Code Overview

Yi Chen (CERN)

TH Institute on HI Jet, 2017 Aug 24

Logistics

- Code location https://github.com/JetQuenchingTools/JetToyHI
- It should work out of the box for the installation on lxplus
  - Takes about 15 minutes to compile everything
- Also works on personal computers (linux / mac), but it is a bit more involved

Sample Location

- Currently we have JEWEL (2.2.0), Pythia8, and thermal background generated
- They can be found on lxplus at this location /eos/project/j/jetquenching/JetWorkshop2017/samples/
- The format of the event is basically a list of particle momentum vectors with PDG ID of the particles (in plain ASCII text)

Example Code

- An example code is provided in the package: runFromFile.cc

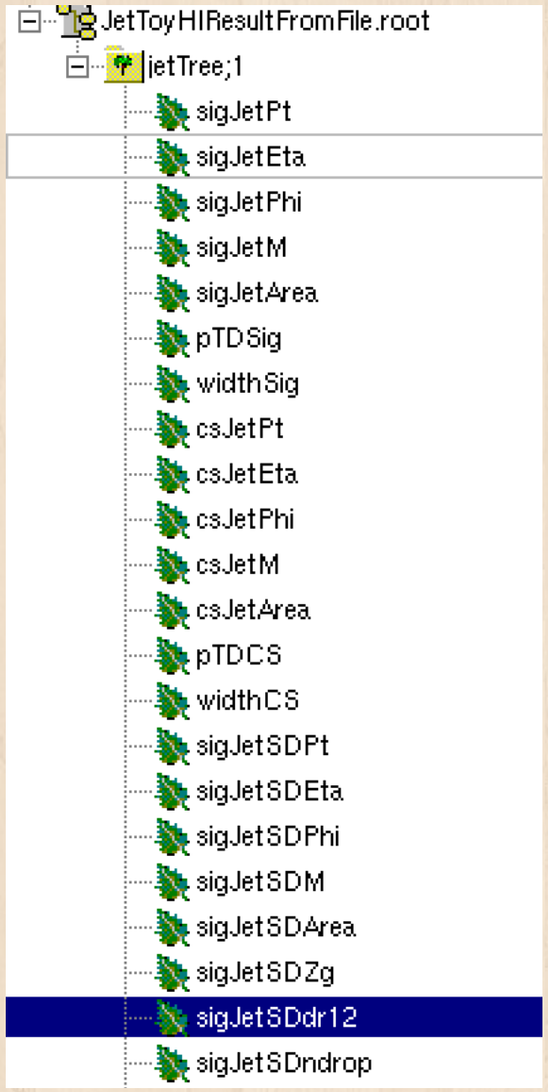
- By default it reads a event and a background event, mixes them together, do jet clustering, background subtraction, grooming, and writes out interesting quantities into a plain root tree

- An example plotting macro is provided to make some simple plots at plot/plotJetEnergyScale.C

Current Functionality

Name	Description
csSubtractor.hh	Takes a jet and do constituent subtraction
csSubtractorFullEvent.hh	Do subtraction on the full event
Angularity.hh	Calculates width, pTD or other jet quantities
randomCones.hh	Random cone study
skSubtractor.hh	Runs soft killer algorithm on jets
softDropGroomer.hh	Runs soft drop
softDropCounter.hh	Runs iterative soft drop

The output



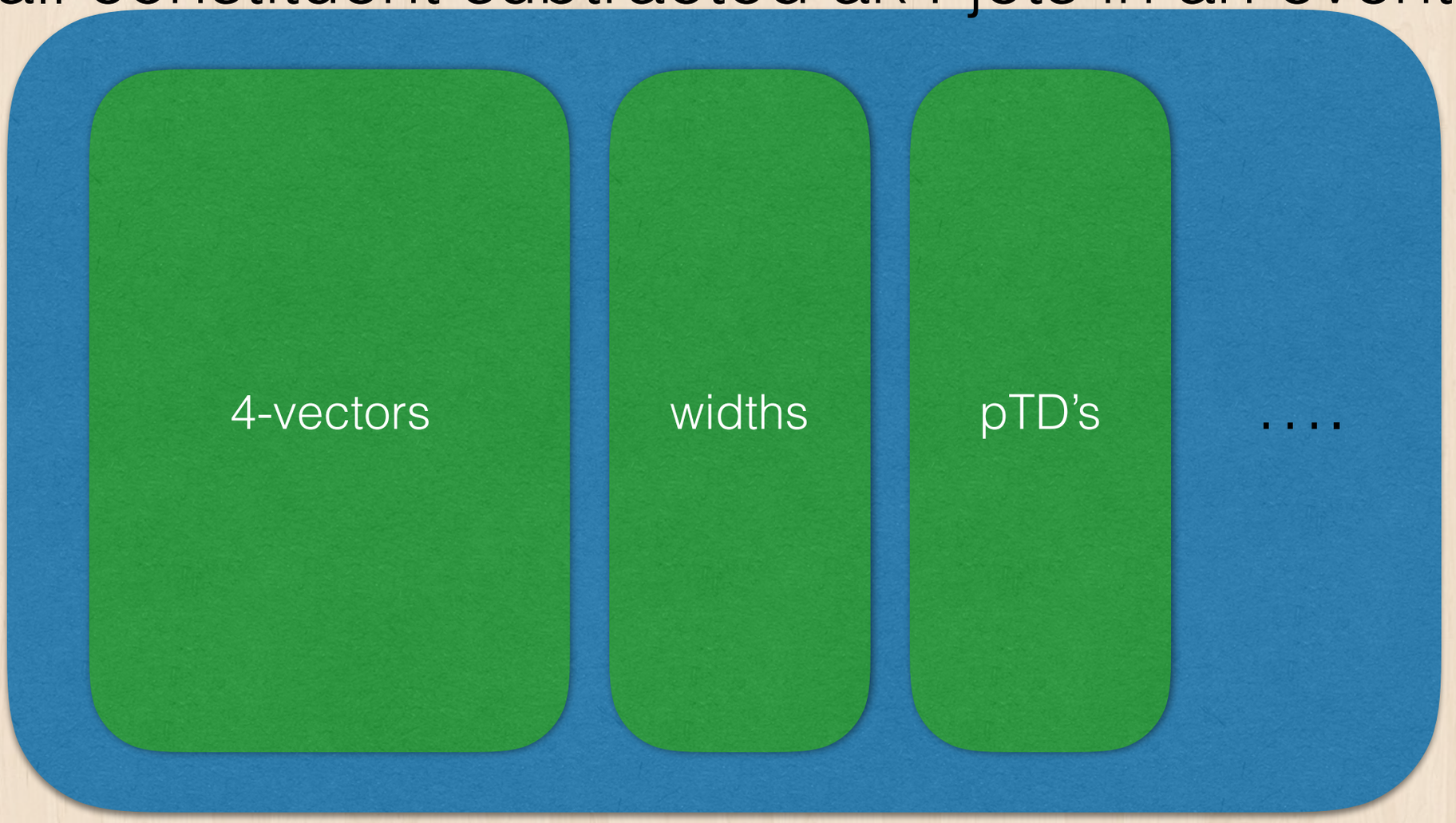
Each entry is an event

Jet quantities are stored as vectors

Default terminology:
“sigJet” = signal jet (jewel/pythia)
“csJet” = mixed + subtracted jets
“sigJetSD” = soft-dropped sigJet
“csJetSD” = soft-dropped csJet

Jet Collection

all constituent subtracted ak4 jets in an event



As an example…

- If you want to add a new quantity for signal jets

```
//calculate some angularities
std::vector<double> widthSig; widthSig.reserve(jetCollectionSig.getJet().size());
std::vector<double> pTDSig;   pTDSig.reserve(jetCollectionSig.getJet().size());
for(fastjet::PseudoJet jet : jetCollectionSig.getJet()) {
  widthSig.push_back(width.result(jet));
  pTDSig.push_back(pTD.result(jet));
}
jetCollectionSig.addVector("widthSig", widthSig);
jetCollectionSig.addVector("pTDSig", pTDSig);
```

```
//calculate some angularities
std::vector<double> widthSig; widthSig.reserve(jetCollectionSig.getJet().size());
std::vector<double> pTDSig;   pTDSig.reserve(jetCollectionSig.getJet().size());
std::vector<double> someVector;
for(fastjet::PseudoJet jet : jetCollectionSig.getJet()) {
  widthSig.push_back(width.result(jet));
  pTDSig.push_back(pTD.result(jet));
  someVector.push_back(1000);
}
jetCollectionSig.addVector("widthSig", widthSig);
jetCollectionSig.addVector("pTDSig", pTDSig);
jetCollectionSig.addVector("someVectorSig", someVector);
```