RIVET

Kathryn Coldham, Brunel University London

01/03/2019

RIVET (Robust Independent Validation of Experiment and Theory)

- RIVET is a toolkit used to test MC event generators and to also compare their outputs to theoretical predictions.

- Theorists can use the plug-ins to have access to, for example, the cuts made in analyses so they can test theories.

- All groups in CMS must now have a Rivet analysis. The list of analyses required by CMS is here.

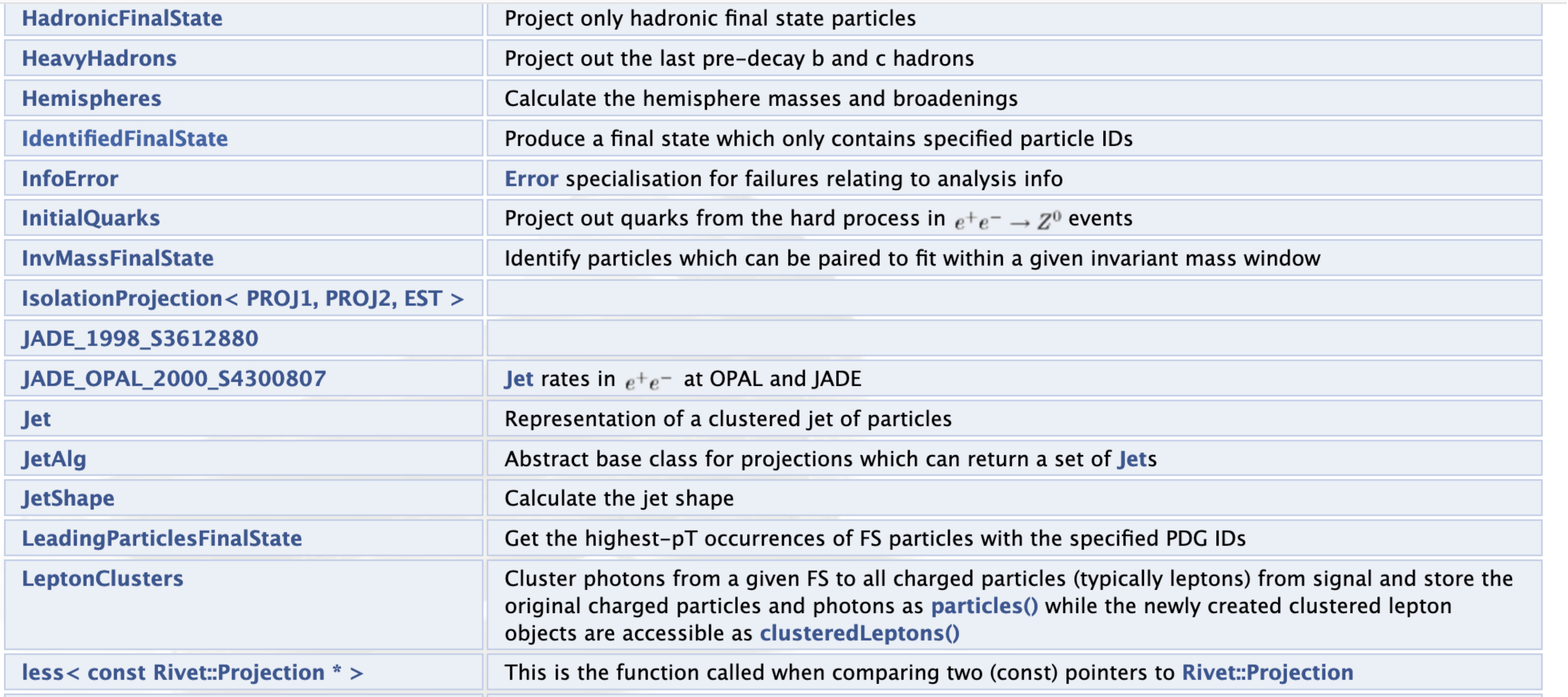
How to install rivet

Type the following commands into your lxplus terminal:

```
source /cvmfs/cms.cern.ch/cmsset_default.sh
cmsrel CMSSW_10_0_0
cd CMSSW_10_0_0/src/
cmsenv
git-cms-init
git-cms-addpkg GeneratorInterface/RivetInterface
git-cms-addpkg Configuration/Generator
git clone https://CERNUSERNAME@gitlab.cern.ch:8443/cms-gen/Rivet.git
source Rivet/rivetSetup.sh
scram b -j8
```

RIVET Classes

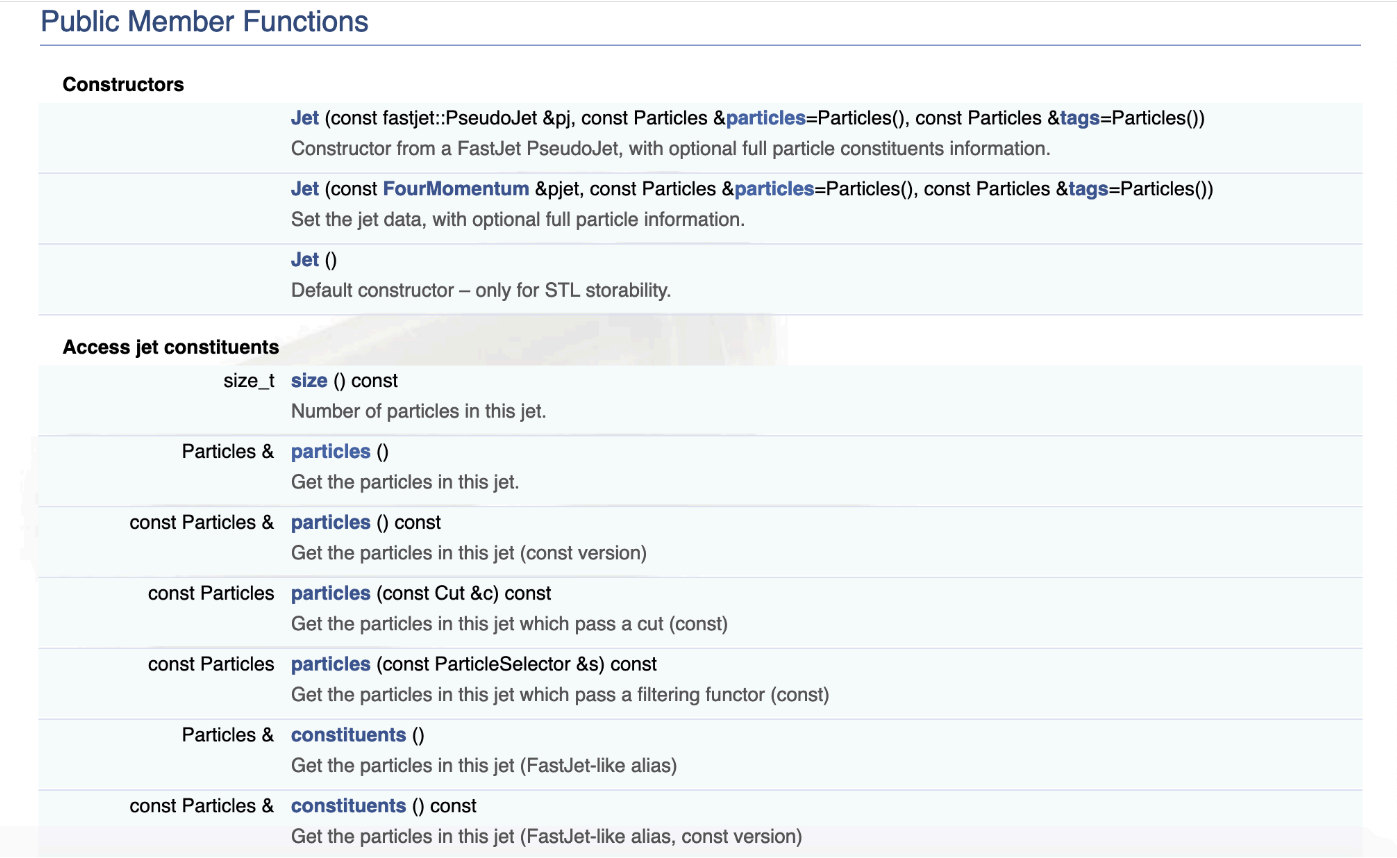
The complete list of classes is available on the RIVET website



HadronicFinalState	Project only hadronic final state particles
HeavyHadrons	Project out the last pre-decay b and c hadrons
Hemispheres	Calculate the hemisphere masses and broadenings
IdentifiedFinalState	Produce a final state which only contains specified particle IDs
InfoError	Error specialisation for failures relating to analysis info
InitialQuarks	Project out quarks from the hard process in $e^+e^- \to Z^0$ events
InvMassFinalState	Identify particles which can be paired to fit within a given invariant mass window
IsolationProjection< PROJ1, PROJ2, EST >	
JADE_1998_S3612880	
JADE_OPAL_2000_S4300807	Jet rates in e^+e^- at OPAL and JADE
Jet	Representation of a clustered jet of particles
JetAlg	Abstract base class for projections which can return a set of Jets
JetShape	Calculate the jet shape
LeadingParticlesFinalState	Get the highest-pT occurrences of FS particles with the specified PDG IDs
LeptonClusters	Cluster photons from a given FS to all charged particles (typically leptons) from signal and store the original charged particles and photons as particles() while the newly created clustered lepton objects are accessible as clusteredLeptons()
less< const Rivet::Projection * >	This is the function called when comparing two (const) pointers to Rivet::Projection

RIVET Functions

- Each RIVET class has a list of public member functions.
- E.g. The list of functions for the jets final state class can be found here.



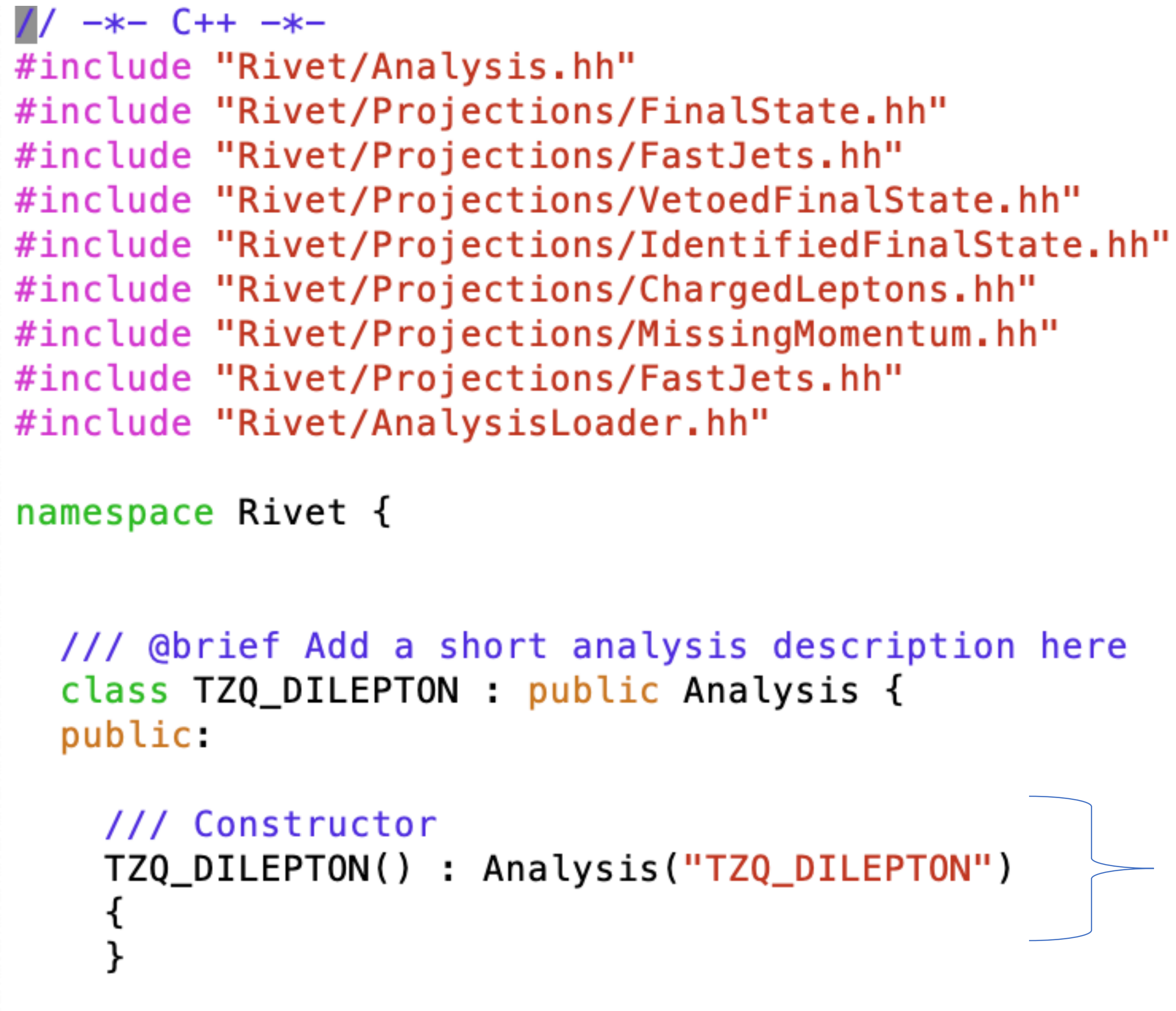
Public Member Functions

Constructors

	Jet (const fastjet::PseudoJet &pj, const Particles &**particles**=Particles(), const Particles &**tags**=Particles()) Constructor from a FastJet PseudoJet, with optional full particle constituents information.
	Jet (const **FourMomentum** &pjet, const Particles &**particles**=Particles(), const Particles &**tags**=Particles()) Set the jet data, with optional full particle information.
	Jet () Default constructor – only for STL storability.

Access jet constituents

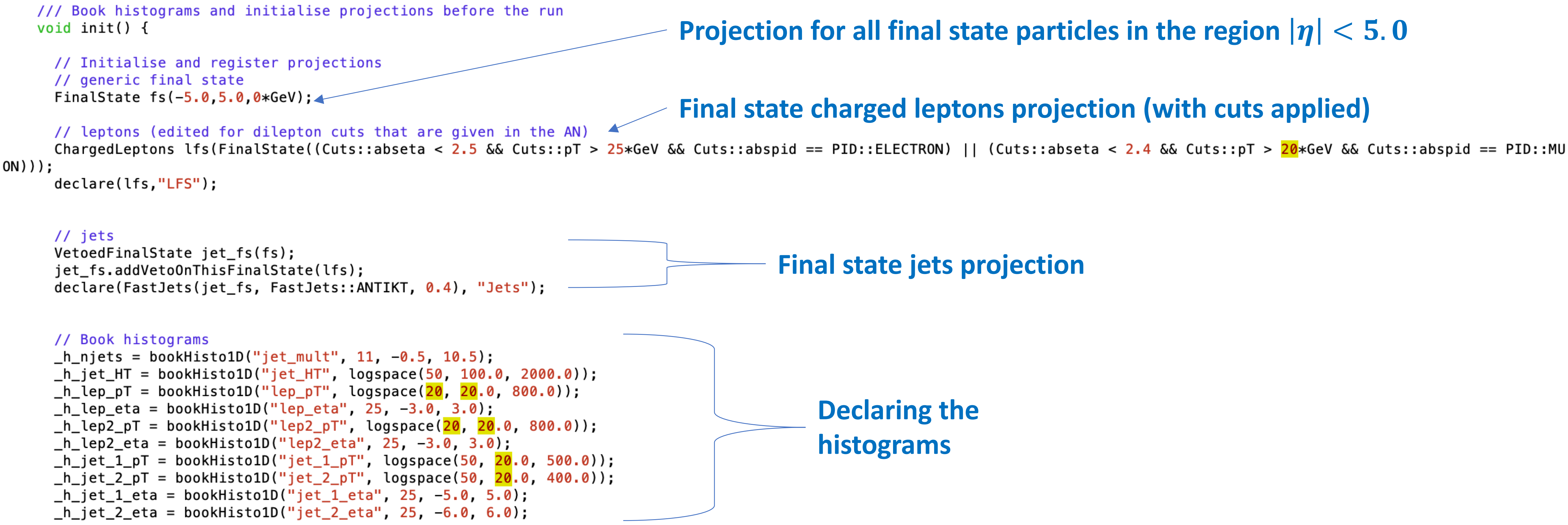
size_t	**size** () const Number of particles in this jet.
Particles &	**particles** () Get the particles in this jet.
const Particles &	**particles** () const Get the particles in this jet (const version)
const Particles	**particles** (const Cut &c) const Get the particles in this jet which pass a cut (const)
const Particles	**particles** (const ParticleSelector &s) const Get the particles in this jet which pass a filtering functor (const)
Particles &	**constituents** () Get the particles in this jet (FastJet-like alias)
const Particles &	**constituents** () const Get the particles in this jet (FastJet-like alias, const version)



```
// -*- C++ -*-
#include "Rivet/Analysis.hh"
#include "Rivet/Projections/FinalState.hh"
#include "Rivet/Projections/FastJets.hh"
#include "Rivet/Projections/VetoedFinalState.hh"
#include "Rivet/Projections/IdentifiedFinalState.hh"
#include "Rivet/Projections/ChargedLeptons.hh"
#include "Rivet/Projections/MissingMomentum.hh"
#include "Rivet/Projections/FastJets.hh"
#include "Rivet/AnalysisLoader.hh"

namespace Rivet {


  /// @brief Add a short analysis description here
  class TZQ_DILEPTON : public Analysis {
  public:

    /// Constructor
    TZQ_DILEPTON() : Analysis("TZQ_DILEPTON")
    {
    }
```

All analyses must use the Rivet namespace

Constructor for the analysis



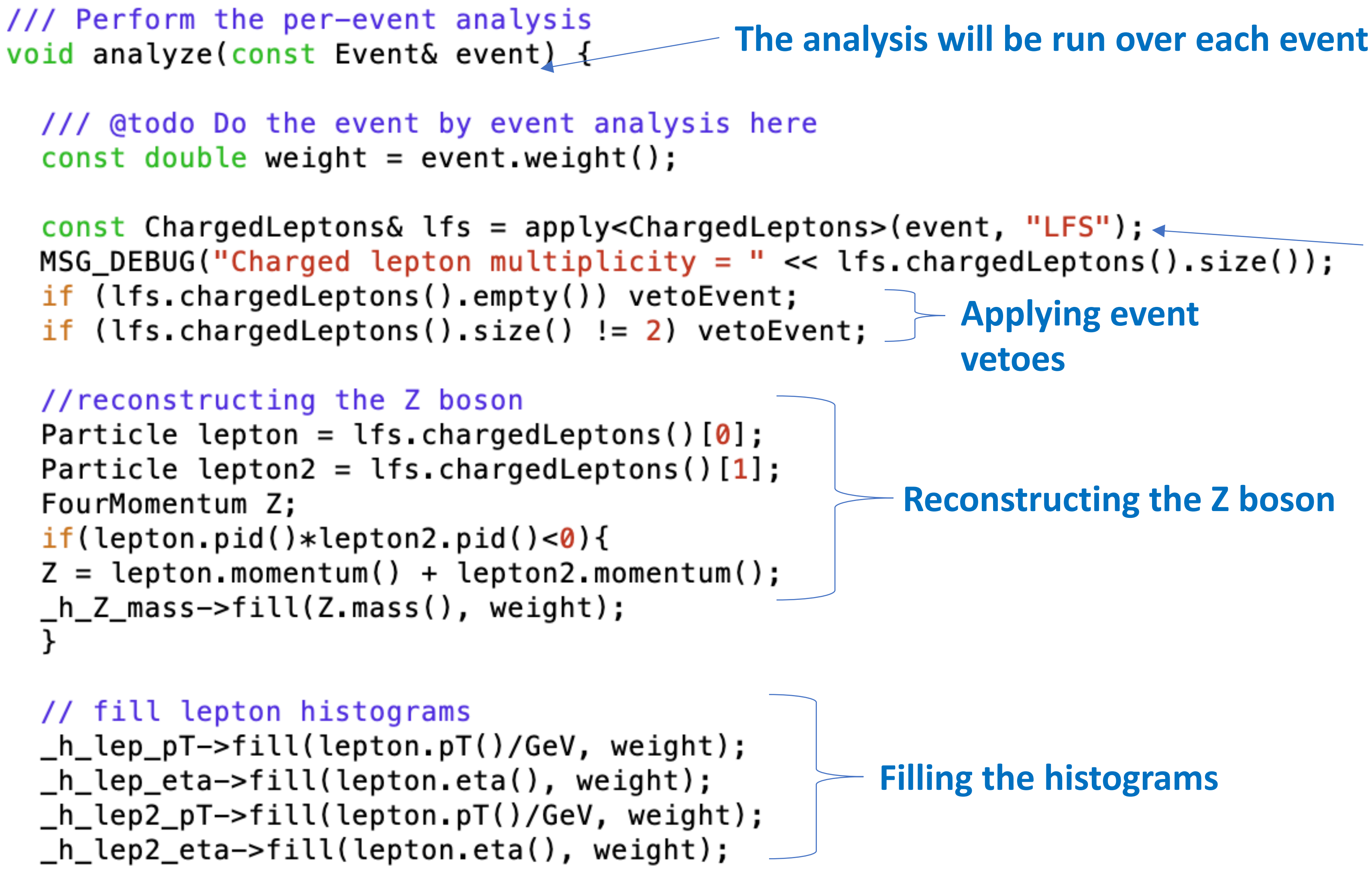
```
/// Book histograms and initialise projections before the run
void init() {

  // Initialise and register projections
  // generic final state
  FinalState fs(-5.0,5.0,0*GeV);

  // leptons (edited for dilepton cuts that are given in the AN)
  ChargedLeptons lfs(FinalState((Cuts::abseta < 2.5 && Cuts::pT > 25*GeV && Cuts::abspid == PID::ELECTRON) || (Cuts::abseta < 2.4 && Cuts::pT > 20*GeV && Cuts::abspid == PID::MU
ON)));
  declare(lfs,"LFS");


  // jets
  VetoedFinalState jet_fs(fs);
  jet_fs.addVetoOnThisFinalState(lfs);
  declare(FastJets(jet_fs, FastJets::ANTIKT, 0.4), "Jets");


  // Book histograms
  _h_njets = bookHisto1D("jet_mult", 11, -0.5, 10.5);
  _h_jet_HT = bookHisto1D("jet_HT", logspace(50, 100.0, 2000.0));
  _h_lep_pT = bookHisto1D("lep_pT", logspace(20, 20.0, 800.0));
  _h_lep_eta = bookHisto1D("lep_eta", 25, -3.0, 3.0);
  _h_lep2_pT = bookHisto1D("lep2_pT", logspace(20, 20.0, 800.0));
  _h_lep2_eta = bookHisto1D("lep2_eta", 25, -3.0, 3.0);
  _h_jet_1_pT = bookHisto1D("jet_1_pT", logspace(50, 20.0, 500.0));
  _h_jet_2_pT = bookHisto1D("jet_2_pT", logspace(50, 20.0, 400.0));
  _h_jet_1_eta = bookHisto1D("jet_1_eta", 25, -5.0, 5.0);
  _h_jet_2_eta = bookHisto1D("jet_2_eta", 25, -6.0, 6.0);
```

Projection for all final state particles in the region $|\eta| < 5.0$

Final state charged leptons projection (with cuts applied)

Final state jets projection

Declaring the histograms



```
/// Perform the per-event analysis
void analyze(const Event& event) {

  /// @todo Do the event by event analysis here
  const double weight = event.weight();

  const ChargedLeptons& lfs = apply<ChargedLeptons>(event, "LFS");
  MSG_DEBUG("Charged lepton multiplicity = " << lfs.chargedLeptons().size());
  if (lfs.chargedLeptons().empty()) vetoEvent;
  if (lfs.chargedLeptons().size() != 2) vetoEvent;

  //reconstructing the Z boson
  Particle lepton = lfs.chargedLeptons()[0];
  Particle lepton2 = lfs.chargedLeptons()[1];
  FourMomentum Z;
  if(lepton.pid()*lepton2.pid()<0){
  Z = lepton.momentum() + lepton2.momentum();
  _h_Z_mass->fill(Z.mass(), weight);
  }

  // fill lepton histograms
  _h_lep_pT->fill(lepton.pT()/GeV, weight);
  _h_lep_eta->fill(lepton.eta(), weight);
  _h_lep2_pT->fill(lepton.pT()/GeV, weight);
  _h_lep2_eta->fill(lepton.eta(), weight);
```

The analysis will be run over each event

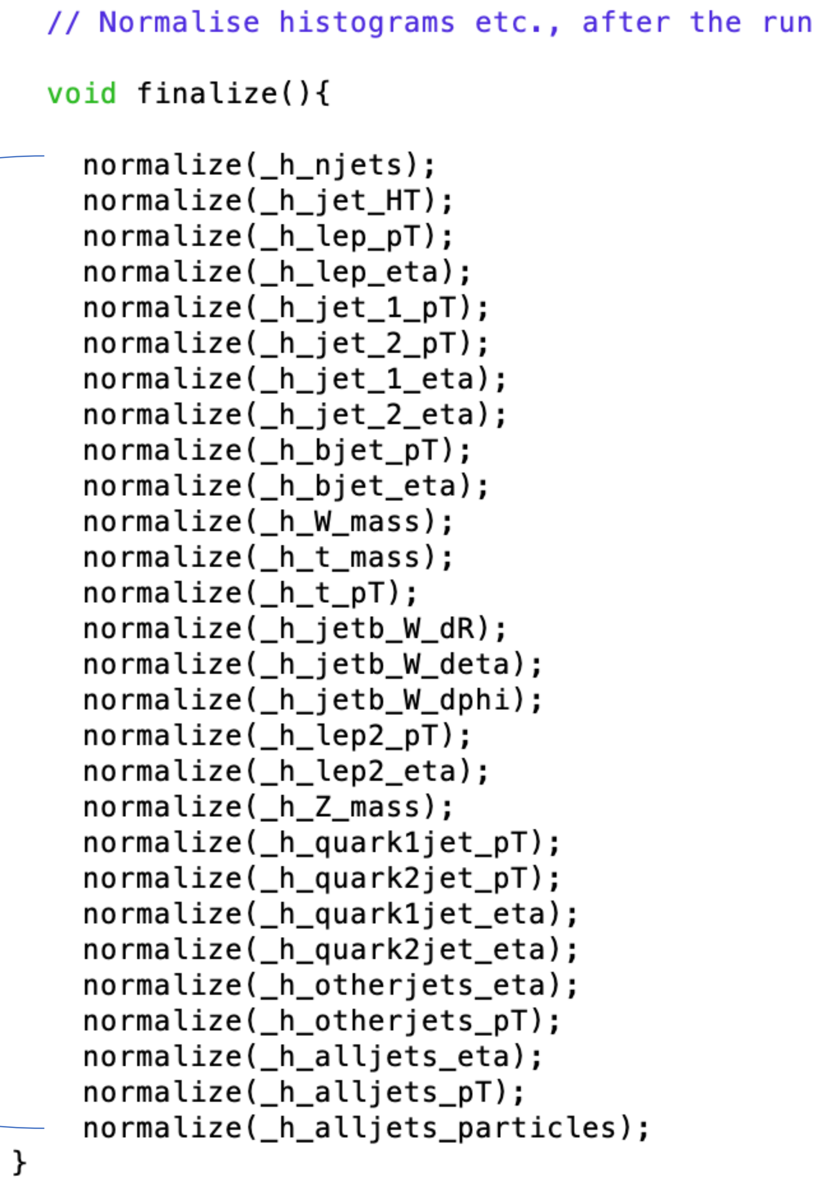
Initialising the ChargedLeptons projection

Applying event vetoes

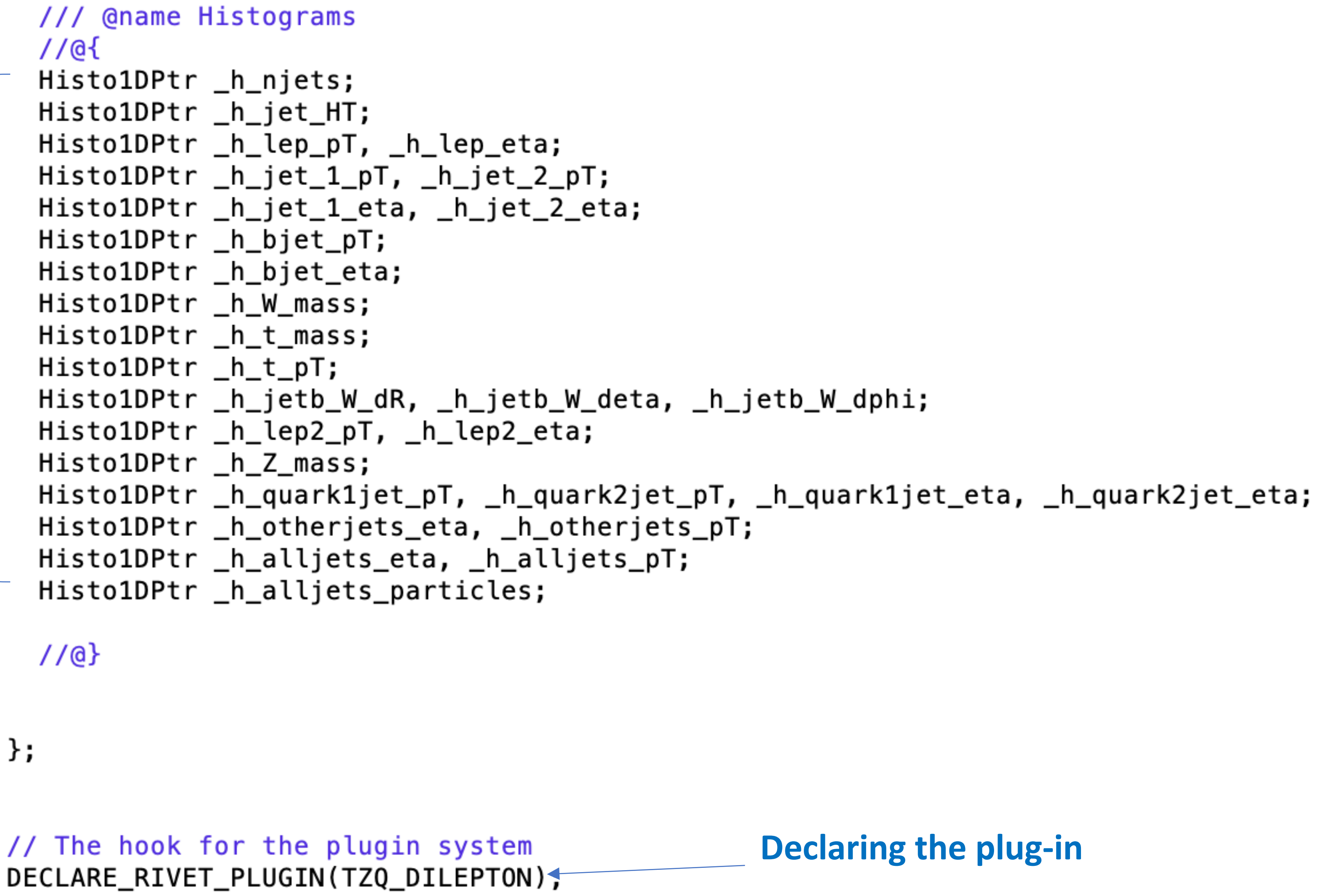
Reconstructing the Z boson

Filling the histograms

Normalizing the histograms



```
 // Normalise histograms etc., after the run

 void finalize(){

  normalize(_h_njets);
  normalize(_h_jet_HT);
  normalize(_h_lep_pT);
  normalize(_h_lep_eta);
  normalize(_h_jet_1_pT);
  normalize(_h_jet_2_pT);
  normalize(_h_jet_1_eta);
  normalize(_h_jet_2_eta);
  normalize(_h_bjet_pT);
  normalize(_h_bjet_eta);
  normalize(_h_W_mass);
  normalize(_h_t_mass);
  normalize(_h_t_pT);
  normalize(_h_jetb_W_dR);
  normalize(_h_jetb_W_deta);
  normalize(_h_jetb_W_dphi);
  normalize(_h_lep2_pT);
  normalize(_h_lep2_eta);
  normalize(_h_Z_mass);
  normalize(_h_quark1jet_pT);
  normalize(_h_quark2jet_pT);
  normalize(_h_quark1jet_eta);
  normalize(_h_quark2jet_eta);
  normalize(_h_otherjets_eta);
  normalize(_h_otherjets_pT);
  normalize(_h_alljets_eta);
  normalize(_h_alljets_pT);
  normalize(_h_alljets_particles);
}
```



```
 /// @name Histograms
 //@{
 Histo1DPtr _h_njets;
 Histo1DPtr _h_jet_HT;
 Histo1DPtr _h_lep_pT, _h_lep_eta;
 Histo1DPtr _h_jet_1_pT, _h_jet_2_pT;
 Histo1DPtr _h_jet_1_eta, _h_jet_2_eta;
 Histo1DPtr _h_bjet_pT;
 Histo1DPtr _h_bjet_eta;
 Histo1DPtr _h_W_mass;
 Histo1DPtr _h_t_mass;
 Histo1DPtr _h_t_pT;
 Histo1DPtr _h_jetb_W_dR, _h_jetb_W_deta, _h_jetb_W_dphi;
 Histo1DPtr _h_lep2_pT, _h_lep2_eta;
 Histo1DPtr _h_Z_mass;
 Histo1DPtr _h_quark1jet_pT, _h_quark2jet_pT, _h_quark1jet_eta, _h_quark2jet_eta;
 Histo1DPtr _h_otherjets_eta, _h_otherjets_pT;
 Histo1DPtr _h_alljets_eta, _h_alljets_pT;
 Histo1DPtr _h_alljets_particles;

 //@}


};


// The hook for the plugin system
DECLARE_RIVET_PLUGIN(TZQ_DILEPTON);
```

Naming the histograms

Declaring the plug-in

The scripts must be placed inside specific directories!

C++ script must be in the directory
/CMSSW_10_0_0/src/GeneratorInterface/RivetInterface/src

Python and crab configuration files must be in the directory
/CMSSW_10_0_0/src/GeneratorInterface/RivetInterface/test

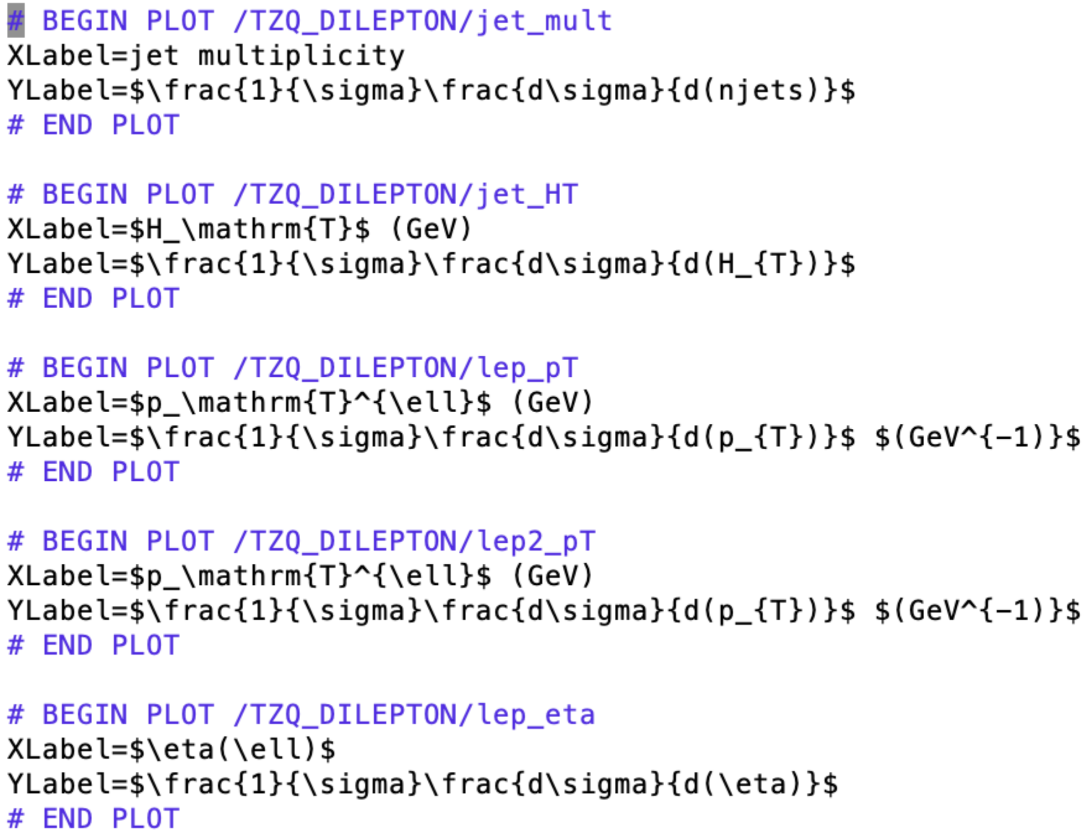
A python configuration file is used to generate events

- An example python config script can be found here.
- The modified scripts for 2016 and 2017/2018 are here and here, respectively.
- Line 8: set maximum number of events
- Line 46: Change to the name of your analysis
- Line 50: Cross section (in pb)
- Line 55: List of root files from the CMS Data Aggregation Service (will need a Grid certificate)

Adding axes titles

- Create a .plot script in the directory: /CMSSW_10_0_0/src/GeneratorInterface/RivetInterface/data
  - Script for 2016
  - Script for 2017 and 2018

- Compile using scram b and then use cmsRun on the python config file



```
# BEGIN PLOT /TZQ_DILEPTON/jet_mult
XLabel=jet multiplicity
YLabel=$\frac{1}{\sigma}\frac{d\sigma}{d(njets)}$
# END PLOT

# BEGIN PLOT /TZQ_DILEPTON/jet_HT
XLabel=$H_\mathrm{T}$ (GeV)
YLabel=$\frac{1}{\sigma}\frac{d\sigma}{d(H_{T})}$
# END PLOT

# BEGIN PLOT /TZQ_DILEPTON/lep_pT
XLabel=$p_\mathrm{T}^{\ell}$ (GeV)
YLabel=$\frac{1}{\sigma}\frac{d\sigma}{d(p_{T})}$ $(GeV^{-1})$
# END PLOT

# BEGIN PLOT /TZQ_DILEPTON/lep2_pT
XLabel=$p_\mathrm{T}^{\ell}$ (GeV)
YLabel=$\frac{1}{\sigma}\frac{d\sigma}{d(p_{T})}$ $(GeV^{-1})$
# END PLOT

# BEGIN PLOT /TZQ_DILEPTON/lep_eta
XLabel=$\eta(\ell)$
YLabel=$\frac{1}{\sigma}\frac{d\sigma}{d(\eta)}$
# END PLOT
```

To compile and run locally

- Use *scram b –j8* to compile scripts inside your working directory.

- While inside the directory that contains the python config file, use *cmsRun file_name* to run a rivet script called file_name.

- Afterwards, use the *rivet-mkhtml output_file_name.yoda* command to generate the plots from an output file called output_file_name.yoda.

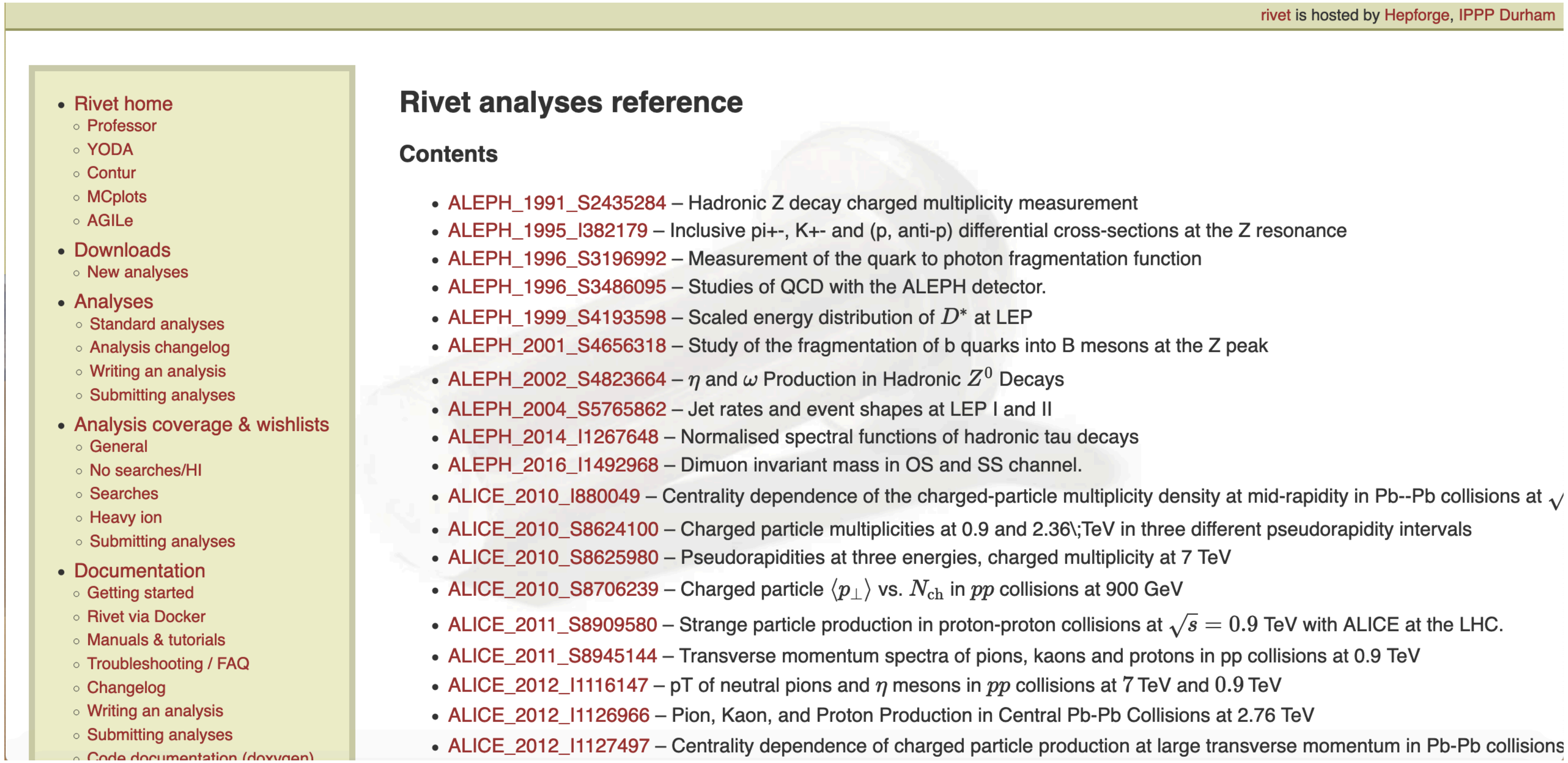
To run using crab

- Place python config file inside the GeneratorInterface/RivetInterface/test directory

- CRAB config files for:
  - 2016
  - 2017 and 2018 (edit lines 4, 5, 10, 11, 15 and 20 to change years)

- Add the following to your bashrc file: *alias crabsetup='source /cvmfs/cms.cern.ch/crab3/crab.sh'*

Validated RIVET Analyses can be found on this webpage



rivet is hosted by Hepforge, IPPP Durham

- Rivet home
  - Professor
  - YODA
  - Contur
  - MCplots
  - AGILe
- Downloads
  - New analyses
- Analyses
  - Standard analyses
  - Analysis changelog
  - Writing an analysis
  - Submitting analyses
- Analysis coverage & wishlists
  - General
  - No searches/HI
  - Searches
  - Heavy ion
  - Submitting analyses
- Documentation
  - Getting started
  - Rivet via Docker
  - Manuals & tutorials
  - Troubleshooting / FAQ
  - Changelog
  - Writing an analysis
  - Submitting analyses
  - Code documentation (doxygen)

Rivet analyses reference

Contents

- ALEPH_1991_S2435284 – Hadronic Z decay charged multiplicity measurement
- ALEPH_1995_I382179 – Inclusive pi+-, K+- and (p, anti-p) differential cross-sections at the Z resonance
- ALEPH_1996_S3196992 – Measurement of the quark to photon fragmentation function
- ALEPH_1996_S3486095 – Studies of QCD with the ALEPH detector.
- ALEPH_1999_S4193598 – Scaled energy distribution of D^* at LEP
- ALEPH_2001_S4656318 – Study of the fragmentation of b quarks into B mesons at the Z peak
- ALEPH_2002_S4823664 – η and ω Production in Hadronic Z^0 Decays
- ALEPH_2004_S5765862 – Jet rates and event shapes at LEP I and II
- ALEPH_2014_I1267648 – Normalised spectral functions of hadronic tau decays
- ALEPH_2016_I1492968 – Dimuon invariant mass in OS and SS channel.
- ALICE_2010_I880049 – Centrality dependence of the charged-particle multiplicity density at mid-rapidity in Pb--Pb collisions at $\sqrt{}$
- ALICE_2010_S8624100 – Charged particle multiplicities at 0.9 and 2.36\;TeV in three different pseudorapidity intervals
- ALICE_2010_S8625980 – Pseudorapidities at three energies, charged multiplicity at 7 TeV
- ALICE_2010_S8706239 – Charged particle $\langle p_\perp \rangle$ vs. N_ch in pp collisions at 900 GeV
- ALICE_2011_S8909580 – Strange particle production in proton-proton collisions at $\sqrt{s} = 0.9$ TeV with ALICE at the LHC.
- ALICE_2011_S8945144 – Transverse momentum spectra of pions, kaons and protons in pp collisions at 0.9 TeV
- ALICE_2012_I1116147 – pT of neutral pions and η mesons in pp collisions at 7 TeV and 0.9 TeV
- ALICE_2012_I1126966 – Pion, Kaon, and Proton Production in Central Pb-Pb Collisions at 2.76 TeV
- ALICE_2012_I1127497 – Centrality dependence of charged particle production at large transverse momentum in Pb-Pb collisions

RIVET plug-in: TZQ_DILEPTON

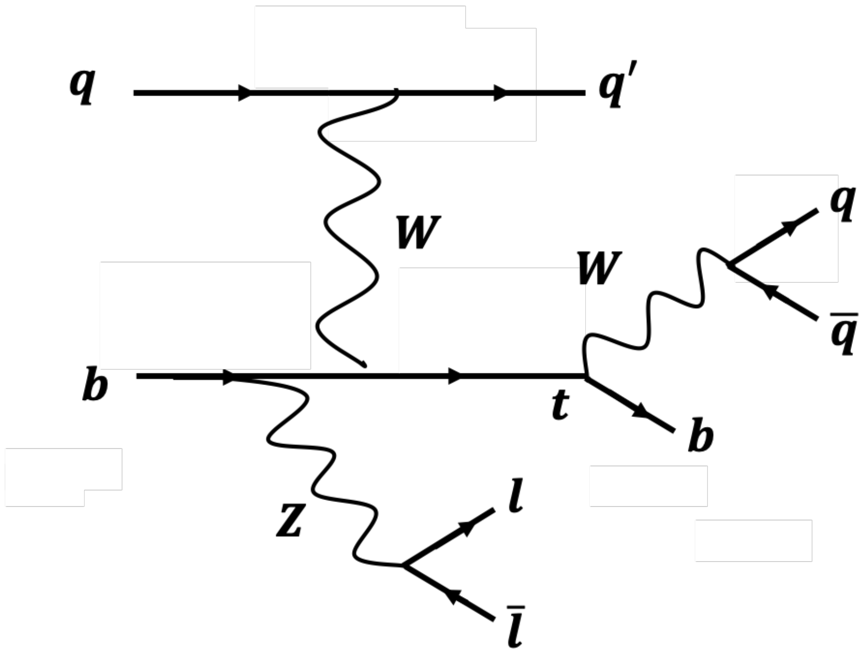
- Final state electrons with $p_T > 25$ GeV and $|\eta| < 2.4$ or muons with $p_T > 20$ GeV and $|\eta| < 2.5$

- Anti-K$_T$ jets with **R = 0.4**.
- Jets: $|\eta| < 4.7$, b-tagged jets: $|\eta| < 2.4$
- For both b-tagged and non b-tagged jets: $p_T > 30$ GeV

- Quark jets are selected with a reconstructed mass within **20 GeV** of the W boson mass.

- The Z boson, top quark and W boson masses are reconstructed.

- So far, 2 million events from Summer 2016-Moroid 2017 Nominal tZq signal samples have been used.



Summary of terminal commands (in lxplus) (1)

```
cd private
cmsrel CMSSW_10_0
cd CMSSW_10_0_0/src/
cmsenv
cmsproxy
cmsinit
rivetsetup
crabsetup
cd GeneratorInterface/RivetInterface/src
```

The RIVET C++ script should be saved in here

<u>Before using these, add the following four lines to your ~/.bashrc file:</u>

```
alias cmsinit='. /cvmfs/cms.cern.ch/cmsset_default.sh'
alias crabsetup='source /cvmfs/cms.cern.ch/crab3/crab.sh'
alias rivetsetup='source Rivet/rivetSetup.sh'
alias cmsproxy='voms-proxy-init -voms cms'
```

Summary of terminal commands (2)

- Once you edit the C++ script:

```
scram b
cd ..
cd test
```

← The RIVET python config script should be saved in here

Once you edit the python script, to run locally use:

```
scram b
cmsRun filename.py
```

Summary of terminal commands (3)

- To run using crab: *crab submit NameOfCrabConfigFile.py*

- To check the status of the jobs:
  - *crab status -d '*crab_projects_dilepton/dilepton_analysis_2017*'*

- To retrieve completed crab jobs:
  - *crab getoutput --quantity="all"*

Summary of terminal commands (4)

- Enter the results directory:
  - *cd test/dilepton/crab_dilepton_analysis_2017/results*

- Use the yodaNormalize.py script (which should be saved in the test directory) on each yoda output file:
  - E.g. *python yodaNormalize.py TZQ_2017_1.yoda TZQ_2017_1_NORM.yoda*

- *Merge the normalized output files:*
  - yodamerge -o TZQ_2017_COMBINED.yoda TZQ_2017_*_NORM.yoda

- Obtain the output distributions:
  - rivet-mkhtml TZQ_2017_COMBINED.yoda

Summary of terminal commands (5)

- Combining the distributions for different years:
  - *rivet-mkhtml TZQ_2016 _COMBINED.yoda TZQ_2017 _COMBINED.yoda TZQ_2018 _COMBINED.yoda*

Thank you – any questions?

Useful links and contacts

- RIVET website
- CMS RIVET Twiki page
- RIVET user manual
- Tutorials
- Gitlab for TOP analyses
- RIVET lectures on the CERN Document Server

- RIVET developers’ email: rivet@projects.hepforge.org
- CMS TOP RIVET contact: otto.heinz.hinrichs@cern.ch