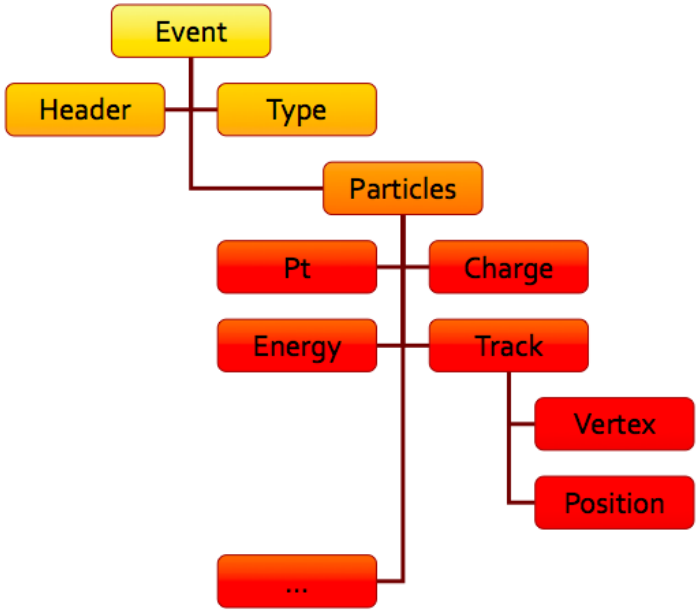
ROOT trees

Anna Simon

November 11, 2015

List mode data vs ROOT tree

x	y	z
-1.10228	-1.79939	4.452822
1.867178	-0.59662	3.842313
-0.52418	1.868521	3.766139
-0.38061	0.969128	1.084074
0.552454	-0.21231	0.350281
-0.18495	1.187305	1.443902
0.205643	-0.77015	0.635417
1.079222	-0.32739	1.271904
-0.27492	-1.72143	3.038899
2.047779	-0.06268	4.197329
-0.45868	-1.44322	2.293266
0.304731	-0.88464	0.875442
-0.71234	-0.22239	0.556881
-0.27187	1.181767	1.470484
0.886202	-0.65411	1.213209
-2.03555	0.527648	4.421883
-1.45905	-0.464	2.344113
1.230661	-0.00565	1.514559
		3.562347



The ROOT Tree is

- extremely efficient write once, read many.
- Designed to store $> 10^9$ with same data structure.
- Trees allow fast direct and random access to any entry (sequential access is the best).
- Optimized for network access (read-ahead).

How to build a ROOT tree

- Create a file:

```
1 TFile * file = TFile::Open("treeFile.root","RECREATE")
```

- Create a tree:

```
1 TTree *t1 = new TTree("t1","sample tree")
```

- Add branches to the tree
- Fill the tree with data
- Write the tree to file

What is a branch?

- a branch is like a directory
  - it can hold a simple variable, a list of variables, an object or even a collection of objects
  - the leaves are the data containers of the branch
  - it is possible to read only a sub-set of all the branches in a tree
- variables or object known to be used together should be put in the same branch
  - branches of the same tree can be written to separate files

How to create a branch

```
1 //define the variables
2 Int_t nValue;
3 Double_t E[5];
4
5 //this is a branch storing a simple variable
6 t1->Branch("var",&nValue,"nValue/I");
7
8 //this is a branch storing a fixed size array
9 t1->Branch("E",E,"E[5]/D");
```

I, D, F ... specifies the size of the data stored in the branch

How to build a ROOT tree

- Create a file:

```
TFile * file = TFile::Open("treeFile.root","RECREATE")
```

- Create a tree:

```
TTree *t1 =  new TTree("t1","sample tree")
```

- Add branches to the tree

```
//this is a branch storing a simple variable
t1->Branch("var",&nValue,"nValue/I");

//this is a branch storing a fixed size array
t1->Branch("E",E,"E[5]/D");
```

- Fill the tree with data

```
t1->Fill()
```

- Write the tree to file

```
file->Write()
```

Sample tree structures

```
*******************************************************************************
*Tree    :t         : tree                                                   *
*Entries :  2365755 : Total =      7424938681 bytes  File  Size =  659294514 *
*        :          : Tree compression factor =  11.26                       *
*******************************************************************************
*Br    0 :ADC       : ADC[180]/I                                             *
*Entries :  2365755 : Total  Size= 1703554523 bytes  File Size  =  107137243 *
*Baskets :     2421 : Basket Size=   14986788 bytes  Compression=  15.90     *
*.............................................................................*
*Br    1 :TDC       : TDC[180]/I                                             *
*Entries :  2365755 : Total  Size= 1703554523 bytes  File Size  =   91559517 *
*Baskets :     2421 : Basket Size=   14986788 bytes  Compression=  18.61     *
*.............................................................................*
```

```
*******************************************************************************
*Tree    :t2        : HECTOR data                                            *
*Entries :   298188 : Total =       230853079 bytes  File  Size =    8259555 *
*        :          : Tree compression factor =  28.11                       *
*******************************************************************************
*Br    0 :ene       : ene[2][8][2]/D                                         *
*Entries :   298188 : Total  Size=   76548080 bytes  File Size  =    2863728 *
*Baskets :     2405 : Basket Size=      32000 bytes  Compression=  26.71     *
*.............................................................................*
*Br    1 :timecfd   : timecfd[2][8][2]/D                                     *
*Entries :   298188 : Total  Size=   76557716 bytes  File Size  =     584379 *
*Baskets :     2405 : Basket Size=      32000 bytes  Compression= 130.92     *
*.............................................................................*
*Br    2 :time      : timefull[2][8][2]/D                                    *
*Entries :   298188 : Total  Size=   76550501 bytes  File Size  =    4731997 *
*Baskets :     2405 : Basket Size=      32000 bytes  Compression=  16.17     *
*.............................................................................*
*Br    3 :mult      : mult/I                                                 *
*Entries :   298188 : Total  Size=    1196424 bytes  File Size  =      26559 *
*Baskets :       38 : Basket Size=      32000 bytes  Compression=  45.01     *
*.............................................................................*
```

```
*******************************************************************************
*Tree    :t         : tree                                                   *
*Entries :  2365755 : Total =      7424938681 bytes  File  Size =  659294514 *
*        :          : Tree compression factor =  11.26                       *
*******************************************************************************
*Br    0 :ADC       : ADC[180]/I                                             *
*Entries :  2365755 : Total  Size= 1703554523 bytes  File Size  =  107137243 *
*Baskets :     2421 : Basket Size=   14986788 bytes  Compression=  15.90     *
*.............................................................................*
*Br    2 :leaf      : energy[48]/F:time[48]/F:id[48]/I:mult/I                 *
*Entries :  2365755 : Total  Size= 1372309002 bytes  File Size  =   90100729 *
*Baskets :     1938 : Basket Size=   14986788 bytes  Compression=  15.23     *
*.............................................................................*
*Br    3 :clover    : energy[12]/F:time[12]/F:id[12]/I:mult/I                 *
*Entries :  2365755 : Total  Size=  354480100 bytes  File Size  =   24576132 *
*Baskets :      518 : Basket Size=   14986788 bytes  Compression=  14.25     *
*.............................................................................*
*Br    4 :bgo       : energy[12]/F:time[12]/F                                 *
*Entries :  2365755 : Total  Size=  227141580 bytes  File Size  =   14354779 *
*Baskets :      330 : Basket Size=   14986788 bytes  Compression=  15.82     *
*.............................................................................*
*Br    5 :de        : renergy[24]/F:rtime[24]/F:rid[24]/I:senergy[8]/F:       *
*         | stime[8]/F:sid[8]/I:rmult/I:smult/I                              *
*Entries :  2365755 : Total  Size=  927490566 bytes  File Size  =  101157839 *
*Baskets :     1322 : Basket Size=   14986788 bytes  Compression=   9.17     *
*.............................................................................*
*Br    6 :e1        : renergy[24]/F:rtime[24]/F:rid[24]/I:senergy[8]/F:       *
*         | stime[8]/F:sid[8]/I:rmult/I:smult/I                              *
*Entries :  2365755 : Total  Size=  927490566 bytes  File Size  =  112963502 *
*Baskets :     1322 : Basket Size=   14986788 bytes  Compression=   8.21     *
*.............................................................................*
*Br    7 :DE        : rEne/F:rTime/F:sEne/F:sTime/F:rID/I:sID/I:rmult/I:      *
*         | smult/I                                                          *
*Entries :  2365755 : Total  Size=   75715764 bytes  File Size  =   40738532 *
*Baskets :      125 : Basket Size=   14986788 bytes  Compression=   1.86     *
*.............................................................................*
*Br    8 :E1        : rEne/F:rTime/F:sEne/F:sTime/F:rID/I:sID/I:rmult/I:      *
*         | smult/I                                                          *
*Entries :  2365755 : Total  Size=   75715764 bytes  File Size  =   42815590 *
*Baskets :      125 : Basket Size=   14986788 bytes  Compression=   1.77     *
*.............................................................................*
*Br    9 :telescope : energy/F:time/F:sID/I:rID/I:pid/I:pid_linear/F          *
*Entries :  2365755 : Total  Size=   57485818 bytes  File Size  =   33799276 *
*Baskets :      100 : Basket Size=   14986788 bytes  Compression=   1.68     *
*.............................................................................*
```

Example

Use the following code to create a tree

```
1   Float_t x,y,z; //variables to fill the branches with
2
3   TFile *outFile = TFile::Open("tree.root","RECREATE");
4   TTree *tree = new TTree("tree","Example of a ROOT tree");
5
6   tree->Branch("x",&x,"x/F");
7   tree->Branch("y",&y,"y/F");
8   tree->Branch("z",&z,"z/F");
9
10  TRandom r; //Declare a random number generator variable
11
12  for (int i=0;i<50000;i++){ //Loop 50000 times
13          x=r.Uniform(-10,10);
14          y=r.Uniform(-10,10);
15          z=r.Uniform(-10,10);
16
17      //Fill the tree with points within a uniform sphere
18      if (sqrt(x*x+y*y+z*z)<10.0){
19          tree->Fill();
20      }
21
22  }
23
24  outFile->Write();
```

Plotting data from a tree (1D)

```
1  TH1F *h1 = new TH1F("h1","x-coordinates",200,-10,10)
2  TH1F *h2 = new TH1F("h2","y-coordinates",200,-10,10)
3  TH1F *h3 = new TH1F("h3","y-coordinates for z>0",200,-10,10)
4  TH1F *h4 = new TH1F("h4","radius distribution",200,-10,10)
5  TH1F *h5 = new TH1F("h5","radius distribution for the first 1000
       events",200,-10,10)
6
7  //fill in a histogram with data and draw it
8  tree->Draw("x>>h1")
9
10 //fill in the second histogram and overlay it with the h1
11 tree->Draw("y>>h2","","same")
12
13 //fill in the histogram with data only when the condition is
       fulfilled
14 tree->Draw("y>>h3","z>0","same")
15
16 //create a new variable from the tree leaves
17 tree->SetAlias("radius","sqrt(x*x + y*y + z*z)")
18 tree->Draw("radius>>h4","","same")
19 //plot radius only for a subset of events
20 tree->Draw("radius->h5","Entry$<1000","same")
```

Creating 2D and 3D histograms

```
TH2F *h2d = new TH2F("h2d","y vs x",200,-10,10,200,-10,10)
TH2F *h2d2 = new TH2F("h2d2","y vs x with x>0",200,-10,10,200,-10,10
    )

TH3F *h3d = new TH3F("h3d","z vs y vs x",20,-10,10,20,-10,10,20,-10,
    10)
//fill in a histogram with data and draw it
tree->Draw("y:x>>h2d","","colz")
tree->Draw("y:x>>h2d2","x>0","colz")
tree->Draw("z:y:x>>h3d")
```

To make a condition using TCut:

- Plot the histogram of data that define the cut (e.g. h2d from above examples)
- using the Toolbar draw a cut around the points of interest (select one quarter of the circle)
- right click on the cut and rename it (e.g. myCut)
- draw the variable with the cut applied:

```
TH1F *h6 = new TH1F("h6","z with (y vs x) cut",200,-10,10)

tree->Draw("z>>h6","myCut","")
```

Exercise

File data.root contains energy (E) and momentum (Px, Py, and Pz) for three particle emitted during each event. For the data in tree t1, the z-axis is the beam direction.

- create new variables (theta[i]) that will store the angle (in degrees) between the particle trajectory and the z-axis plot the new variables
- plot E:theta for particle 0 and theta[0]:theta[1]
- draw a cut around the "blob" in theta[0]:theta[1] plot (theta[0]>12 && theta[1]<45)
- draw E[0] energy and E[0] with the cut on the "blob" on the same canvas
- create variables that store the rest mass of particles 0 and 1 ($m_0 = sqrt(E^2 - |P_{tot}|^2)$).
- plot the masses, what type of particles are they?

TChain

A series of files containing a tree (t1) with the same branch structure can be chained together and analyzed the same way as if it was a single tree by using TChain.

```
1  //create a chain
2     TChain chain("t1");
3
4  //add files to the chain
5     chain.Add("file1.root");
6     chain.Add("file2.root");
7     chain.Add("file3.root");
8
9  //use the chain as if it was a tree
10    chain.Draw("x");
```