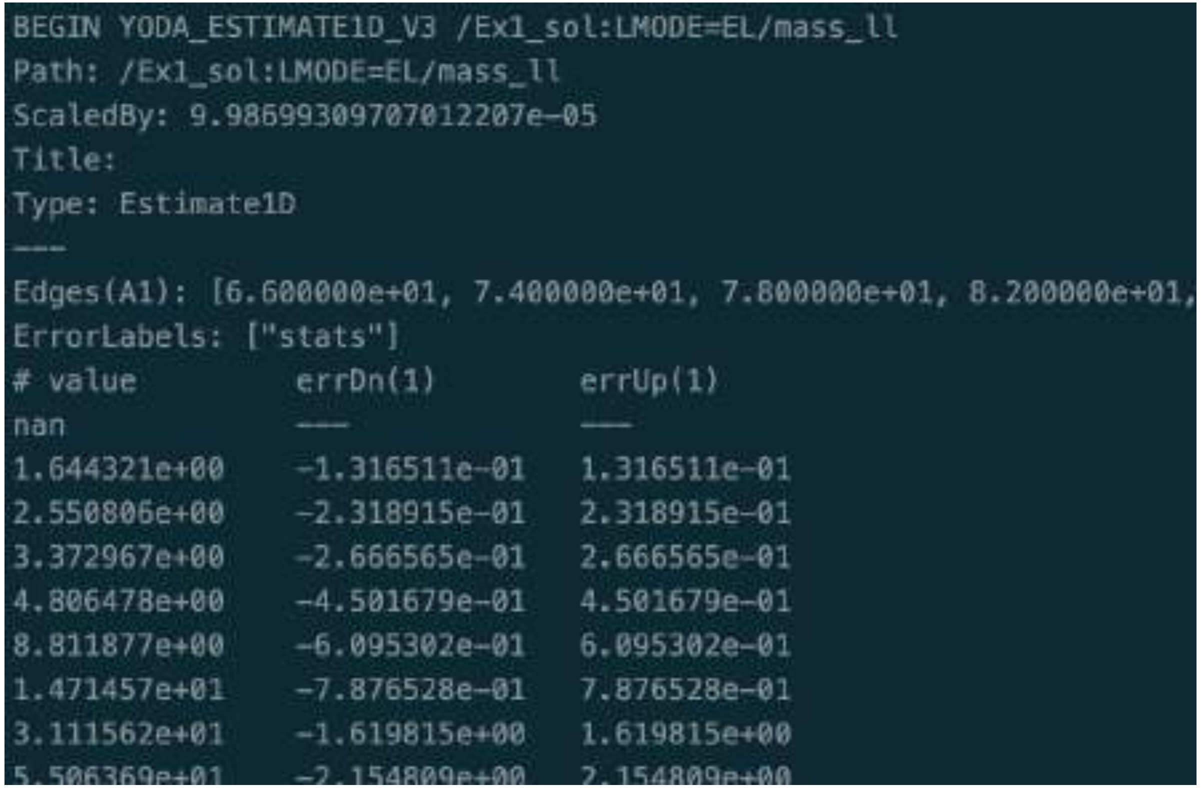






Example of Output Files

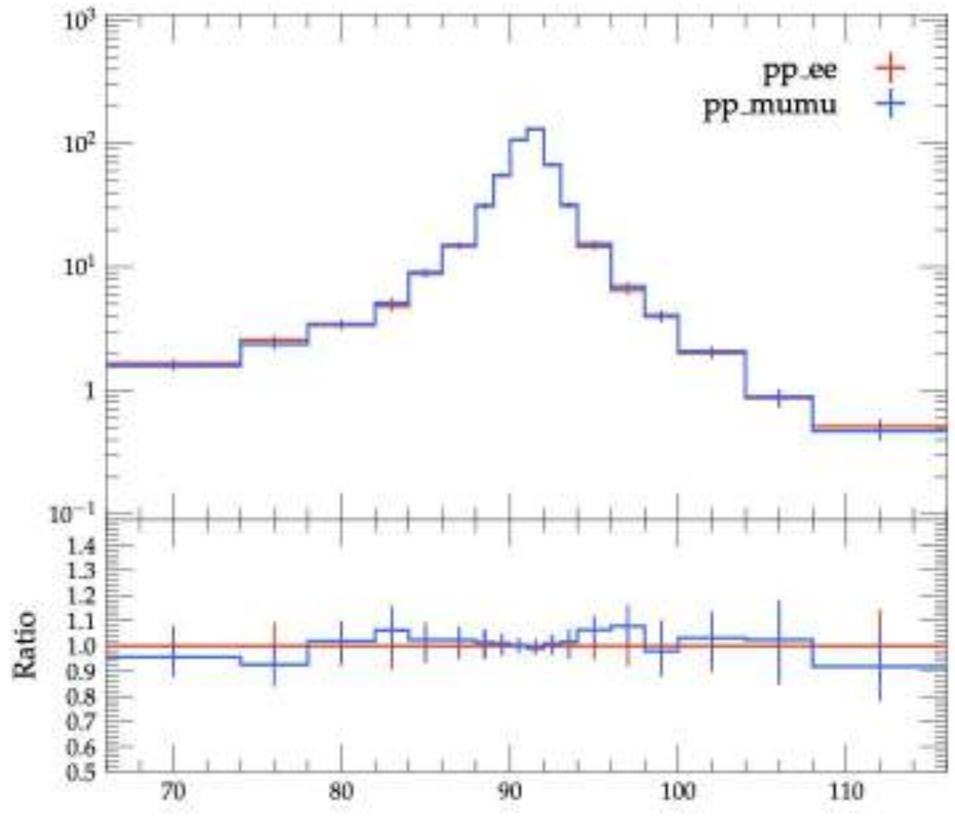
The output .yoda file should look like this:



```
BEGIN YODA_ESTIMATE1D_V3 /Ex1_sol:LMODE=EL/mass_ll
Path: /Ex1_sol:LMODE=EL/mass_ll
ScaledBy: 9.98699309707012207e-05
Title:
Type: Estimate1D
---
Edges(A1): [6.600000e+01, 7.400000e+01, 7.800000e+01, 8.200000e+01,
ErrorLabels: ["stats"]
# value        errDn(1)        errUp(1)
nan            ---             ---
1.644321e+00   -1.316511e-01   1.316511e-01
2.550806e+00   -2.318915e-01   2.318915e-01
3.372967e+00   -2.666565e-01   2.666565e-01
4.806478e+00   -4.501679e-01   4.501679e-01
8.811877e+00   -6.095302e-01   6.095302e-01
1.471457e+01   -7.876528e-01   7.876528e-01
3.111562e+01   -1.619815e+00   1.619815e+00
5.506369e+01   -2.154809e+00   2.154809e+00
```

Exercise 1

Plot should look like this: Solutions here



Projection Examples

- **Example 4: `FastJets`**

```
#include “Rivet/Projections/FastJets.hh” // Make sure relevant header fils is included
…
void init(){ // Declare the Projection here
     …
     // First define the prompt Final State particles and apply jet algorithms
     FinalState fs;
     FastJets jets(fs, JetAlg::ANTIKT, 0.4, JetMuons::NONE, JetInvisibles::NONE);

     declare(jets, “FJets”)
}

void analyze(){
     …
     // Apply jetsByPt method in analyze section
     const Jets& all_jets = apply<FastJets>(event, “FJets”).jetsByPt();
     …
}
```

Jet Algorithm

Jet Radius Parameter

Dealing with muons and invisible particles in the clustering (`NONE, DECAY, ALL`)

Exercise 2

Modify the .cc file so that it:

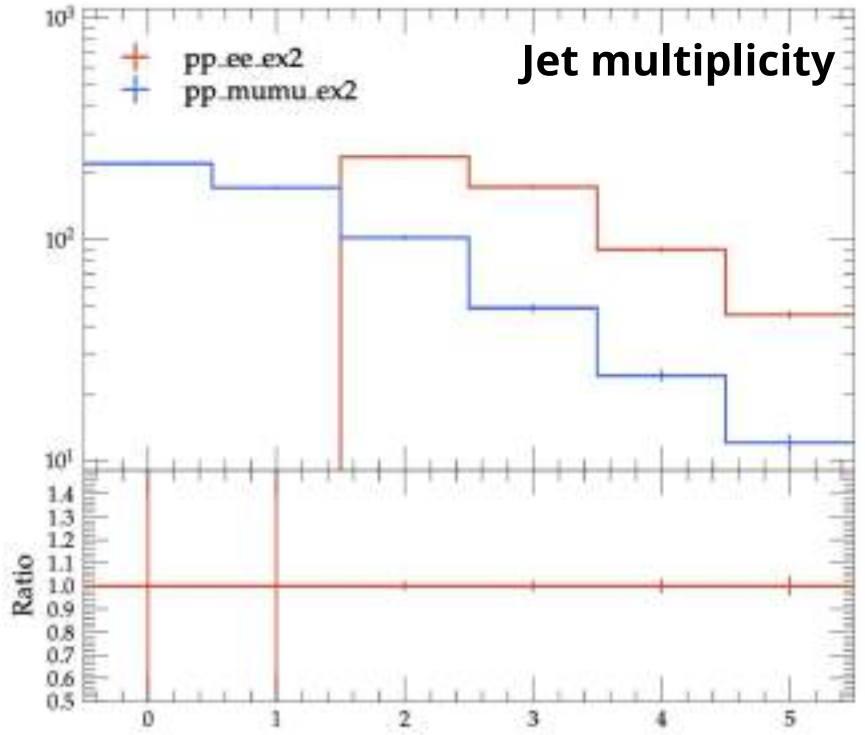
1. **Selects events with 66 GeV < m_{ll} < 116 GeV using dressed-level leptons**
2. **Use anti-kT algorithm with a jet-radius parameter of 0.4, cluster all final state particles within |η| < 4.9, except for muons and invisible particles.**
3. **Select all jets with pT > 10GeV and |y| < 4.5 (`Cuts::absrap`).**
4. **Count the number of jets and plot exclusive jet multiplicity.**
5. **Plot also the scalar jet pT sum**
6. **Run for each lepton channel and compare the curves**

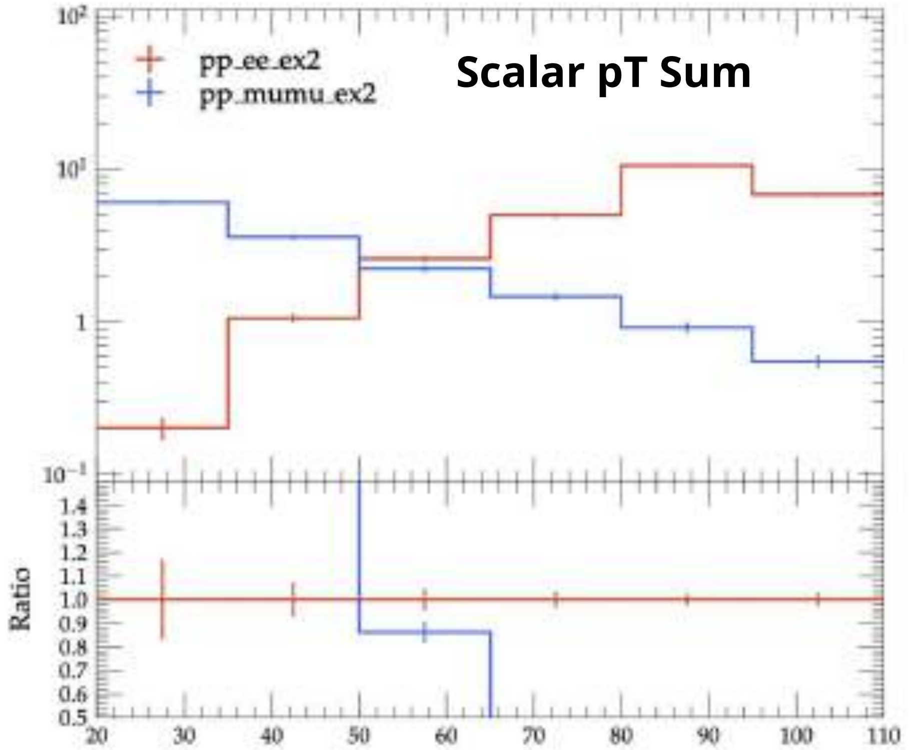
You may find Rivet's built-in sum function helpful

```
sum(Jets, Kin::pT, 0.0)
```

Exercise 2

Plots for Ex2: difference between two channels seen as only muons are removed in the Anti-kT algorithm. Solutions here

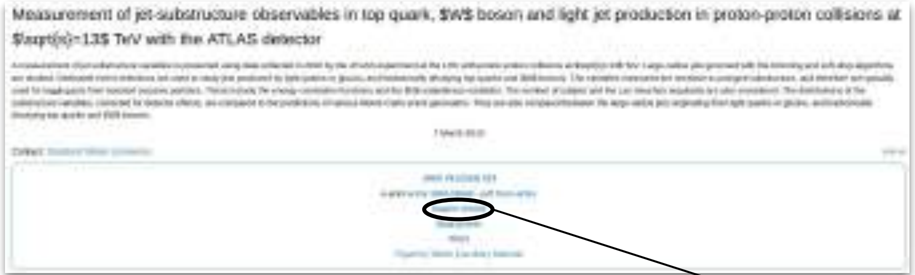




Writing your own Rivet routine

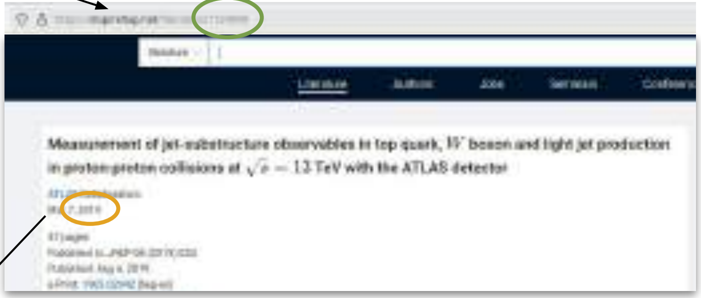
- Imitation is the highest form of flattery: copy an existing analysis!

Example:


Measurement of jet-substructure observables in top quark, W boson and light jet production in proton-proton collisions at $\sqrt{s}=13$ TeV with the ATLAS detector

[illegible]

→ATLAS webpage


Literature Authors Jobs Seminars Conferences

Measurement of jet-substructure observables in top quark, W boson and light jet production in proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector

ATLAS Collaboration

Jul 7, 2019

47 pages

Published in: JHEP 08 (2019) 033

Published: Aug 6, 2019

e-Print: 1903.02942 [hep-ex]

→inspirehep.net/literature/1724098

Rivet analyses reference

ATLAS_2019_I1724098

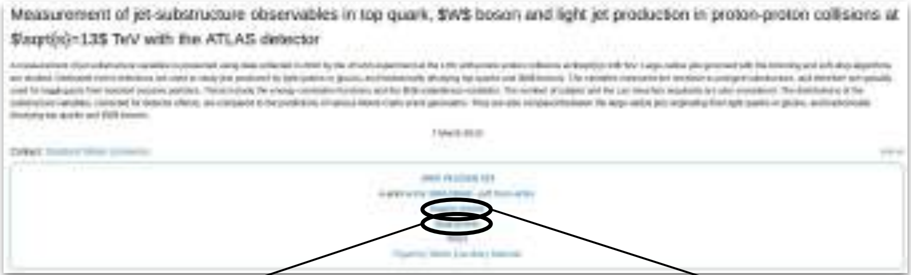
[illegible]

→rivet.hepforge.org/analyses/ATLAS_2019_I172409
8

Writing your own Rivet routine

- Imitation is the highest form of flattery: copy an existing analysis!

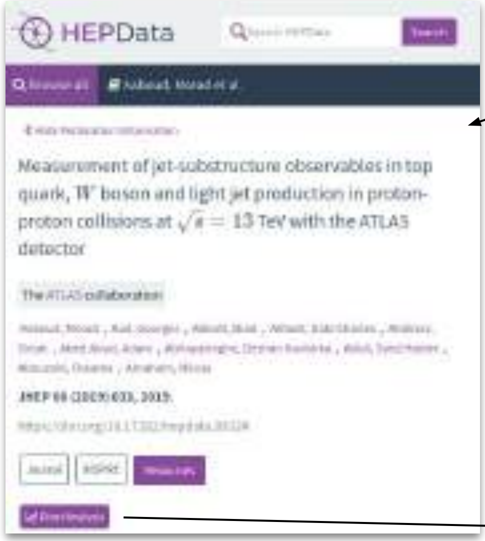
Example:



Measurement of jet-substructure observables in top quark, W boson and light jet production in proton-proton collisions at $\sqrt(s)=13$ TeV with the ATLAS detector

[illegible]

→ATLAS webpage



Measurement of jet-substructure observables in top quark, W boson and light jet production in proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector

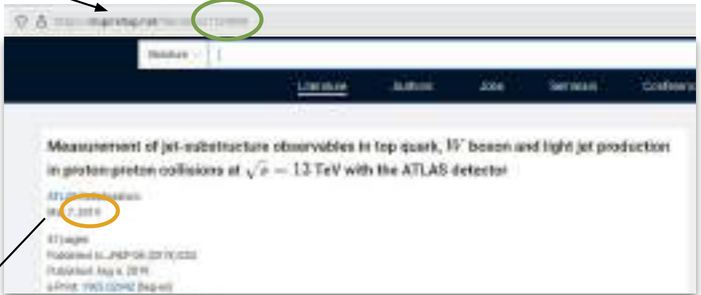
The ATLAS collaboration

JHEP 08 (2019) 033, 2019.

→HEPData record

[illegible]

→rivet.hepforge.org/analyses/ATLAS_2019_I1724098



Measurement of jet-substructure observables in top quark, W boson and light jet production in proton-proton collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector

ATLAS Collaboration

Mar 7, 2019

47 pages

Published in: JHEP 08 (2019) 033

Published: Aug 6, 2019

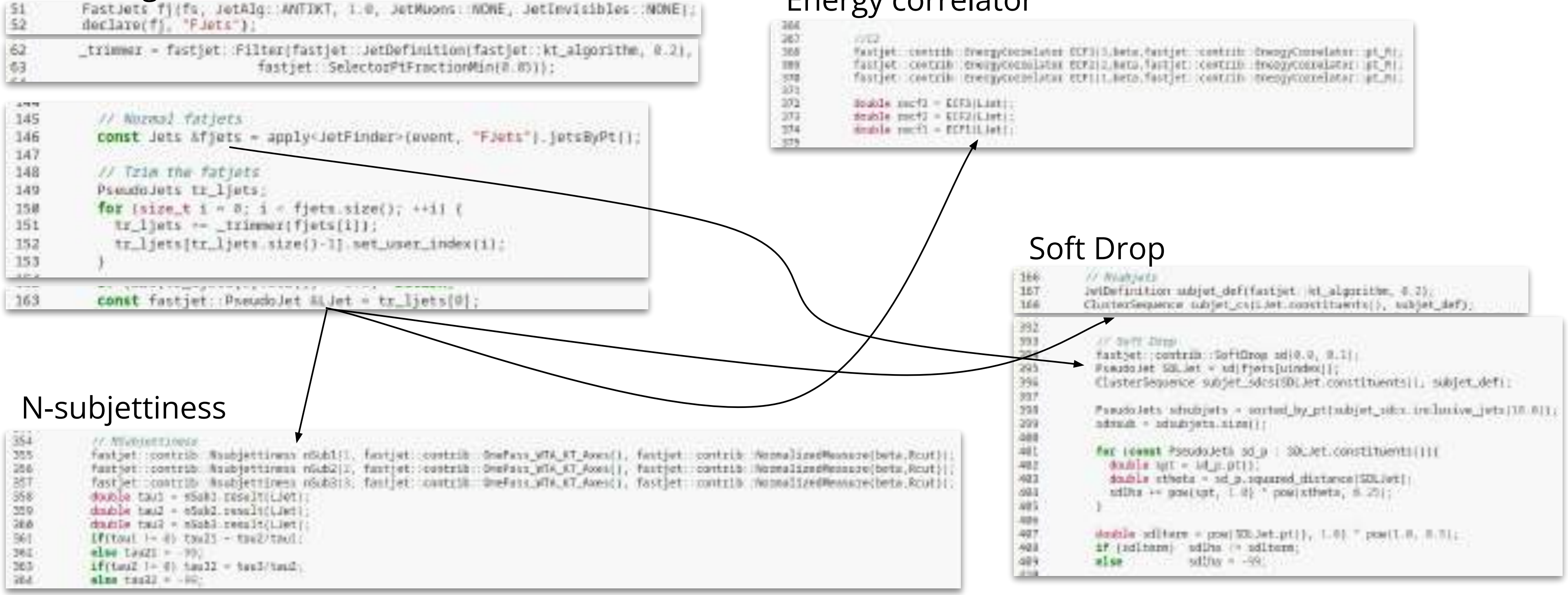
e-Print: 1903.02942 [hep-ex]

→inspirehep.net/literature/1724098

Writing your own Rivet routine

- Imitation is the highest form of flattery: copy an existing analysis!

Example: Jet substructure at 13 TeV →rivet.hepforge.org/analyses/ATLAS_2019_I1724098



Trimming

```cpp
      FastJets fj(fs, JetAlg::ANTIKT, 1.0, JetMuons::NONE, JetInvisibles::NONE);
      declare(fj, "FJets");
```

```cpp
      _trimmer = fastjet::Filter(fastjet::JetDefinition(fastjet::kt_algorithm, 0.2),
                                 fastjet::SelectorPtFractionMin(0.05));
```

```cpp
      // Normal fatjets
      const Jets &fjets = apply<JetFinder>(event, "FJets").jetsByPt();

      // Trim the fatjets
      PseudoJets tr_ljets;
      for (size_t i = 0; i < fjets.size(); ++i) {
        tr_ljets += _trimmer(fjets[i]);
        tr_ljets[tr_ljets.size()-1].set_user_index(i);
      }
```

```cpp
      const fastjet::PseudoJet &LJet = tr_ljets[0];
```

Energy correlator

```cpp

      //EC2
      fastjet::contrib::EnergyCorrelator ECF3(3,beta,fastjet::contrib::EnergyCorrelator::pt_R);
      fastjet::contrib::EnergyCorrelator ECF2(2,beta,fastjet::contrib::EnergyCorrelator::pt_R);
      fastjet::contrib::EnergyCorrelator ECF1(1,beta,fastjet::contrib::EnergyCorrelator::pt_R);

      double recf3 = ECF3(LJet);
      double recf2 = ECF2(LJet);
      double recf1 = ECF1(LJet);

```

Soft Drop

```cpp
      // Rsubjets
      JetDefinition subjet_def(fastjet::kt_algorithm, 0.2);
      ClusterSequence subjet_cs(LJet.constituents(), subjet_def);
```

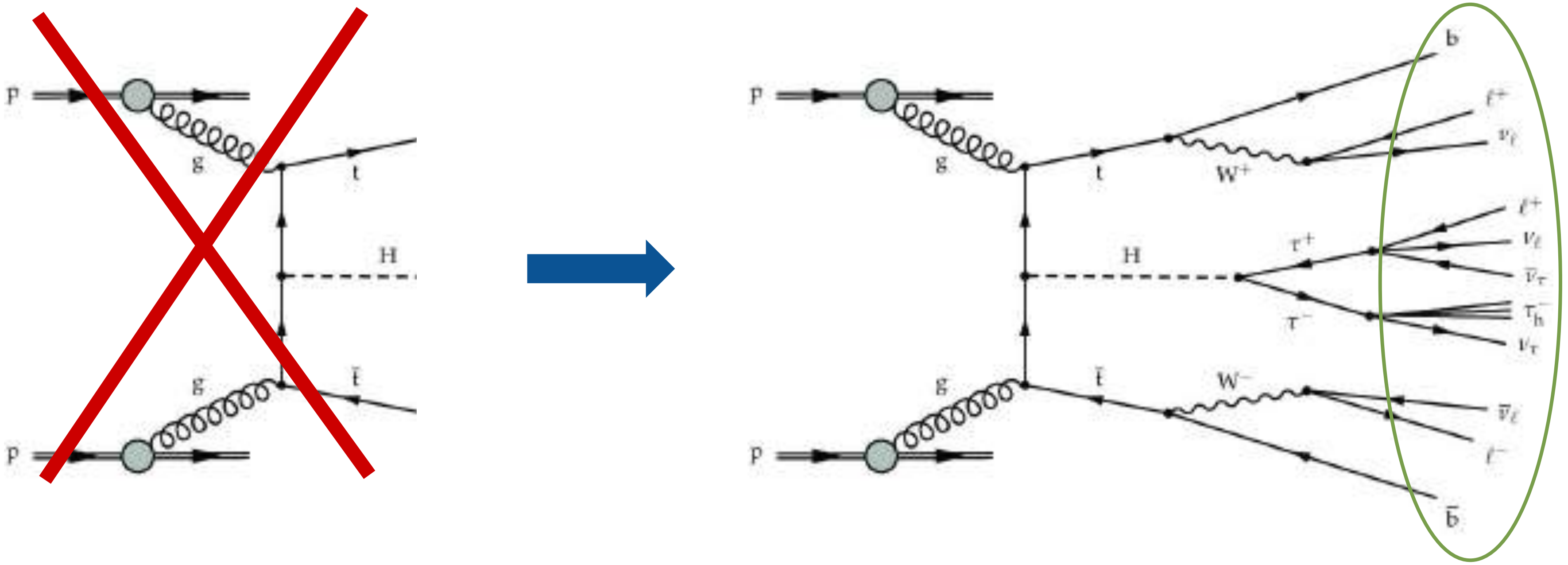
```cpp

      // Soft Drop
      fastjet::contrib::SoftDrop sd(0.0, 0.1);
      PseudoJet SDLJet = sd(fjets[uindex]);
      ClusterSequence subjet_sdcs(SDLJet.constituents(), subjet_def);

      PseudoJets sdsubjets = sorted_by_pt(subjet_sdcs.inclusive_jets(10.0));
      sdnsub = sdsubjets.size();

      for (const PseudoJet& sd_p : SDLJet.constituents()){
        double spt = sd_p.pt();
        double stheta = sd_p.squared_distance(SDLJet);
        sdlhs += pow(spt, 1.0) * pow(stheta, 0.25);
      }

      double sdlterm = pow(SDLJet.pt(), 1.0) * pow(1.0, 0.5);
      if (sdlterm)  sdlhs /= sdlterm;
      else          sdlhs = -99;
```

N-subjettiness

```cpp
      // NSubjettiness
      fastjet::contrib::Nsubjettiness nSub1(1, fastjet::contrib::OnePass_WTA_KT_Axes(), fastjet::contrib::NormalizedMeasure(beta,Rcut));
      fastjet::contrib::Nsubjettiness nSub2(2, fastjet::contrib::OnePass_WTA_KT_Axes(), fastjet::contrib::NormalizedMeasure(beta,Rcut));
      fastjet::contrib::Nsubjettiness nSub3(3, fastjet::contrib::OnePass_WTA_KT_Axes(), fastjet::contrib::NormalizedMeasure(beta,Rcut));
      double tau1 = nSub1.result(LJet);
      double tau2 = nSub2.result(LJet);
      double tau3 = nSub3.result(LJet);
      if(tau1 != 0) tau21 = tau2/tau1;
      else tau21 = -99;
      if(tau2 != 0) tau32 = tau3/tau2;
      else tau32 = -99;
```

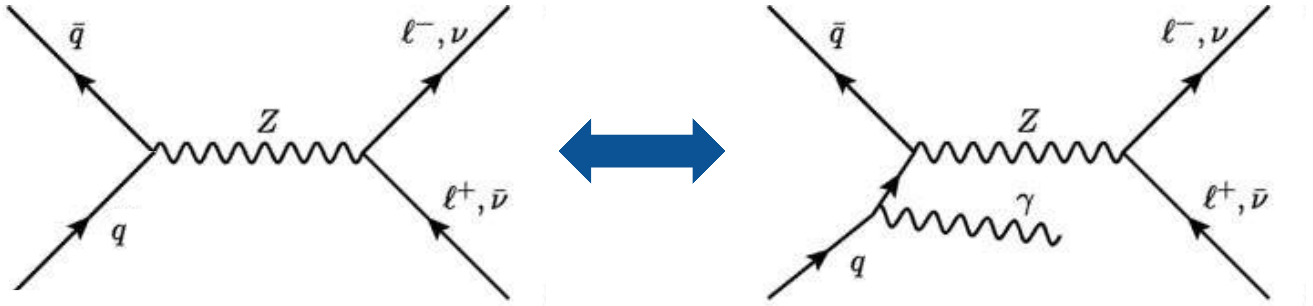
Common pitfalls I

- **Colour triplets ≠ final-state particles**
  - MC event generators do not guarantee the physicality of a "final state top"
- **Electroweak scale particles** (*W*, *Z*, *H*) **≠ final state particles**
  - Focus on leptons and hadrons for the decay channel you care about



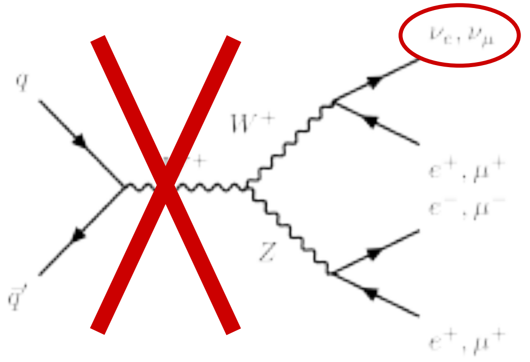
Common pitfalls II

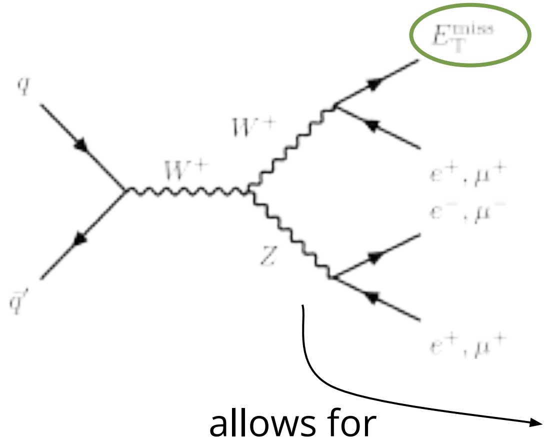
- **Hidden vetoes**
  - All important cuts should be reflected in **fiducial cross section definition**
  - ➔ E.g. veto on isolated photons in dilepton analysis may make no difference to the result when running on SM sample which is LO in the electroweak coupling
  - ➔ But what happens if more precise calculation is used which may include EW radiation?

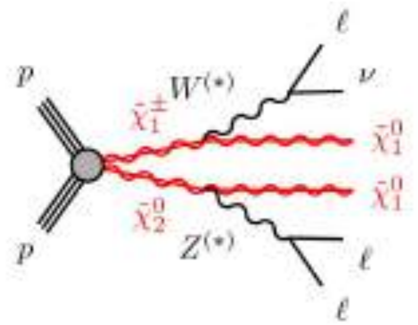


Common pitfalls III

- **Missing energy and neutrinos**
  - Explicit use of neutrino flavour and momentum very problematic, especially when more than one neutrino in event
  - Better: Use **particle-level missing transverse momentum** - correctly accounts for possible additional (BSM or other) sources of missing momentum







Rivet tutorial: Simple example in Athena

see also →**ATLAS Readme**

1. Athena setup

```
# log into your favourite system with /cvmfs access
ssh <username>@lxplus.cern.ch

# clone tutorial git repository
git clone ssh://git@gitlab.cern.ch:7999/mhabedan/rivet-tutorial.git
cd rivet-tutorial/athena

# load ATLAS environment
export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase;
source ${ATLAS_LOCAL_ROOT_BASE}/user/atlasLocalSetup.sh
asetup AthGeneration,23.6.38
setupRivet
lsetup panda # needed later for grid submission
lsetup rucio # needed later for download from grid
voms-proxy-init -voms atlas

# or
source ./setupATLAS.sh
```

Rivet tutorial: Simple example in Athena

2. Generate events

- Generate $Z \rightarrow e^+ e^-$ events with MadGraph
- Shower events with Pythia, output as LHE files
- Pass LHE files to EVGEN, output as EVNT

```
cd generate_events/
athena ./runargs.generate.py  EvgenJobTransforms/skel.GENtoEVGEN.py
```

Rivet tutorial: Simple example in Athena

3. Process events with Rivet locally

- Run Rivet locally using Athena:

```
cd ../run_rivet_locally
athena local_jO.py
```

  - Make sure to set ANALYSIS_NAME, ANALYSIS_PATH, and INPUT_FILE in local_jO.py correctly

```
cat local_jO.py
>>
from AthenaCommon.AlgSequence import AlgSequence
from Rivet_i.Rivet_iConf import Rivet_i

ANALYSIS_NAME = "MY_ANALYSIS" # or your custom analysis name
ANALYSIS_PATH = "../fallback" # or path to your own compiled analysis
INPUT_FILE = ["../fallback/EVNT.root"] # or path(s) to your own EVNT file(s)
OUTPUT_FILE = "Rivet.yoda.gz"
[...]
```

Rivet tutorial: Simple example in Athena

Digression: Using events from rucio

- Get events from rucio:

```
rucio download --nrandom 1 \
mc16_13TeV.700320.Sh_2211_Zee_maxHTpTV2_BFilter.merge.EVNT.e8351_e7400
```

- Adjust the INPUT_FILE in local_jO.py
- Pass events to Rivet as before:

```
athena local_jO.py
```

Rivet tutorial: Simple example in Athena

4. Process events with Rivet on the grid

- Submit a job running Rivet to the grid:

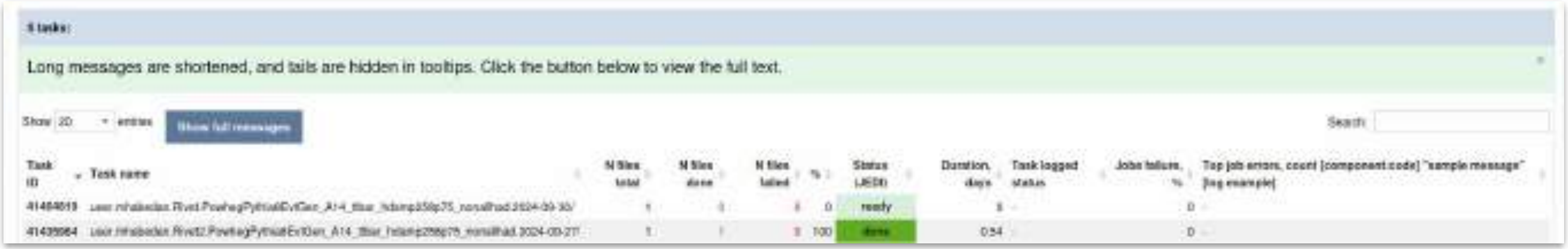
```
cd ../run_rivet_grid

CURDAT=`date "+%Y-%m-%d"`
ANALYSIS_PATH="../fallback" # or path to your own compiled analysis
INDS="mc16_13TeV.700320.Sh_2211_Zee_maxHTpTV2_BFilter.merge.EVNT.e8351_e7400"
NAME=`echo $INDS | cut -d "." -f 3`

pathena --extOutFile=Rivet.yoda.gz \
        --inDS=${INDS} --outDS=user.`whoami`.Rivet.${NAME}.${CURDAT} \
        --extFile=${ANALYSIS_PATH}/RivetAnalysis.so \
        grid_jO.py

# or
./submit.sh
```

  - Make sure to set ANALYSIS_NAME and ANALYSIS_PATH in grid_jO.py correctly

Rivet tutorial: Simple example in Athena

4. Process events with Rivet on the grid

- Inspect job progress on BigPanda: →bigpanda.cern.ch/user/<your username>:



2 tasks:

Long messages are shortened, and tails are hidden in tooltips. Click the button below to view the full text.

Show 20 entries [Show full messages] Search:

Task ID	Task name	N files total	N files done	N files failed	%	Status (JEDI)	Duration, days	Task logged status	Jobs failure, %	Top job errors, count [component:code] "sample message" [log example]
41484813	user.mhabedan.Rivet.PowhegPythia8EvtGen_A14_ttbar_hdamp258p75_nonallhad.2024-03-30/	1	0	0	0	ready	5	-	0	-
41439984	user.mhabedan.Rivet2.PowhegPythia8EvtGen_A14_ttbar_hdamp258p75_nonallhad.2024-03-27/...	1	1	0	100	done	0.54	-	0	-

Or pbook

```
pbook
>>> show()
JediTaskID      ReqID       Status   Fin%  TaskName
____________________________________________________


...
```

Rivet tutorial: Simple example in Athena

4. Process events with Rivet on the grid

- Download results

```
rucio download
user.mhabedan.Rivet3.PowhegPythia8EvtGen_A14_ttbar_hdamp258p75_nonallhad.2024-
09-30_EXT0
```

Rivet tutorial – summary

- A **physics-oriented system for physicists**
- Vision: Rivet as **standard for “truth-level” observables**, across collider phys.
- **Easy to use**, well integrated in ATLAS software
- **Plenty of support** in-/outside ATLAS available

→Athena GitLab Readme →CEDAR Mattermost Channel

→Rivet Webpage →Rivet E-Mail Contact

➔ **Add now a Rivet routine to your own analysis!**

Backup

YODA 2 - new features

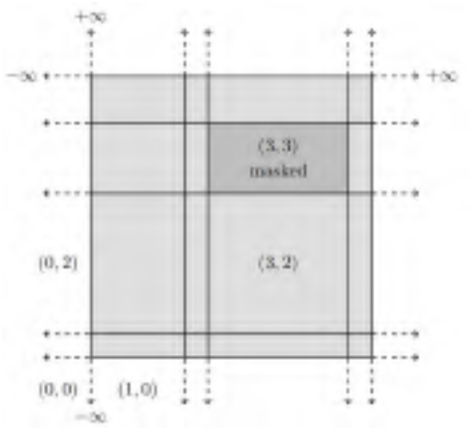
[**Excerpt**! - see→Paper]

- Bin partitioning
  - New Axis class templated on edge type
  - (*classic*) **continuous axis** triggered by std::is_floating_point trait
    - `N` bins defined by `N+1` edges, plus under- and overflow bin
    - Active uses of IEEE 754 FP standard; infinity binning:
      - bin edges: -inf   -1.0   -0.5   0.0   0.5   1.0   +inf
      - bin widths: +inf   0.5   0.5   0.5   0.5   +inf
  - (*new*) **discrete axis** for all other types
    - Bins along discrete axis only have their edge label
    - `N` bins defined by `N` edges, plus otherflow bin
    - Useful for multiplicities, cutflows, …

- Reduced internal code duplication to support C++ and Python API

- New BinnedStorage class can hold arbitrary bin-content types
  - Supports index-based bin(i) and coordinate-based binAt(x) lookups
  - Supports bin masking (mask(i), maskAt(x)) to emulate “gaps" (in place of bin erasure)

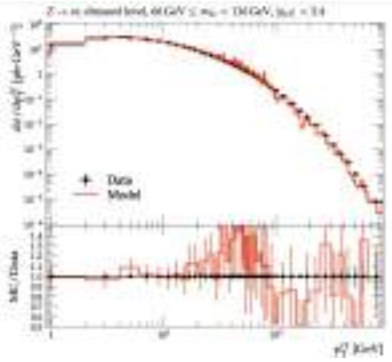
- matplotlib-based plotting machinery produces self-consistent Python scripts for better customisation of plots (no YODA installation required)



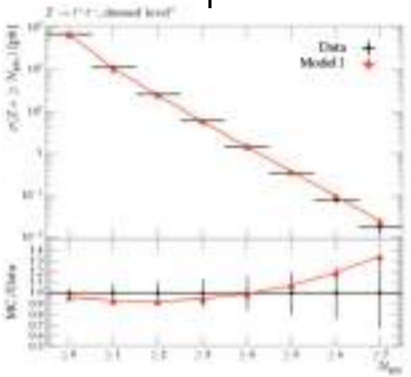
YODA 2.0.1 - new features

[**Excerpt**! - see→MR]

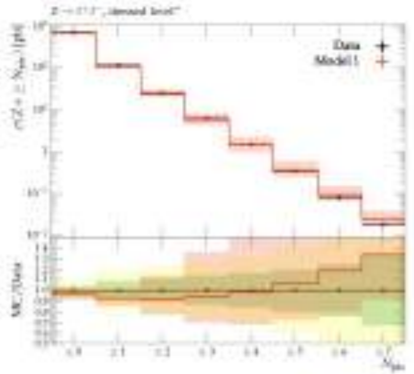
Inhomogeneous binning



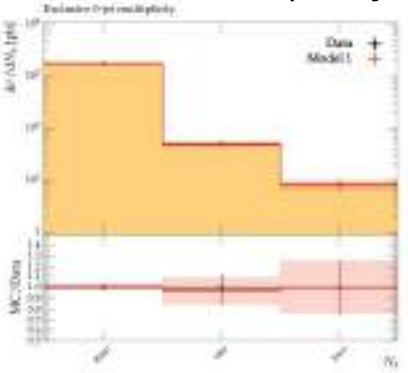
Marker improvements



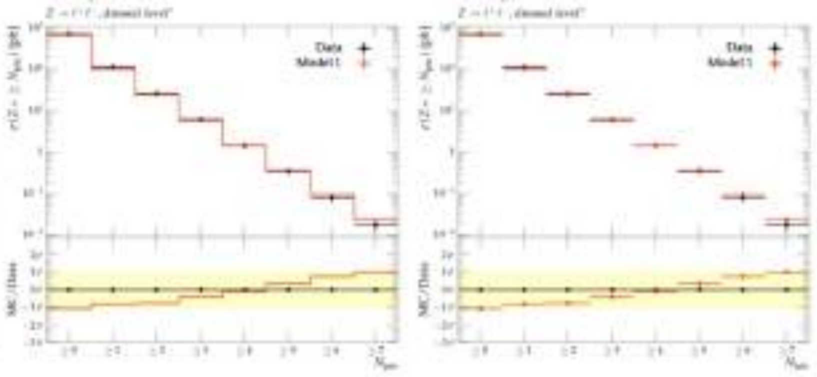
Multiple sigma-level bands



Fill colour and opacity



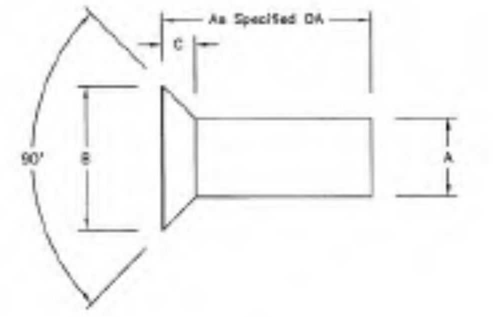
Histogram vs. scatter style



Automatic legend positioning

Rivet design philoshophy

- **Ease of use**
  - Big emphasis on “more physics, less noise”!
  - Minimal boilerplate analysis code, HEPData sync
  - Event loop and histogramming basically familiar
  - Tools to avoid having to touch the raw event graph
- **Embeddable**
  - OO C++ library, Python wrapper, sane user scripts
  - Generator independence: communication via HepMC
  - Analysis routines factorised: loaded as “plugins”
- **Efficient**
  - Avoid recomputations via “projection” caching
- **Physical**
  - Measurements primarily from final-state particles only



Rivet 4 - Projection streamlining

- Clean-up of projection arguments in favour of self-documenting scoped enum classes:
  - FastJets::Algo::KT → JetAlg::KT
  - JetAlg::Muons::NONE → JetMuons::NONE
  - JetAlg::Invisibles::DECAY → JetInvisibles::DECAY
- Boolean arguments for treatment of tau/muon decay products replaced with TauDecaysAs and MuDecaysAs enum classes
- DressedLeptons renamed LeptonFinder, akin to existing JetFinder and ParticleFinder
  - Old DressedLeptons now alias for vector<DressedLepton>
- Similarly, ZFinder renamed DileptonFinder
- WFinder removed entirely!
  - Significantly improves self-documentation of analysis code, clarifying previously obscure model-dependent assumptions woven into the measurement data
  - new closestMatchIndex() metafunction to help identify *W* candidates, e.g. `const int bestmatch = closestMatchIndex(leptons, pmiss, Kin::mass, 80.4*GeV);`

Rivet 4 - More API updates

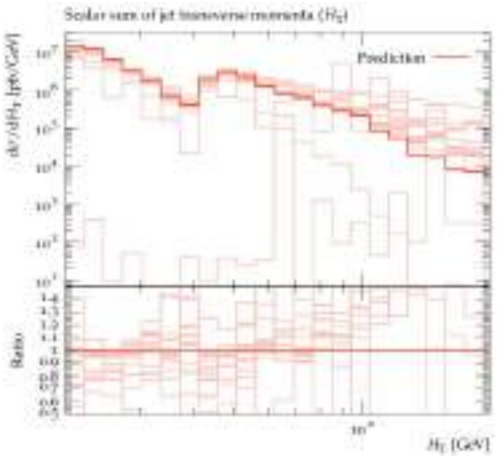
- Smearing of NLO sub-events now generalised to arbitrary dimensions and axis types
- `BinnedHistogram` class was standalone, didn't quite fit, awkward syntax

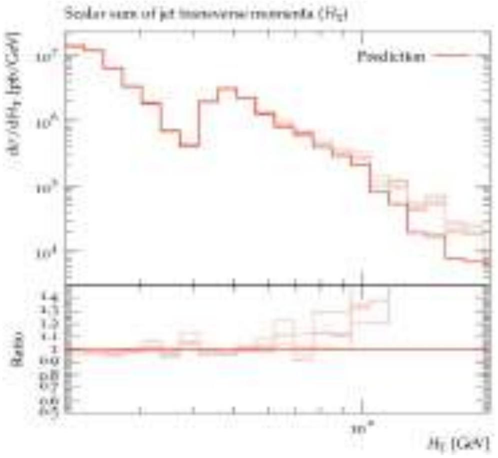
➔ Replaced with respective YODA class `HistoGroup`

- New interface to `HDF5` and `HighFive` for storing and loading analysis-specific auxiliary data
- New (optional) interface to ONNX Runtime as (current) best option for ML preservation
- Better support for massively parallel applications
- Reduced I/O load from parsing info files in the initialisation phase

Multiweights and re-entry

- **MC weight vectors** allow expression of **increasingly complex theory uncertainties**
  - But burden for analysis chains: have to propagate and correctly combine *O*(200) weight streams!
- Rivet 3: **complex automatic handling of weights**
  - ~Invisible to users: data objects *look* like histograms etc. but are secretly multiplexed
- Can now **re-call finalisation to combine runs**
  - RAW histogram stage preserves pre-finalize objects
  - ➔ "re-entrant" perfect data-object merging
  - Key for e.g. *pA/pp or W/Z* ratios, + BSM recasting
- **Data types are important:**
  - Glimpses of fully coherent separation of semantics from presentation





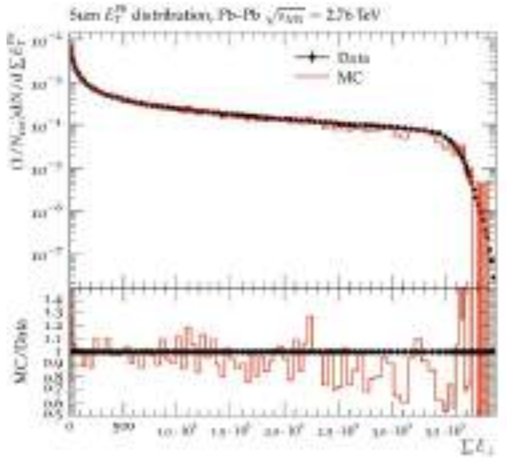
Heavy ions in Rivet

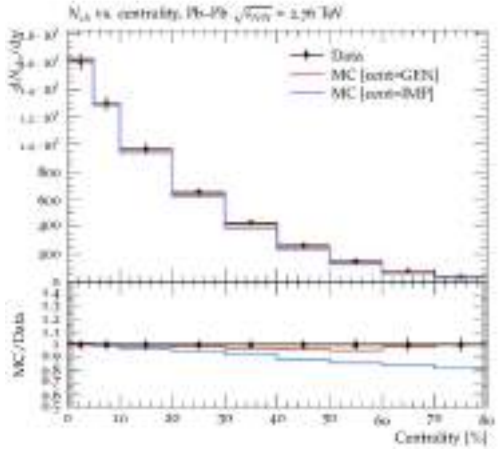
- “Adding heavy-ion support” sounds trivial!
- Actually: **stern test**, with far-reaching impacts
  - HI observables often require **centrality calibration** curves

  ➔ Need 2-pass run

  ➔ Wasn't planned
  - And event/event correlations… centrality-binned!

  ➔ Need **swappable definitions**:
  few HI generators general-purpose enough to do e.g. both forward E_T *and* jet quenching
- →Paper
- HI MC standards also in flux: having a common tool enables discussion on common standards





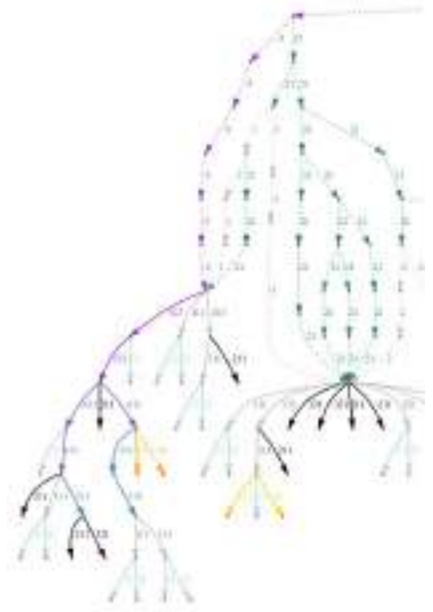
Physically safe analysis methods

- **Usage of partons, bosons**, etc. directly from the event graph: **Unphysical**, depend on approximations, may not even exist!

Consequences:

- Refine the "fiducial" idea, define **unfolding targets**
  - Today: "Rivet = level that is unfolded to"
- Hadronisation as a "**decoherence barrier**
  - Use natural dividing line between quantum-interfering hard process & semi-classical decays: ~ no tempting partons!
- Bringing **truth tagging** closer to rec
  - First used *b*-ancestry of jet constituents to set HF labels: Too inclusive!

  ➔ *Associate* the hard-fragmenting, weakly-decaying *B*



Physically safe analysis methods

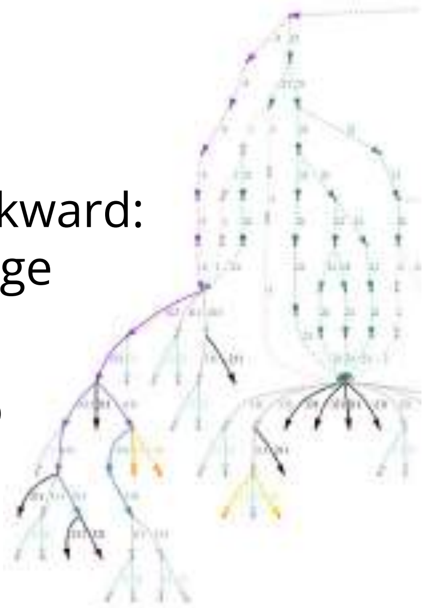
- **Usage of partons, bosons**, etc. directly from the event graph: **Unphysical**, depend on approximations, may not even exist!

Consequences:

- Unfolding targets
- Decoherence barrier
- Truth tagging

- **Promptness**/directness tests
  - Don't identify particle "from the hard process"; do it backward: Label as *indirect* via recursive checks for hadron parentage

- **Dressed leptons**
  - Now primarily *dress* truth leptons with their photon halo



The future of Rivet

- Vision: **Rivet** as **standard for “truth-level” observables**, across collider phys.
- Not just standalone, but as a library in pheno & experiment frameworks, too: standard MC definitions (cf. CMS), seamless systematics handling, etc.
- Core: a **physics-oriented system for physicists** to compare MC predictions to one another and to data, on many simultaneous observables, in myriad ways We don’t know all the use-cases yet!
- Challenges:
  - Extension of HEPData and other community infrastructure for ever more precise data: Even our compressed data format struggle with volume of analyses and data

  ➔ Work needed on multiweight-oriented data format and tools