Tutorial ROOT with fastjet

Laboratório de Instrumentação e Física experimental de Partículas

July 16, 2019

Introduction

This tutorial is optional. So far all tutorials were wide scope, meaning that they were meant to teach you general things on ROOT that are common to most of the projects you'll be involved with. Fastjet however is only useful if you're gonna use jets in your analysis. Nonetheless it's good to understand the basic of how it works. In this tutorial you'll use the ROOT macro you've built in the advance ROOT tutorial, convert it to a compilable C++ program, make a local instalation of fastjet and link it to your code.

One step at a time

1. Convert your macro into a C++ compilable program. For this you need to change the name of your function to main, and the return type from void to int. You'll need to add the headers of all the ROOT objects you use, i.e:

```
#include "TFile.h"
#include "TTree.h"
...
```

2. Compile it with:

```
g++ <yourc++programname> -o out.exe
$(/usr/share/root/bin/root-config --cflags --libs) -ltbb
```

3. If everything works fine move on, ask help otherwise. To run the code you need to set an env variable first (only need to this once per terminal session):

```
export LD_LIBRARY_PATH=/usr/share/root/lib/
```

   and then run the output.

4. Now we need to install fastjet locally. To do so you have the following guiding steps:

   (a) Go to the directory /home/lip and make a directory called fastjet:

```
mkdir fastjet
```

   (b) Get the source code for fastjet:

```
wget http://fastjet.fr/repo/fastjet-3.3.2.tar.gz
```

   (c) Uncompress the archive:

```
tar xzvf fastjet-3.3.2.tar.gz
```

   (d) Enter the uncompressed archive:

```
cd fastjet-3.3.2
```

(e) Configure the installation with your created directory as the install location. If you downloaded the source and created the directory fastjet in your home, the code will be:

```
./configure --prefix=/home/lip/fastjet
```

(f) After the configuration is completed compile the source code:

```
make -j4
```

(g) Finnaly install the software:

```
make install
```

5. After installing fastjet include the following headers and namespace to the beggining of your code:

```
#include "fastjet/PseudoJet.hh"
#include "fastjet/ClusterSequence.hh"
#include "fastjet/Selector.hh"

using namespace fastjet;
```

6. Now you have everything in your code to use fastje objects. Compile the code to check if all is well with:

```
g++ <yourc++programname> -o out.exe
$(/usr/share/root/bin/root-config --cflags --glibs --libs) -ltbb
$(/home/lip/fastjet/bin/fastjet-config --cxxflags --libs)
```

7. If all is indeed well move on to the next steps, ask for help otherwise.

8. The first thing we need to tell our code is what is a jet, or rather how will it reconstruct the jet. Jets are clusters of particles more or less colimated. To reconstruct them we need to declare a jet definition. Include the following line of code in the beggining of your main function:

```
JetDefinition jet_def( antikt_algorithm, .4 );
```

9. What we are doing is telling our code that jets will be reconstructed with the anti-kt algorithm (more ref) with a jet parameter of 0.4 (which is naively the radius of the jet).

10. Next we need to save our particles in objects that fastjet can deal with namely PseudoJet. For this create a vector of PseudoJets (just like you would a vector of doubles) and in your particle loop add a line like:

```
PseudoJet part( (*px)[ iPar ], (*py)[ iPar ], (*pz)[ iPar ], (*En)[ iPar ] );
parts.push_back( part );
```

were parts is your vector of PseudoJets.

11. Finally after the particle loop we define a cluster sequence, that we will use to get the event's jets:

```
ClusterSequence clust( parts, jet_def );
```

it takes in our jet definition and the particles of the event.

12. And to get our vector of jets:

```
vector< PseudoJet > jets = clust.inclusive_jets( 100 );
```