Final project: Fully automated analysis of HIJING output

Last update: 20200711

HIJING (*Heavy Ion Jet INteraction Generator*) is a widely used Monte Carlo generator in high-energy proton-proton, proton-nucleus and nucleus-nucleus collisions. The physics incorporated in this model is based on QCD-inspired models for jets production, and includes multiple mini-jet production, soft excitation, nuclear shadowing of parton distribution functions and jet interaction in dense matter.

In this final project, you are challenged to use combined **Linux**, **Bash** and **ROOT** functionalities covered in the lecture, in order to fully automate the data analysis over one typical HIJING dataset, generated in 10 separate jobs on a local batch farm.

Challenge #0: The dataset. Download the compressed HIJING dataset (the compressed size is around 170 MB) from the following direct link: https://cernbox.cern.ch/index.php/s/BJern5Ky7ajoULd . After downloading, extract the dataset (the size will be around 680 MB after this step!) by executing

```
xxxxxxxxxx
tar xf HIJING_LBF_test.tar.gz
```

This dataset corresponds to the HIJING prediction for the collisions of heavy ions (Pb-Pb) at a collision energy of 2.76 TeV (this was the collision energy of Run 1 operations, 2009-2013, at Large Hadron Collider).

Inside the directory `HIJING_LBF_test` there are 10 subdirectories named `0, 1, ..., 9`, and in each subdirectory 5 files. Each subdirectory corresponds to the working directory of a separate process that was running an independent HIJING simulation. Besides the various config or log files in each subdirectory, the most important file is ASCII file `HIJING_LBF_test_small.out`, in which the final output of HIJING is stored. Each file `HIJING_LBF_test_small.out` contains the detailed output for 10 heavy-ion collisions. Therefore, the total dataset for the analysis in the final project amounts to 10x10 = 100 heavy-ion collisions.

The file `HIJING_LBF_test_small.out` has the following example structure and content:

```
xxxxxxxxxx
... some irrelevant header information ...
 BEGINNINGOFEVENT
           1       52443   780888.375            753         175         171   703661455
           1         221             0            11    0.147132292       -0.159234047        -4.90655518         4.94190931
           2         331             0            11   -0.313573092       -0.375932842        -7.37812376         7.45607901
           3         111             0             1    -6.50095940E-02    -8.47864971E-02   -0.190811187        0.256999820
           4         211             0             1    0.497839928         7.74994195E-02   -0.866787255         1.01225448
           5        -211             0             1   -0.753928840        0.411572814       -0.657382011         1.09061456
           6         111             0             1    0.401703835       -0.226781890        0.116536394        0.494572222
           7         321             0             1   -0.139996752         2.07054317E-02    0.166310146        0.539747834
           8        -311             0            11     6.48042858E-02     3.16939428E-02     2.06294954E-02    0.503323913
           9        -211             0             1    0.223366082       -1.79702844E-02    0.514695168        0.578458726

... not showing 40k+ lines ...

       52434         130         52352             1    0.333040059       -0.420676589       -67.7665329         67.7704849
       52435         310         52354             1   -0.198645145         2.50143819E-02   -47.0591469         47.0622101
       52436         111         52378             1   -0.130700290       -0.176906317       -0.433586091        0.504579365
       52437         111         52378             1     1.17687508E-02    -2.19080187E-02    -5.60622960E-02    0.148278296
       52438         111         52378             1     4.29953672E-02     3.46547589E-02   -0.177609831        0.229825601
       52439        -211         52390             1   -0.115880452       -0.135553151       -0.383471161        0.445355147
       52440         211         52390             1    0.667943239       -0.345367700        -2.69137526         2.79793072
       52441         310         52416             1   -0.294742197       -0.217407599         4.86402702         4.90311813
       52442         310         52424             1     3.96723598E-02    0.293893129        -1.05792081         1.20617115
       52443         310         52428             1   -0.235836297       -0.537985206        -21.5725250         21.5862579
 BEGINNINGOFEVENT
           2       49425   758120.688            644         172         168  1821316801
           1        -213             0            11    0.433982253       -0.860428333        -13.9620905         14.0100384
           2         113             0            11    0.609804869       -0.628097296        -7.59074020         7.68368912
           3         211             0             1    0.105007850        -6.46358531E-04   -0.609461606        0.634002090
           4        -213             0            11    0.421225607       -0.366191477        -11.0717869         11.1153040
           5         223             0            11    -9.12701711E-02    0.117829926        -19.7575741         19.7740669
           6         213             0            11    -5.82319610E-02    0.232945740        -11.1643286         11.1927309
```

```
        7          223            0            11    0.786985457        4.14502472E-02    -16.2853432          16.3231220
        8          331            0            11    0.749920487       -0.644374013       -7.72359276          7.84525061
        9         -211            1             1    0.168157816        2.57476717E-02    -3.08301115          3.09085488

... not showing 40k+ lines ...

    49416          211        49366             1    0.277626634        1.59957302        -99.0034943          99.0168991
    49417          111        49366             1    0.569299519       0.941452265        -95.1648254          95.1712799
    49418         -211        49367             1   -0.295076400       0.202438921        -62.0908966          62.0920868
    49419          211        49367             1    -9.28900763E-03    5.50258160E-03    -15.6468039          15.6474295
    49420          221        49367            11   -0.769395888       0.276162803        -161.382507          161.385498
    49421           22        49384             1    0.230599046        8.52777585E-02   -0.653446436         0.698169351
    49422           22        49384             1   -0.255980730      -0.116494276       -0.381368935         0.473855704
    49423          211        49420             1   -0.277680546        8.05142373E-02    -34.2214851          34.2229919
    49424         -211        49420             1   -0.280915141        2.66515389E-02    -66.7800903          66.7808304
    49425          111        49420             1   -0.210800111       0.168997005        -60.3809128          60.3816643

... and so on for remaining events ...
```

The meaning of different entries above is as follows:

1. The beginning of data for each new event is marked with the tag **BEGINNINGOFEVENT**
2. In the very next line is the event summary data (e.g. event number, the total number of particles, total energy, etc.)
3. After that line, each line holds information about individual particles. For instance, the entries in the line

```
xxxxxxxxxx
3           211              0              1    0.105007850        -6.46358531E-04   -0.609461606        0.634002090
```

have the following meaning:

- `3` : particle label within a particular event
- `211` : PID, i.e. particle identity (211 = positively charged pion, -2212 = antiproton, etc.). To get the standardized PID code for all particles in high-energy physics, consult http://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf
- `0` : this is the primary particle, i.e. this particle is not a product of resonance decay. Otherwise, this column indicates the label of the parent particle
- `1` : final or directly produced particle (alternatively, '11' in the 4th column indicates that this particle has decayed)
- `0.105007850` : *x* component of momentum (in GeV/c)
- `-6.46358531E-04` : *y* component of momentum (in GeV/c)
- `-0.609461606` : *z* component of momentum (in GeV/c)
- `0.634002090` : particle energy (in GeV)

Challenge #1: Splitting. Develop the script **Splitter.sh** which takes one argument, the top directory to your local HIJING dataset. That script splits in each of the subdirectories `0, 1, ..., 9` the large HIJING output file `HIJING_LBF_test_small.out` in 10 separate files named `event_0.dat, ..., event_9.dat`. Each of these new files contains the data only for a particular event.

At the end of this step, the situation in your local dataset is schematically as follows:

```
xxxxxxxxxx
<top-directory>/0:
HIJING_LBF_test_small.out
event_0.dat
...
event_9.dat

<top-directory>/1:
HIJING_LBF_test_small.out
event_0.dat
...
event_9.dat

...
...

<top-directory>/9:
```

```
HIJING_LBF_test_small.out

event_0.dat

...

event_9.dat
```

Hint: Use **grep -n BEGINNINGOFEVENT HIJING_LBF_test_small.out** to get the line numbers at which the entry for each new event begins. Then, programmatically extract those line numbers with **awk**. Finally, the output of **awk** use as an input to **sed** to split the file `HIJING_LBF_test_small.out` into 10 chunks, each chunk corresponding to the data of one event, something like:

```
xxxxxxxxxx

sed -n 123,123456p HIJING_LBF_test_small.out > event_0.dat
```

Challenge #2: Filtering. Develop the script **Filter.sh** which takes one argument, the top directory to your local HIJING dataset. Then, it filters in each of the subdirectories `0, 1, ..., 9` out of each new file `event_?.dat` obtained in the previous step only the information for the primary particles (i.e. particles with the label `0` in the 3rd column).

Hint #1: Collect all files `event_?.dat` with **find**, loop over them via **while+read**, something like:

```
xxxxxxxxxx

while read File; do

 cd $(dirname $File) # go to the directory where the current file sits

 ... filter out the current file, programmatically its name is $(basename $File) ...

 cd - # go back

done < <(find <top-dir> -type f -name "event_*.dat")
```

Hint #2: The following pseudo code can do the actual filtering:

```
xxxxxxxxxx

cp event_0.dat backup_0.dat

while read Line; do

 ... test with awk if the 3rd field in Line is 0, if so, simply echo that line ...

done < backup_0.dat 1>event_0.dat
```

Challenge #3: Transferring. Develop the script **Transfer.sh** which takes one argument, the top directory to your local HIJING dataset. This script is responsible to process all files `event_?.dat` and store for each event for each particle its PID and kinematics (three components of momenta and energy) into **ROOT**'s famous container `TTree`. Make one `TTree` container for each event, and then all `TTree` containers save in one common **ROOT** file named `HIJING_LBF_test_small.root`, in each of the subdirectories `0, 1, ..., 9`.

TBC

Hint: As an example code snippet how to read external ASCII file directly into `TTree`, and then save that container in the ROOT file, use the following prototype:

```
xxxxxxxxxx

// Example content of external file 'basic_1.dat' is formatted as:

/*

10 100 1000 10000

20 200 2000 20000

30 300 3000 30000

*/


// Example content of external file 'basic_2.dat' is formatted as:

/*

11 111 1111 11111

22 222 2222 22222

33 333 3333 33333
```

```
44 444 4444 44444
55 555 5555 55555
*/

// For a different format (number of columns!), you need to adapt: tree->ReadFile(filename,"px:py:pz:E"); used below

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

void importASCIIfileIntoTTree(const char *filename)
{
 TFile *file = new TFile("output.root","update"); // open ROOT file named 'output.root', where TTree will be saved.
 TTree *tree = new TTree("chunk","data from ascii file"); // make new TTree

 Long64_t nlines = tree->ReadFile(filename,"px:py:pz:E"); // whatever you specify here, will be relevant when you start later reading the
 tree->Write(); // save TTree to 'output.root' file
 file->Close();
}
```

Use then the above code snippet in the following way:

```
xxxxxxxxxx
root -l -b -q importASCIIfileIntoTTree.C\(\"basic_1.dat\"\) // or abs-path to 'basic_1.dat', if macro and the file are not in the same di
root -l -b -q importASCIIfileIntoTTree.C\(\"basic_2.dat\"\) // or abs-path to 'basic_2.dat', if macro and the file are not in the same di
```

If you now inspect (e.g. with `TBrowser`) the content of `output.root` file, you will see that in contains two `TTree` containers, one for each file.

From this point onward, only the filtered dataset stored in `TTree` containers in the **ROOT** files is used in the final analysis. Such optimization is relevant especially for instance when we want to re-run over the same dataset multiple times by varying the track selection criteria, in order to estimate systematical error.

Step #2: TRENDING. Trending is an important part of dataset validation (a.k.a. quality assurance (QA)), where basically we check if some specific part of the dataset is systematically off from the rest. If so, only that specific part of the dataset is excluded from the final analysis. Please provide a trending plot 'average # of particles' vs. 'subdirectory number', and dump it as '.pdf', '.eps', '.png' and '.C' file.

Hint #1: By using 'here-documents' in **Bash** make a template **ROOT** macro for plotting, which then you will just update by using **sed** with the concrete numbers for each subdirectory. After that, when you have entries from all sudirectories, just execute the macro and dump the figures.

Hint #2: To read entries from `TTree`, please have a look at the following code snippet (which corresponds to the above example!): ```c // Example macro to read TTree from the file, and then all particles from the current TTree

void readDataFromTTree(const char *filename) {

TFile *file = new TFile("output.root","update"); // there multiple TTrees in this file, each corresponds to different event

TList *lofk = file->GetListOfKeys();

for(Int_t i=0; iGetEntries(); i++) {

TTree *tree = (TTree*) file->Get(Form("%s;%d",lofk->At(i)->GetName(),i+1)); // works if TTrees in ROOT file are named e.g. 'chunk;1', 'chunk;2'. Otherwise, adapt for your case

if(!tree || strcmp(tree->ClassName(),"TTree")) // make sure the pointer is valid, and it points to TTree { cout<<Form("%s is not TTree!",lofk->At(i)->GetName())<<endl; continue; }

//tree->Print(); //from this printout, you can for instance inspect the names of the branches

cout<<Form("Accessing TTree named: %s",tree->GetName())<<": "<<tree<<endl; Int_t nParticles = (Int_t)tree->GetEntries(); // number of particles cout<<Form("=> It has %d particles.",nParticles)<<endl;

// Attach local variables to branches: Float_t px = 0., py = 0., pz = 0. , E = 0.; tree->SetBranchAddress("px",&px); // that the name of this branch is px, you can inspect from tree->Print() above, and so on tree->SetBranchAddress("py",&py); tree->SetBranchAddress("pz",&pz); tree->SetBranchAddress("E",&E);

for(Int_t p = 0; p < nParticles; p++) // loop over all particles in a current TTree { tree->GetEntry(p); cout<<Form("%d: %f %f %f %f",p,px,py,pz,E)<<endl; }

cout<<"Done with this event, marching on...\n"<<endl;

} // for(Int_t i=0; iGetEntries(); i++)

file->Close();

}

```
xxxxxxxxxx

Use above macro for instance as:
```c
```
```

```
root -l readDataFromTTree.C\(\"output.root\"\)
```

Step #3: ANALYSIS. For the whole dataset, please provide the figure (in 4 standard formats .pdf,.eps, .png and .C) with 3 histograms plotted together, holding the distributions of transverse momentum for pions, kaons and protons. Transverse momentum is the Lorentz invariant quantity defined as: $$p_T \equiv \sqrt{p_x^2 + p_y^2}$$

Please develop a script which will utilize combined Linux, Bash and ROOT utilities covered in the Lecture, and automate the 3 data analysis steps detailed below.

The user want to use your script in the following way: ```bash source yourScript.sh abs-path-do-top-directory-with-HIJING-data

```
and get automagically at the end of the day the plots explained in the steps below.
```