The ROC curve, 2013/01/17

The ROC (Receiver Operating Characteristic) curve is a common way in physics to evaluate a selection cut discrimination performance. By comparing ROC curves of different cuts it is possible to choose a selection cut rather than another one. A ROC curve is usually obtained by plotting background rejection versus selection efficiency while varying a cut on a specific observable.

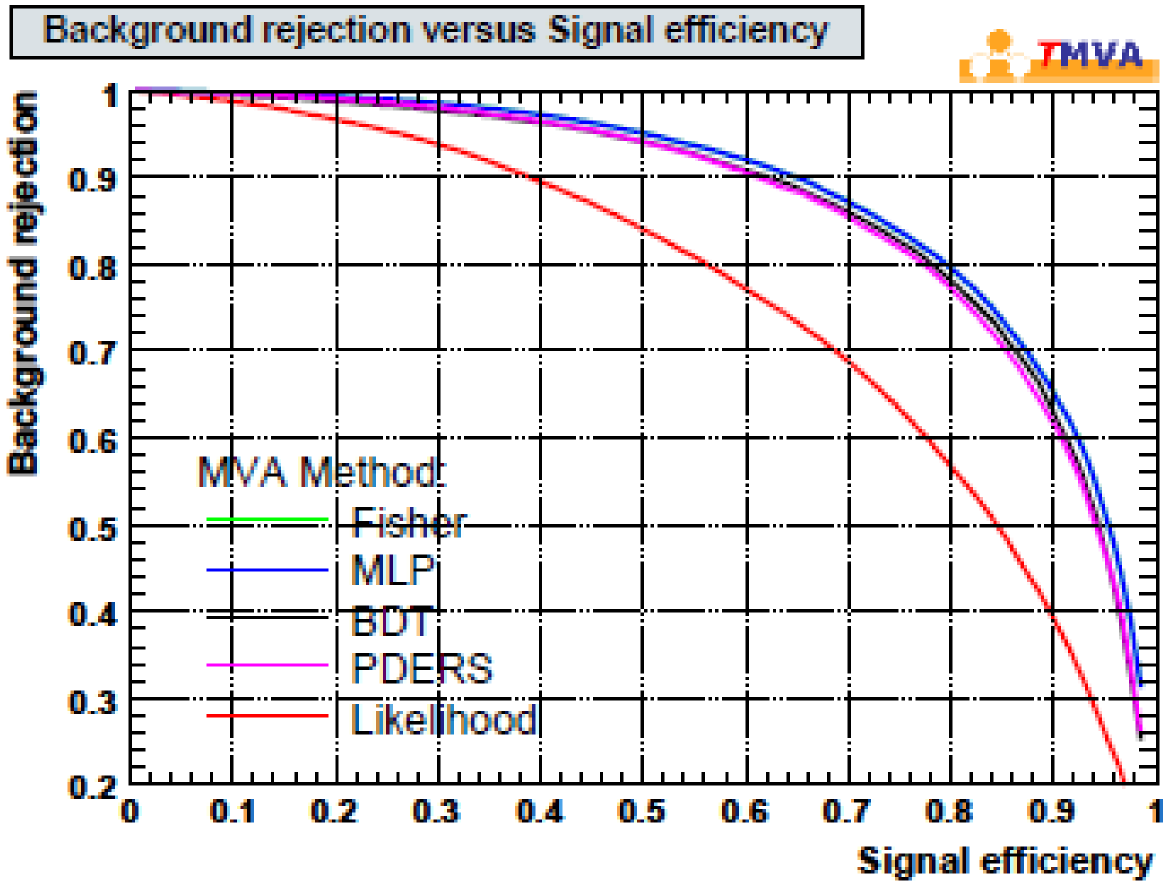
Background rejection is defined as the number of background events that are discarded (not selected) by a given cut divided by the number of events in the background sample.

Selection efficiency is defined as the number of signal events that are selected by a given cut divided by the number of events in the signal sample.

Both background rejection and selection efficiency values vary in the range [0,1].

The best ROC curve is the one with the higher background rejection at a given selection efficiency.

The ROC curve, 2013/01/17



ROC curves example

Goals of this work

The aim of this exercise is to find out which is the best selection variable out of two given observables in order to select a signal sample with 90% efficiency.

Background informations: you will work on a set of events coming out from a PAMELA experiment simulation. The given file `/home/mocchiut/pamela/data/pamelasimu.root`
contains the TTree pamcalotree, storing data with the PamCalo class, header file: `/home/mocchiut/pamela/PamCalo/inc/PamCalo.h`
so library: `/home/mocchiut/pamela/PamCalo/lib/Linux/libPamCalo.so`.

The ROOT file contains about 5.700.000 events: protons, electrons and positrons mixed together in an energy range from 10 to 300 GeV.

Positrons to electrons ratio in this file is about 0.1 . Electrons to protons ratio in this file is about 0.007 .

In this contest, consider only the energy range [18-20] GeV and the two selection variables “noint” and “qt ≡ qtrack/qtot”. Signal events are represented by electrons, background events are represented by protons.

Exercise 1

Create an <u>executable compiled program</u> which reads the input file `/home/mocchiut/pamela/data/pamelasimu.root`

and gives as output a new ROOT file containing a TTree with three variables (a TBranch for each one):

- pID
- noint
- qt

where qt is defined as qtrack/qtot.

Save into the new file events which satisfy ALL the following three conditions:

1. events are simulated protons or electrons (hint: `if ( pID==0 || pID ==1)...`);
2. the event energy is between 18 and 20 GeV (hint: pay attention to the sign of "energy"! use "fabs" function);
3. events for which the so-called energy-momentum-match is between 50 and 500 MIP/GeV (hint: `fabs(pc->qtot/pc->energy) > 50. && fabs(pc->qtot/pc->energy) < 500.` , where `pc` is an object of the class PamCalo).

Hints:

- to compile, remember to add also the compilation flags:

```
-I/home/mocchiut/pamela/PamCalo/inc -L/home/mocchiut/pamela/PamCalo/lib/Linux/ -lPamCalo
```

- to run, remember to export LD_LIBRARY_PATH:

```
export LD_LIBRARY_PATH=/home/mocchiut/pamela/PamCalo/lib/Linux/:$LD_LIBRARY_PATH
```

- the output file should have a size of 465K, if you have quota problem you can write the output on the linux temporary directory "/tmp/".

Exercise 2

Create a ROOT-CINT script which reads the output file of exercise 1 (should be similar to this one: `/home/mocchiut/scripts/EM_output_ex1.root` , use this file if you are not able to complete or run exercise 1) and gives as output on the screen and on the disk (pdf format) the ROC curves for the "noint" and "qt"variables.

Hints:

- Use TGraphErrors to plot the curves, use 11 point for each curve.
- To get each ROC curve you need a nested loop: in the main loop you change at any iteration the selection cut; for each selection cut of the main loop you must loop over all the events of the file to count the total number of protons (hint: `if ( pID== 0 ) bkgsample++;`), the number of protons that are NOT selected (hint: `if ( pID==0 && noint > nointUpperCut ) bkgnotsel++;`), the total number of electrons (hint: `if (pID == 1) sigsample++;`) and the total number of selected electrons (hint: `if ( pID==1 && noint <= nointUpperCut )  sigsel++;`) where the counters (`bkgsample, bkgnotsel, ...`) are integer variables set to zero at each iteration of the main loop.
- Inside the main loop and after the loop over the events you should calculate the background rejection value (hint: `bkgnotsel/bkgsample` , pay attention to integer numbers, you will need casting!) and the signal efficiency (hint: `sigsel/sigsample`) for that selection cut and save these two values into two arrays of floats (of 11 elements) that will be used to create the TGraphErrors objects. As errors use the square root of the number of events that pass the selection (hint: `sqrt(bkgnotsel)/bkgsample` and `sqrt(sigsel)/sigsample`) to be saved in other two arrays of floats.
- For the "noint" variable use as selection cut formula this one: `Float_t nointUpperCut = 10. - j;` where j is the main loop iterator that goes from 0 to 10 included. Signal events are selected if `noint` is lower equal than `nointUpperCut` .
- For the "qt" variable select signal events for which qt is greater than `qtLowerCut` and lower than `qtUpperCut`, where `Float_t qtUpperCut = 0.55 + j * 0.03;` and `Float_t qtLowerCut = 0.55 -j*0.03;`
- Use different symbols and colors for the two ROC curves. Adding a legend could be nice (look for TLegend in the reference manual, there are examples of how to plot legends).

Exercise 3

Change the ROOT-CINT script of exercise 2 adding a fit of the ROC curves. Print out on the screen the value of the two functions at 90% efficiency.

Which variable gives the best background rejection at 90% signal efficiency?

Hints:

- Use the following formula for fitting: "`[0]+[1]/(x-[2])`".
- Set the fitting range between 0. and 1.
- Set the initial parameter to (2.,1.,2.) in both cases.
- Set fitting options to "`MER`".

Preparing the output

- create a directory named with the following format: YourInitials_C++2012

(for example in my case it would be: EM_C++2012)

put inside this directory ALL the files you want me to correct and look at.

- ALL files names format (but Makefile, if any) must be like:

YourInitials_something.extension

(for example in my case I would create files: EM_main.cpp, EM_myscript.C, EM_OutputHistogram1.pdf, etc. etc.)

- create a README text file (named like EM_README.txt), inside the file write:
  - **your name and surname**
  - a list of the files you are submitting
  - **in details** how to compile and run the programs
  - any other comment and answer to question(s) rised in the exercise description
- create a compressed tarfile containing the directory:

```
bash> ls
EM_C++2012
bash> tar zcf EM_C++2012.tar.gz EM_C++2012/
```

- copy the tarzipped file on the USB key I will circulate

Timing and rules

- You have four hours time to do your work.
- You can search the web, look at manuals, look at any note you wrote during the course, etc.
- We will discuss what you have written at the oral examination on YYYY/MM/DD, until that (if needed) you can change and improve your programs. In that case prepare an electronic version we can look at during the oral examination, we will compare it to the one handed in today and we will discuss any change and/or correction.

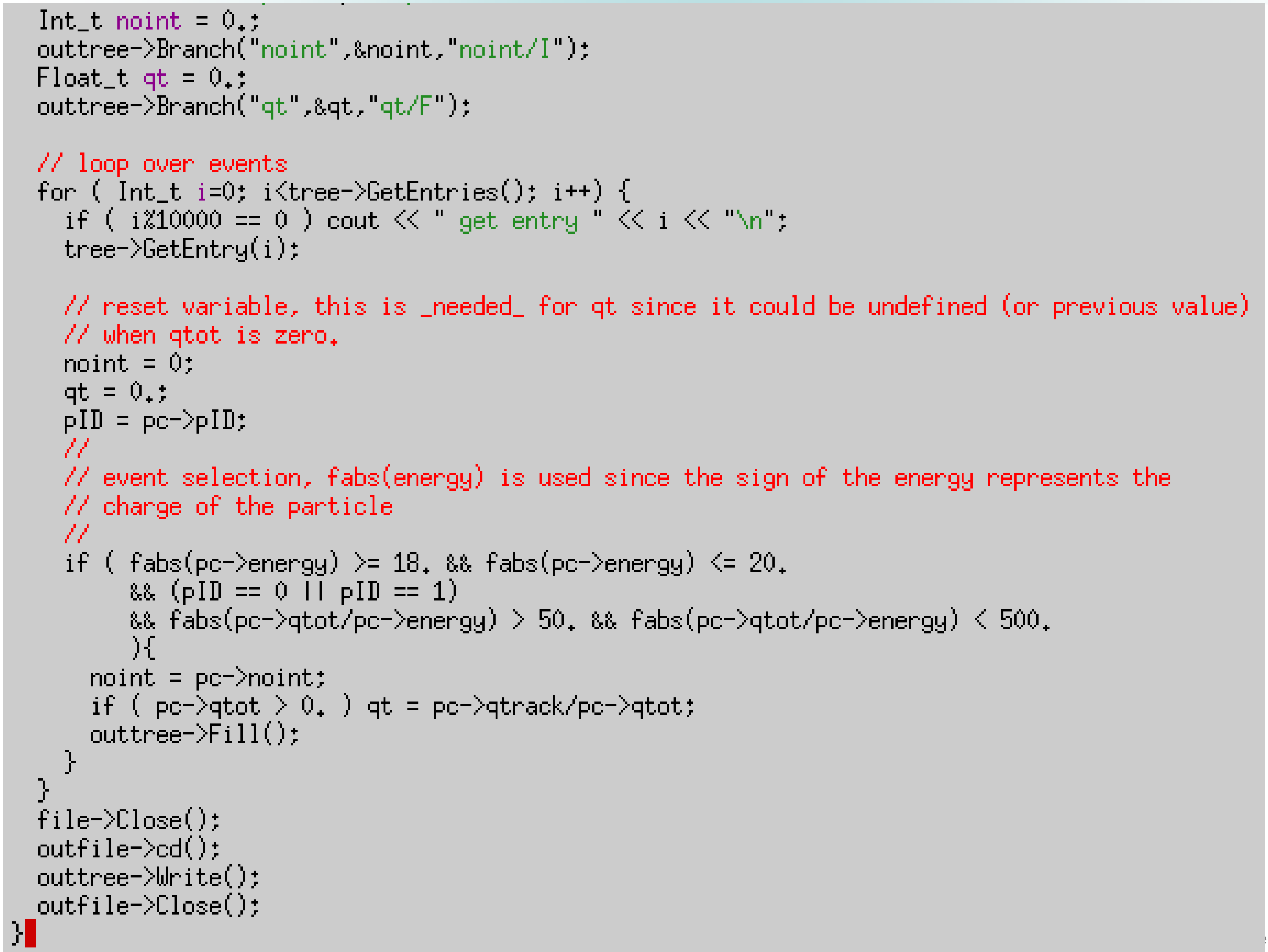
Solution exercise 1

- source code: examination_hints.pdf
- source code: slides_2_121012.pdf page 30 (main), 31 and following (commenting etc.)
- source code: mainTemplate.cpp (handling input parameters)
- how to compile: slides_7_121123.pdf page 28

Solution exercise 1

```
//
// Emiliano Mocchiutti, 17/01/2013
// This program reads the input file "pamelasimu.root", selects electrons and protons in the
// energy range 18-20 GeV applying an energy/momentum match, saves as output a ROOT file
// with pID, noint and qt=qtrack/qtot variables
//
#include <iostream>
#include <cmath>
#include <TFile.h>
#include <TTree.h>
#include <PamCalo.h>
using namespace std;

//
// main function, open files, loop over events, save file
//
void copy(TString inputFile="", TString outputFile="prova.root"){

  // Open the input file using the PamCalo class
  TFile *file = TFile::Open(inputFile,"READ");
  TTree *tree = (TTree*)file->Get("pamcalotree");
  PamCalo *pc = new PamCalo();
  tree->SetBranchAddress("PamCalo",&pc);

  // Open the output file
  TFile *outfile = TFile::Open(outputFile,"RECREATE");
  outfile->cd();
  TTree *outtree = new TTree("selectedtree","PamCalo copy of some variables");
  Int_t pID = 0;
  outtree->Branch("pID",&pID,"pID/I");
```

Solution exercise 1



```
  Int_t noint = 0.;
  outtree->Branch("noint",&noint,"noint/I");
  Float_t qt = 0.;
  outtree->Branch("qt",&qt,"qt/F");

  // loop over events
  for ( Int_t i=0; i<tree->GetEntries(); i++) {
    if ( i%10000 == 0 ) cout << " get entry " << i << "\n";
    tree->GetEntry(i);

    // reset variable, this is _needed_ for qt since it could be undefined (or previous value)
    // when qtot is zero.
    noint = 0;
    qt = 0.;
    pID = pc->pID;
    //
    // event selection, fabs(energy) is used since the sign of the energy represents the
    // charge of the particle
    //
    if ( fabs(pc->energy) >= 18. && fabs(pc->energy) <= 20.
         && (pID == 0 || pID == 1)
         && fabs(pc->qtot/pc->energy) > 50. && fabs(pc->qtot/pc->energy) < 500.
         ){
      noint = pc->noint;
      if ( pc->qtot > 0. ) qt = pc->qtrack/pc->qtot;
      outtree->Fill();
    }
  }
  file->Close();
  outfile->cd();
  outtree->Write();
  outfile->Close();
}
```

Solution exercise 1

```
//
// function used to print on STDOUT the help
//
void usage(){
  cout << "------------------------------------------------------------\n";
  cout << "This program calculates the area of a trapezium given minor basis, major basis and height\n";
  cout << "\nUSAGE:\n";
  cout << "--inputFile  filename.root \n";
  cout << "--outputFile filename.root \n";
  cout << "--help, -h   print this help \n";
  cout << "------------------------------------------------------------\n";
}



//
// main, check arguments and call main function
// the program accepts as inputs the input filename string
// and the output filename string
//
int main(int argc, char *argv[]){
  // default values for input and output filenames
  TString inputFile = "/home/mocchiut/pamela/data/pamelasimu.root";
  TString outputFile = "EM_output.root";

  //
  // check for command line arguments
  //
  if(argc>1){

    if(!strcmp(argv[1], "-h") || !strcmp(argv[1], "--help") ){
      usage();
      return 0; // exit normally, if help is invoked the user is supposed not to run the program.
    }
```

Solution exercise 1

```
    for (int i = 1; i < argc; i++){
      // ------------------------------------------------------//
      if (!strcmp(argv[i], "--inputFile")){
        if (++i >= argc) return 1;
        inputFile = argv[i];
        cout << "-> input filename set to " << inputFile.Data() << " \n";
        continue;
      }
      if (!strcmp(argv[i], "--outputFile")){
        if (++i >= argc) return 1;
        outputFile = argv[i];
        cout << "-> output filename set to " << outputFile.Data() << " \n";
        continue;
      }
      // ------------------------------------------------------//
      else{
        cout << "WARNING: unidentified input parameter. Ignored. \n";
      }
    }
  }else{
    usage();
    return 0;
  }

  // call the copy function
  copy(inputFile,outputFile);

  return 0;
}
```

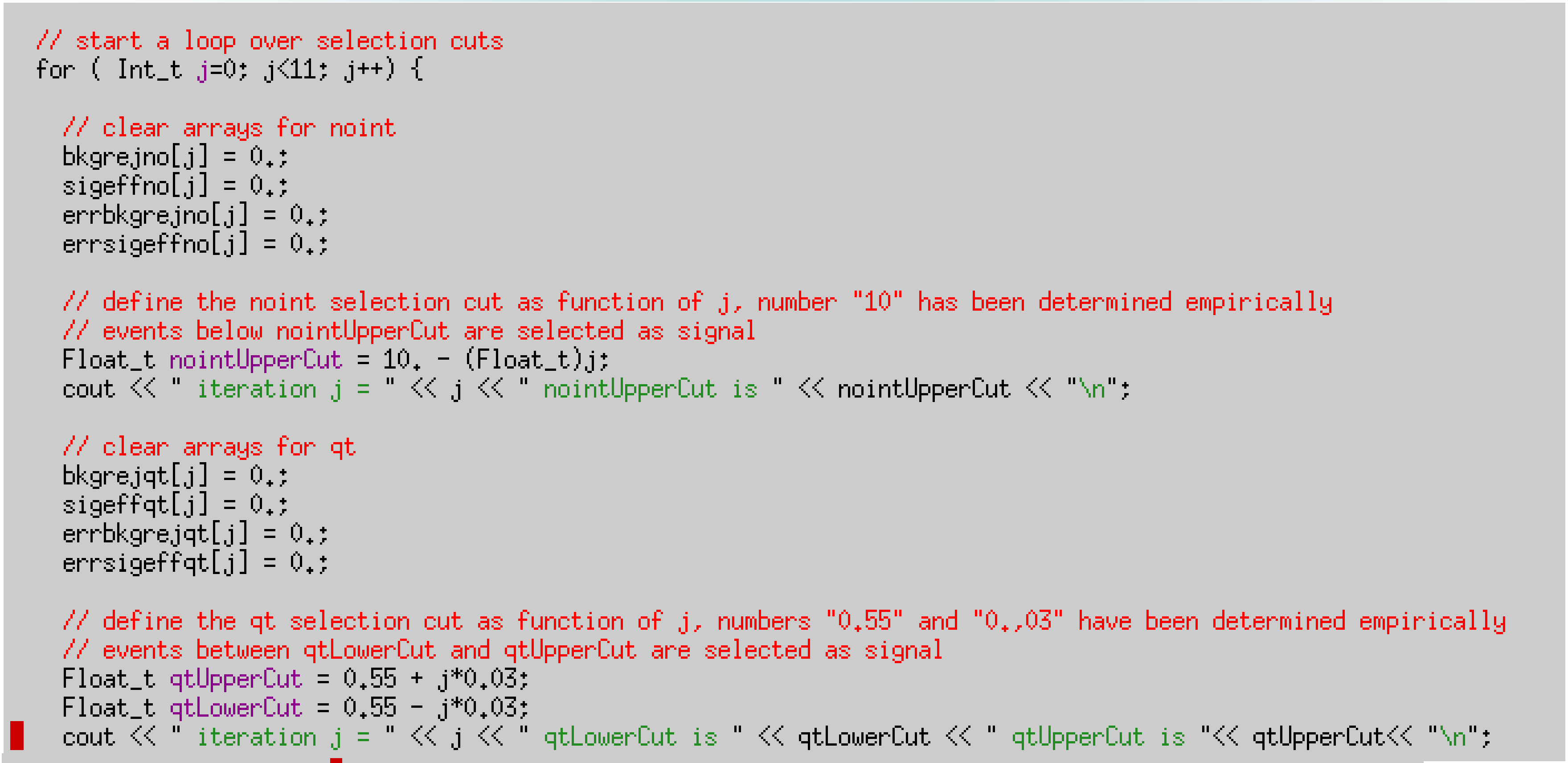
Solution exercise 2

- how to use CINT scripts: slides_8_121130.pdf page 22
- how read a file: slides_8_121130.pdf page 27
- TGraphErrors: slides_9_121207.pdf page 35
- fitting: slides_9_121207.pdf page 42

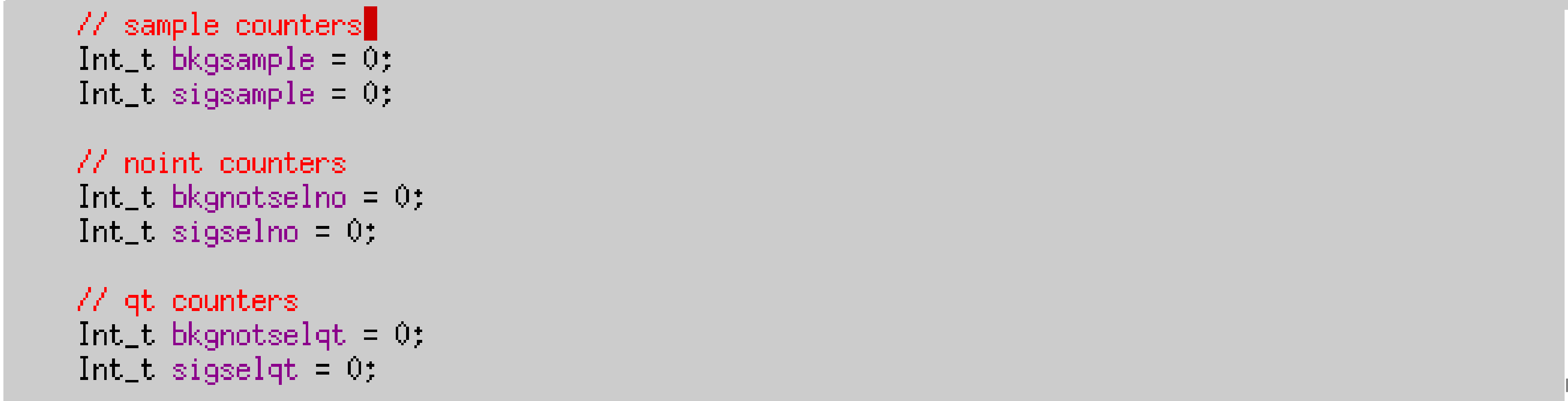
Solution exercise 2

```
//
// Emiliano Mocchiutti, 17/01/2013
// this script read a file with a tree called "selectedtree"
// which contains three variables (pID, noint and qt).
// ROC curve is calculated for noint and qt where background events
// have pID == 0 and signal events have pID == 1
//
#include <iostream>
#include <cmath>
#include <TFile.h>
#include <TTree.h>
#include <PamCalo.h>

void ROC(TString inputFile="prova.root"){

  // Open the input file and set branch addresses
  TFile *file = TFile::Open(inputFile,"READ");
  TTree *tree = (TTree*)file->Get("selectedtree");
  Int_t pID = 0;
  tree->SetBranchAddress("pID",&pID);
  Int_t noint = 0.;
  tree->SetBranchAddress("noint",&noint);
  Float_t qt = 0.;
  tree->SetBranchAddress("qt",&qt);

  // define arryas which will contain the output values for ROC curves
  Float_t bkgrejno[11];
  Float_t sigeffno[11];
  Float_t errbkgrejno[11];
  Float_t errsigeffno[11];

  Float_t bkgrejqt[11];
  Float_t sigeffqt[11];
  Float_t errbkgrejqt[11];
  Float_t errsigeffqt[11];
```

Solution exercise 2



```
// start a loop over selection cuts
for ( Int_t j=0; j<11; j++) {

  // clear arrays for noint
  bkgrejno[j] = 0.;
  sigeffno[j] = 0.;
  errbkgrejno[j] = 0.;
  errsigeffno[j] = 0.;

  // define the noint selection cut as function of j, number "10" has been determined empirically
  // events below nointUpperCut are selected as signal
  Float_t nointUpperCut = 10. - (Float_t)j;
  cout << " iteration j = " << j << " nointUpperCut is " << nointUpperCut << "\n";

  // clear arrays for qt
  bkgrejqt[j] = 0.;
  sigeffqt[j] = 0.;
  errbkgrejqt[j] = 0.;
  errsigeffqt[j] = 0.;

  // define the qt selection cut as function of j, numbers "0.55" and "0.,03" have been determined empirically
  // events between qtLowerCut and qtUpperCut are selected as signal
  Float_t qtUpperCut = 0.55 + j*0.03;
  Float_t qtLowerCut = 0.55 - j*0.03;
  cout << " iteration j = " << j << " qtLowerCut is " << qtLowerCut << " qtUpperCut is "<< qtUpperCut<< "\n";
```



```
  // sample counters
  Int_t bkgsample = 0;
  Int_t sigsample = 0;

  // noint counters
  Int_t bkgnotselno = 0;
  Int_t sigselno = 0;

  // qt counters
  Int_t bkgnotselqt = 0;
  Int_t sigselqt = 0;
```

Solution exercise 2

```
// loop over the events of the file
for ( Int_t i=0; i<tree->GetEntries(); i++) {

  if ( i%10000 == 0 ) cout << " get entry " << i << "\n";
  tree->GetEntry(i);

  // pID == 0 selects background events
  if ( pID == 0 ){
    // background sample events
    bkgsample++;

    // background events that do not pass the selection
    if (noint > nointUpperCut ) bkgnotselno++;
    if (qt > qtUpperCut || qt <qtLowerCut ) bkgnotselqt++;
  }

  // pID == 1 selects signal events
  if ( pID == 1 ){
    // signal sample events
    sigsample++;

    // signal events that DO pass the selection
    if( noint <= nointUpperCut ) sigselno++;
    if( qt <= qtUpperCut && qt >= qtLowerCut ) sigselqt++;
  }
}
```

Solution exercise 2

```
   // noint:

   // calculate background rejection and error (Poissonian error is used as rough approximation)
   bkgrejno[j] = (Float_t)bkgnotselno/(Float_t)bkgsample;
   errbkgrejno[j] = sqrt((Float_t)bkgnotselno)/(Float_t)bkgsample;

   // calculate signal efficiency and error (Poissonian error is used as rough approximation)
   sigeffno[j] = (Float_t)sigselno/(Float_t)sigsample;
   errsigeffno[j] = sqrt((Float_t)sigselno)/(Float_t)sigsample;

   // qt:

   // calculate background rejection and error (Poissonian error is used as rough approximation)
   bkgrejqt[j] = (Float_t)bkgnotselqt/(Float_t)bkgsample;
   errbkgrejqt[j] = sqrt((Float_t)bkgnotselqt)/(Float_t)bkgsample;

   // calculate signal efficiency and error (Poissonian error is used as rough approximation)
   sigeffqt[j] = (Float_t)sigselqt/(Float_t)sigsample;
   errsigeffqt[j] = sqrt((Float_t)sigselqt)/(Float_t)sigsample;

 }
```

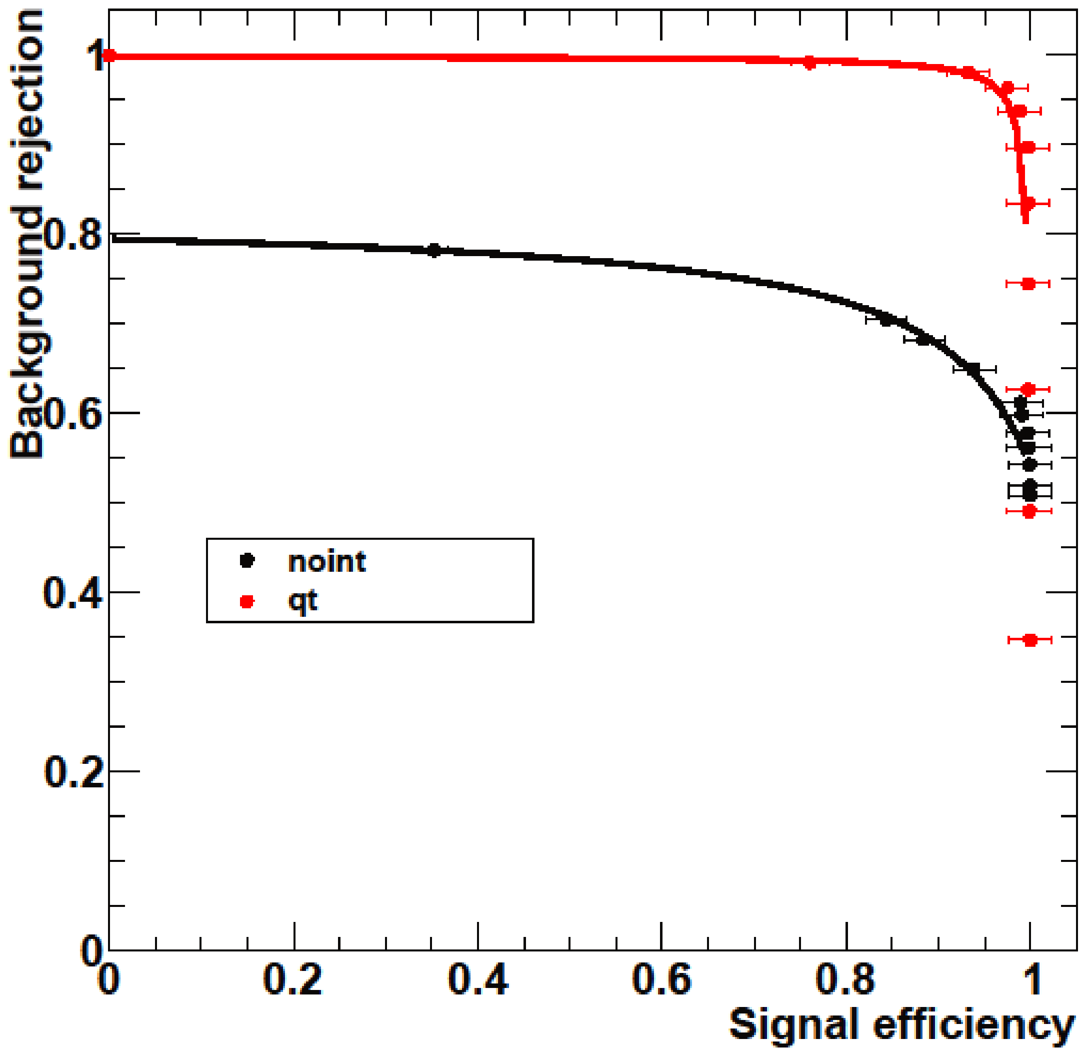
Solution exercise 2

```
// create a new canvas
TCanvas *c = new TCanvas("c","",800,800);
c->SetTicks();
c->Draw();

// create a 2D histogram as a background image
TH2D *histo = new TH2D("histo",";Signal efficiency;Background rejection",1000,0.,1.05,1000,0.,1.05);
histo->SetStats(0);
histo->Draw();

// create the ROC curve for noint
TGraphErrors *ROC = new TGraphErrors(11,sigeffno,bkgrejno,errsigeffno,errbkgrejno);
ROC->SetMarkerStyle(20);
ROC->Draw("PSame");

// create the ROC curve for qt
TGraphErrors *ROCqt = new TGraphErrors(11,sigeffqt,bkgrejqt,errsigeffqt,errbkgrejqt);
ROCqt->SetMarkerStyle(20);
// set red colors
ROCqt->SetMarkerColor(kRed);
ROCqt->SetLineColor(kRed);
ROCqt->Draw("PSame");

// add a legend
TLegend *legend = new TLegend(0.18,0.38,0.45,0.45,"");
legend->SetLineColor(1);
legend->SetFillColor(0);
legend->AddEntry(ROC, "noint", "P");
legend->AddEntry(ROCqt, "qt", "P");
legend->Draw();
```

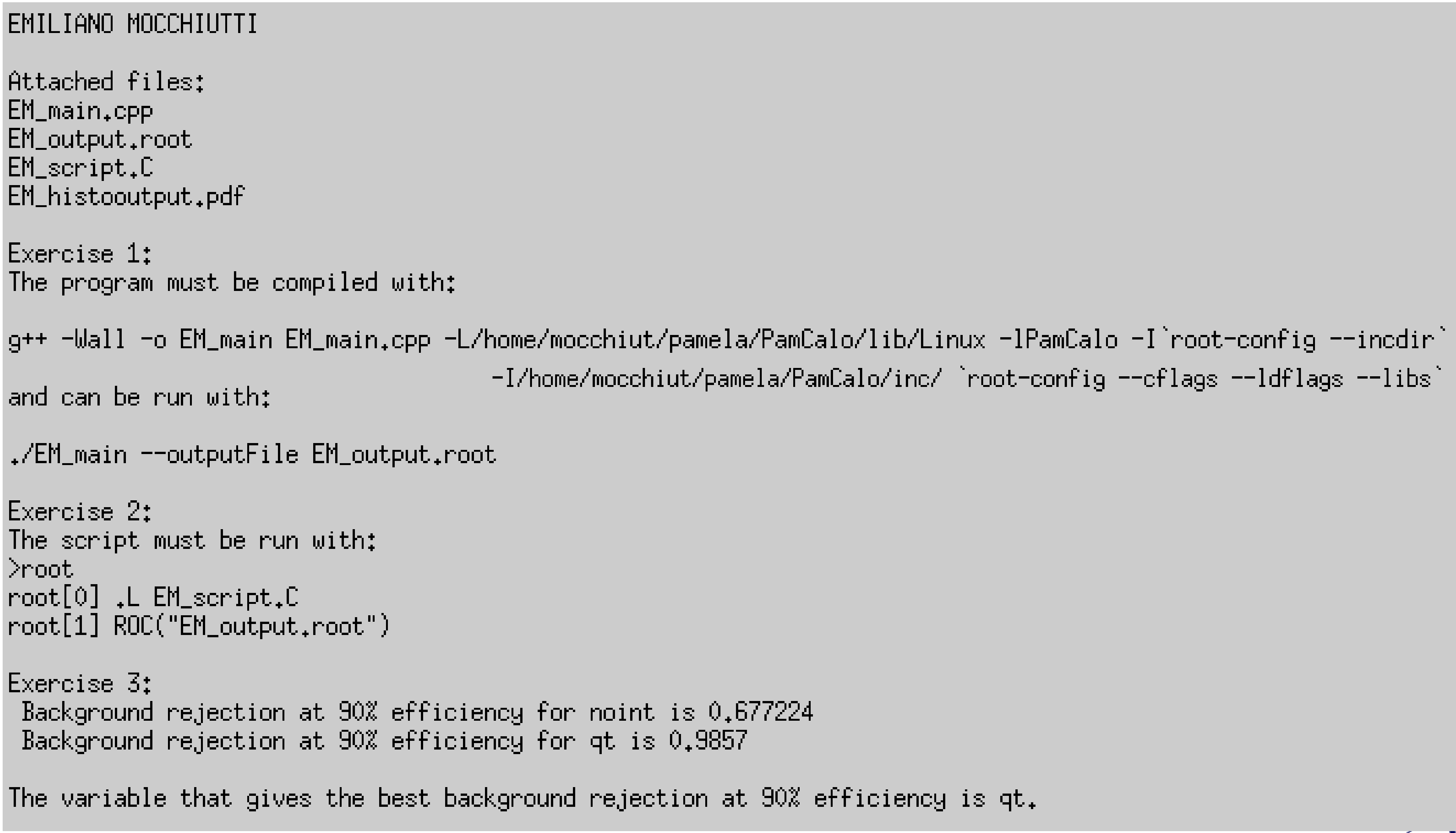
Solution exercise 2-3

```
  // fit the noint ROC
  TF1 *funFit = new TF1("funFit","[0]+[1]/(x-[2])",0.,1.);
  funFit->SetParameters(2.,1.,2.);
  funFit->SetLineColor(kBlack);
  ROC->Fit("funFit","MER","same");
  cout << " Background rejection at 90% efficiency for noint is " << funFit->Eval(0.9) << "\n";

  // fit the qt ROC
  TF1 *funFitQT = new TF1("funFitQT","[0]+[1]/(x-[2])",0.,1.);
  funFitQT->SetParameters(2.,1.,2.);
  funFitQT->SetLineColor(kRed);
  ROCqt->Fit("funFitQT","MER","same");
  cout << " Background rejection at 90% efficiency for qt is " << funFitQT->Eval(0.9) << "\n";

  // save the output pdf file
  c->SaveAs("EM_histooutput.pdf");

}
```

Solution exercise 2-3



Solution exercise, README



```
EMILIANO MOCCHIUTTI

Attached files:
EM_main.cpp
EM_output.root
EM_script.C
EM_histooutput.pdf

Exercise 1:
The program must be compiled with:

g++ -Wall -o EM_main EM_main.cpp -L/home/mocchiut/pamela/PamCalo/lib/Linux -lPamCalo -I`root-config --incdir`
                                     -I/home/mocchiut/pamela/PamCalo/inc/ `root-config --cflags --ldflags --libs`
and can be run with:

./EM_main --outputFile EM_output.root

Exercise 2:
The script must be run with:
>root
root[0] .L EM_script.C
root[1] ROC("EM_output.root")

Exercise 3:
 Background rejection at 90% efficiency for noint is 0.677224
 Background rejection at 90% efficiency for qt is 0.9857

The variable that gives the best background rejection at 90% efficiency is qt.
```