Contents

1 Installing Standard Programs:

1.1 Installing HepMC

```
wget -c 'http://lcgapp.cern.ch/project/simu/HepMC/download/HepMC-2.06.09.tar.gz'
tar -xf 'HepMC-2.06.09.tar.gz'
cd 'HepMC-2.06.09/'
'./configure' '--with-momentum=GEV' '--with-length=MM'
make -j4
sudo make install
```

1.2 Installing FastJet:

```
wget -c 'http://www.fastjet.fr/repo/fastjet-3.3.0.tar.gz'
tar -xf './fastjet-3.3.0.tar.gz'
cd fastjet-3.3.0/
./configure
make -j4
sudo make install
```

1.3 Installing Pythia8:

```
wget -c 'http://home.thep.lu.se/~torbjorn/pythia8/pythia8230.tgz'
tar -xf 'pythia8230.tgz'
cd 'pythia8230/'
./configure '--with-hepmc2'
make -j4
sudo make install
```

1.4 Installing Root-6

Download the binary distribution from https://root.cern.ch/content/release-61206 for the appropriate platform and extract. Once you have extracted, you will get a directory "root" wherever you extracted, cd to this directory and run `source root/bin/thisroot.sh` (assuming you are using bash

shell, otherwise modify accordingly). If you are installing `root` to a non standard directory, it is preferable to put "source root/bin/thisroot.sh" into your `" /.bashrc"` (or some other initialization file for your shell).

If this method does not produce a good executable then you should compile root from source, please follow the instructions from the official (https://root.cern.ch/building-root) site (please do not use non-standard scripts / commands from random forums).

1.5 Installing `Delphes`

```
wget -c 'http://cp3.irmp.ucl.ac.be/downloads/Delphes-3.4.1.tar.gz'
tar -xf './Delphes-3.4.1.tar.gz'
cd './Delphes-3.4.1/'
'./configure'
make -j4
```

Delphes does not have any "make install" but all the executables you need are produced in the source directory and can be copied anywhere required. An important step required is to manually copy "libDelphes.so" to '/usr/local/lib' or any other library search path and also copy all the root dictionaries (`'ClassesDict_rdict.pcm'`, `'ExRootAnalysisDict_rdict.pcm'`, `'FastJetDict_rdict.pcm'`, `'ModulesDict_rdict.pcm'`) to the root dictionary search path and copy the header files (`'classes'`, `'external'` folders) to the default headers search paths (`'/usr/local/include'`).

2 Generating the events:

The event generation is done in `PYTHIA8` and written in `HepMC` formate. The example program `main41.cc` inside pythia examples folder contains a sample program to generate events using `PYTHIA8` and write to `HepMC` formate. `HepMC` files produced tend to be very big so its recommended to not store them on disk instead just generate them and use them in `Delphes` on the fly, this is done using fifo. The complete arrangement can be found in the `Generator` class of `all.cc` (this also requires `Delphes` to be installed and in the accessible in the folders listed in `PATH` variable of the shell). `Delphes` also requires the detector model to simulate, we are using `delphes_card_CMS.tcl` (this does not simulate any pile-up).

3 Results:

A simple starting point for tagging might be:
(arXiv:0802.2470 [hep-ph]) **Jonathan M. Butterworth, Adam R. Davison, Mathieu Rubin, Gavin P. Salam:** Jet substructure as a new Higgs search channel at the LHC.
This method is incorporated in the class `NewHEPHeaders::MassDropTagger::HardSubStructureFinder` of the file `NewHEPHeaders4.hh`

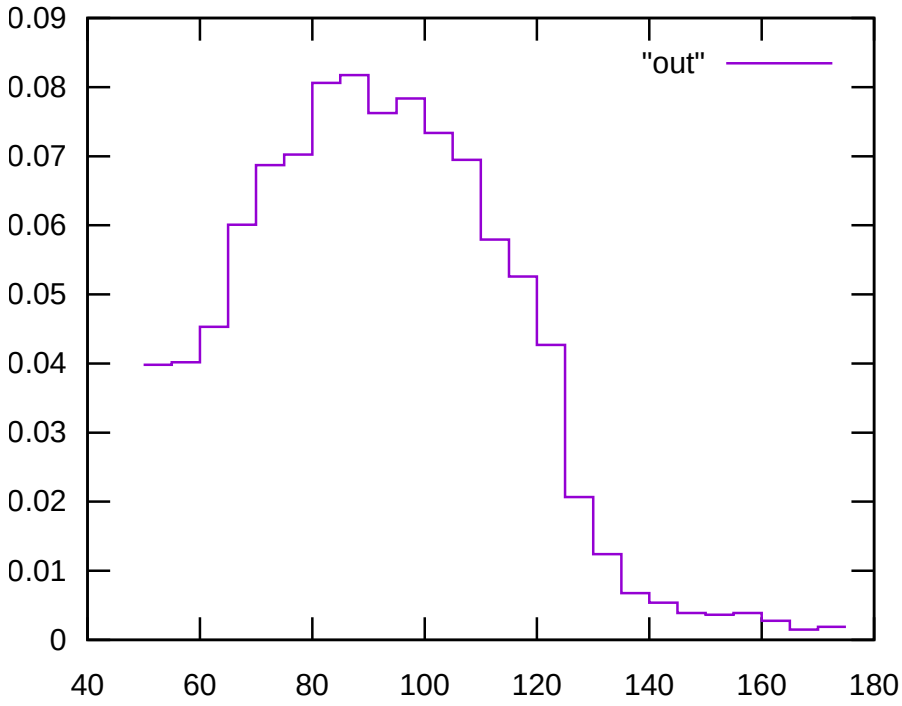

Figure 1: The invariant mass of the leading (by p_T) fat jet ($R = 1.0$, $p_T^j > 100$ GeV) from the tutorial session at IISER, Pune.

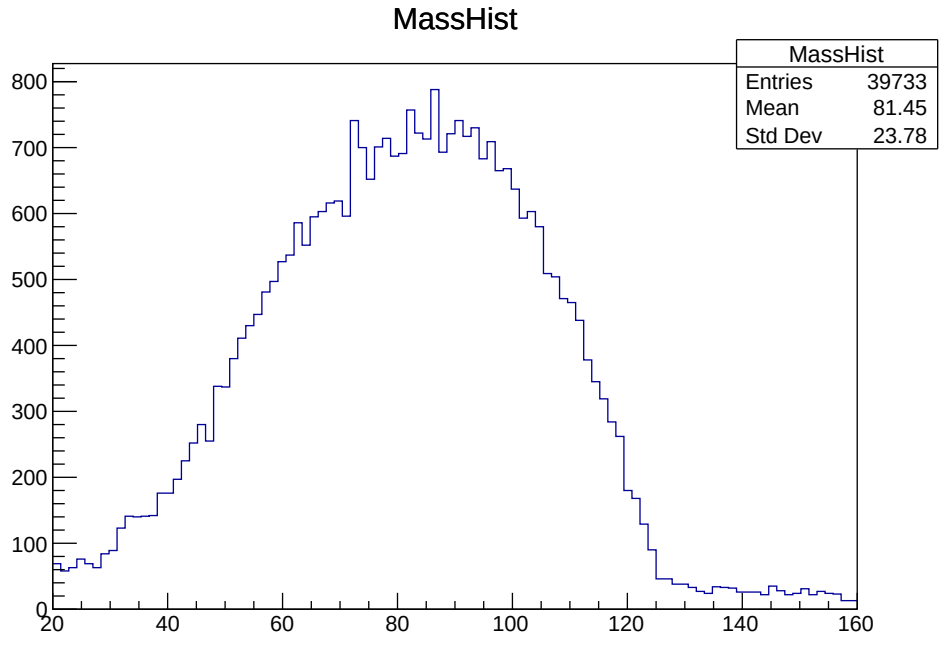

Figure 2: The invariant mass of the leading (by p_T) fat jet ($R = 1.0$, $p_T^j > 100$ GeV) after applying the mass drop check (for 2 prong) and filtering on the fat jet. Generator level vectors are used here without any `Delphes` effects.

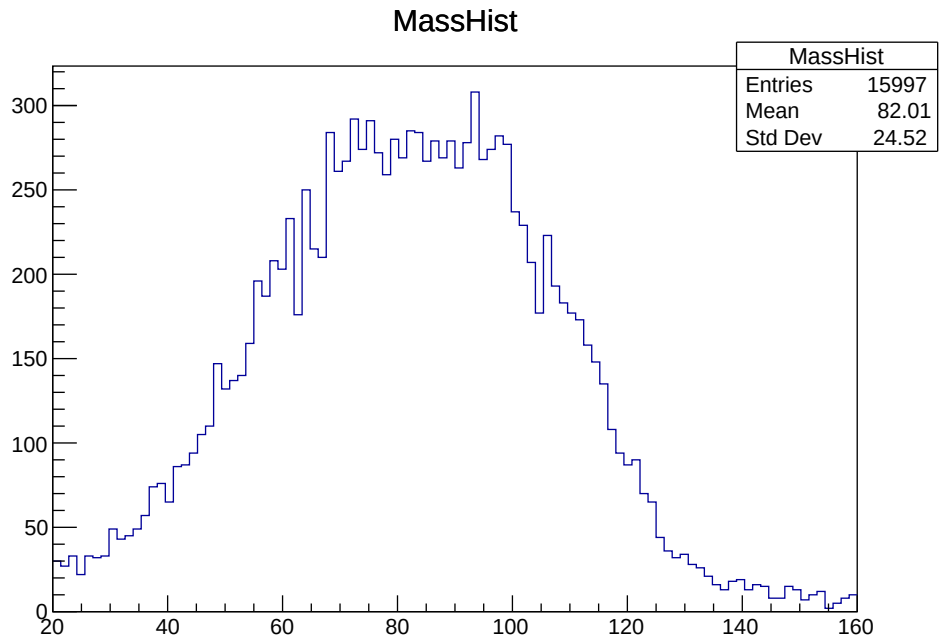

Figure 3: The invariant mass of the leading (by p_T) fat jet ($R = 1.0$, $p_T^j > 100$ GeV) after applying the mass drop check (for 2 prong) and filtering on the fat jet. EFlow vectors are used here for the event type Higgs ($\rightarrow \tau\bar{\tau}$) + Z ($\rightarrow \nu\bar{\nu}$) with a p_T cut of 500 GeV on phase space.

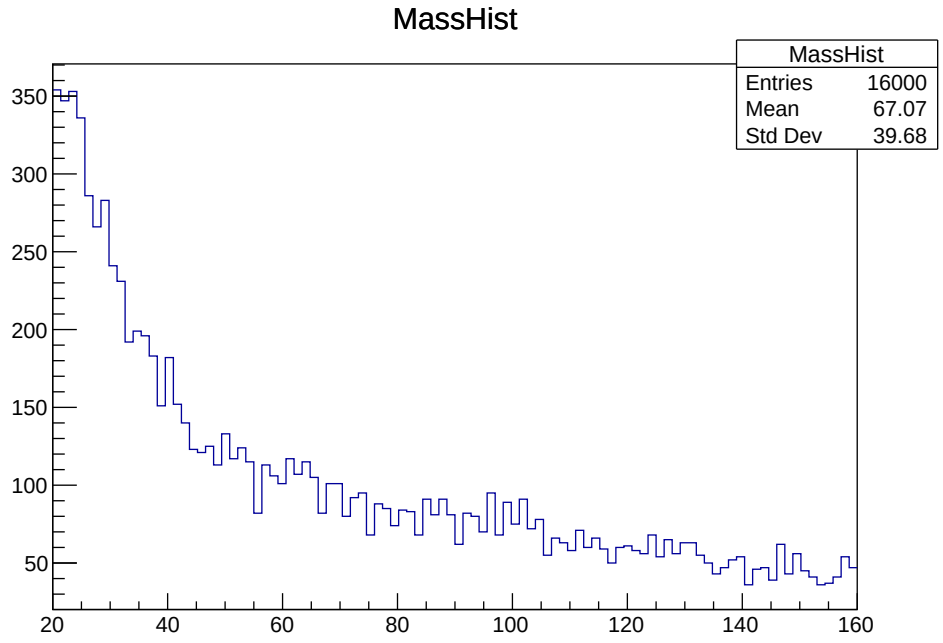

Figure 4: The invariant mass of the leading (by p_T) fat jet ($R = 1.0$, $p_T^j > 100$ GeV) after applying the mass drop check (for 2 prong) and filtering on the fat jet. EFlow vectors are used here for the event type QCD with a p_T cut of 500 GeV on phase space.

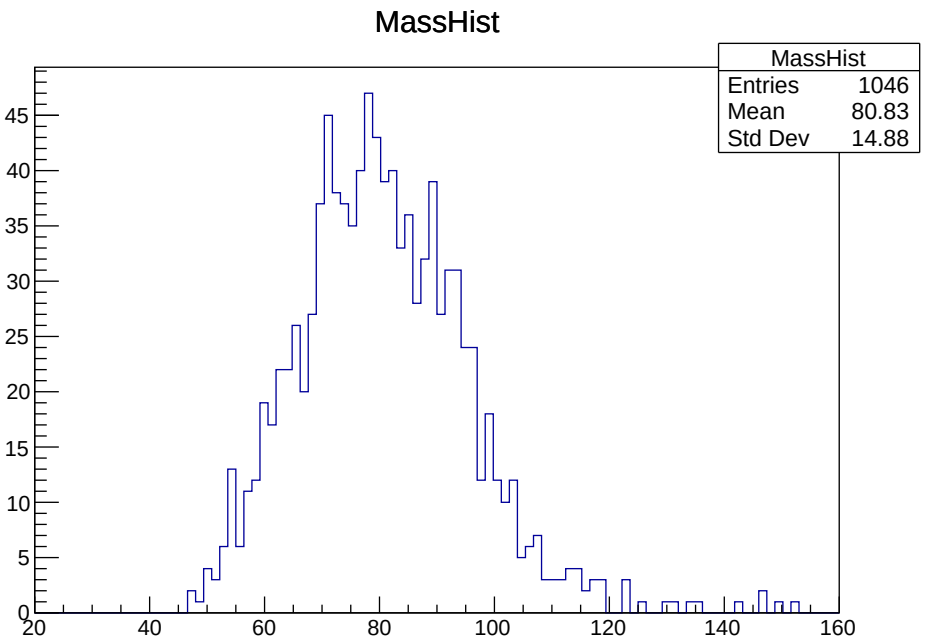

Figure 5: The Di-τ invariant mass (obtained by checking for τ-tagged jets in `Delphes`) for the event type Z ($\rightarrow \nu\bar{\nu}$) with no p_T cut on phase space.

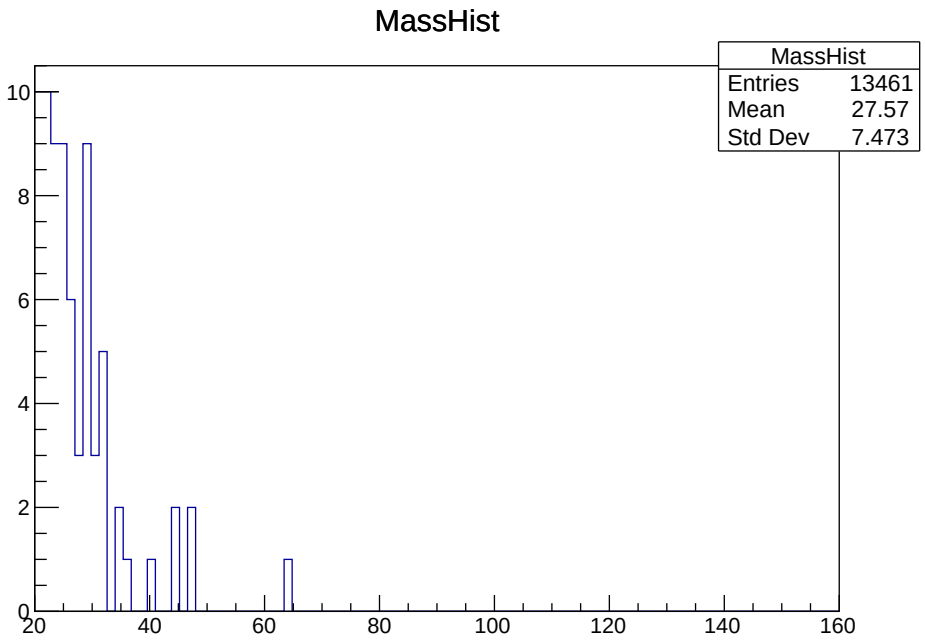

Figure 6: The invariant mass of the leading (by p_T) fat jet ($R = 1.0$, $p_T^j > 100$ GeV) after applying the mass drop check (for 2 prong) and filtering on the fat jet. EFlow vectors are used here for the event type Z ($\rightarrow \nu\bar{\nu}$) with no p_T cut on phase space.

3.1 An implementation of the BDRS algorithm:

```
class HardSubStructureFinder {
private:
    double max_subjet_mass, mass_drop_threshold, Rfilt, minpt_subjet;
    double mhmax, mhmin, mh; double zcut, rcut_factor;
    size_t nfilt;
    inline void find_structures (const fastjet::PseudoJet & this_jet) {
        fastjet::PseudoJet parent1(0,0,0,0), parent2(0,0,0,0);
        bool haskid=this_jet.validated_cs()->has_parents(this_jet,parent1,parent2);
        if(haskid) {
            if (parent1.m()<parent2.m()) {std::swap(parent1,parent2);}
            double kidmass    = parent1.m()  + parent2.m() ;
            double parentmass = this_jet.m()               ;
            if(kidmass<parentmass*mass_drop_threshold){
                t_parts.push_back(parent1);
                t_parts.push_back(parent2);
                return;
            } else if (parent1.m()>mass_drop_threshold*parent2.m()) {find_structures(parent1);return;}
        } else {return;}
        if(this_jet.m()<max_subjet_mass){t_parts.push_back(this_jet);}
        else {
            fastjet::PseudoJet parent1(0,0,0,0), parent2(0,0,0,0);
            bool haskid = this_jet.validated_cs()->has_parents(this_jet,parent1,parent2);
            if (haskid) {
                if (parent1.m()<parent2.m()) {std::swap(parent1,parent2);}
                find_structures(parent1);
                if (parent1.m()<mass_drop_threshold*this_jet.m()) {find_structures(parent2);}
            }
        }
    }
    inline void run (fastjet::PseudoJet&injet) {
        t_parts.clear(); find_structures(injet);
        if(t_parts.size()>1) {
            t_parts=sorted_by_pt(t_parts);
            size_t i=0; size_t j=1;
            triple = fastjet::join (t_parts[i],t_parts[j]) ;
            filt_tau_R = std::min ( Rfilt , 0.5*sqrt(t_parts[i].squared_distance(t_parts[j])) ) ;
            fastjet::JetDefinition filtering_def(fastjet::cambridge_algorithm,filt_tau_R);
            fastjet::Filter filter(filtering_def,fastjet::SelectorNHardest(nfilt)*fastjet::SelectorPtMin(minpt_subjet));
            taucandidate=filter(triple); filteredjetmass=taucandidate.m();
            if((mhmin<filteredjetmass)&&(filteredjetmass<mhmax)&&(taucandidate.pieces().size()>1)){
                fastjet::JetDefinition    reclustering (fastjet::cambridge_algorithm,10.0)   ;
                fastjet::ClusterSequence cs_top_sub    (taucandidate.pieces(),reclustering) ;
                tau_subs=sorted_by_pt(cs_top_sub.exclusive_jets(2));
                if (tau_subs[1].perp()>minpt_subjet) {
                    HiggsTagged=true;
                    Higgs=tau_subs[0]+tau_subs[1];
                    deltah=CPPFileIO::mymod(taucandidate.m()-mh);
                    tau_hadrons=taucandidate.constituents();
                    double Rprun=injet.validated_cluster_sequence()->jet_def().R();
                    fastjet::JetDefinition jet_def_prune(fastjet::cambridge_algorithm, Rprun);
                    fastjet::Pruner pruner(jet_def_prune,zcut,rcut_factor);
                    prunedjet=pruner(triple);
                    prunedmass=prunedjet.m();
                    unfiltered_mass=triple.m();
                }
            }
        }
    }
    inline void initialize () {
        t_parts.clear(); tau_subs.clear(); tau_hadrons.clear();
        max_subjet_mass=30; Rfilt=0.3; minpt_subjet=20;
        mass_drop_threshold=0.7; nfilt=4; filteredjetmass=0.0;
        mh=125.0; mhmax=mh+100.0; mhmin=mh-100.0; filt_tau_R=0;
        zcut=0.1; rcut_factor=0.5; prunedmass=0.0; unfiltered_mass=0.0;
        deltah=10000; HiggsTagged=false;
    }
public:
    double             filteredjetmass , deltah    , filt_tau_R   , prunedmass    , unfiltered_mass ;
    pseudojets         tau_subs        , t_parts   , tau_hadrons ;
    fastjet::PseudoJet prunedjet       , triple    , Higgs        , taucandidate ;
    bool HiggsTagged ;
    inline void operator () () {initialize();}
    inline void operator () (fastjet::PseudoJet&injet) {run(injet);}
    HardSubStructureFinder(){initialize();}
    ~HardSubStructureFinder(){}
};
```