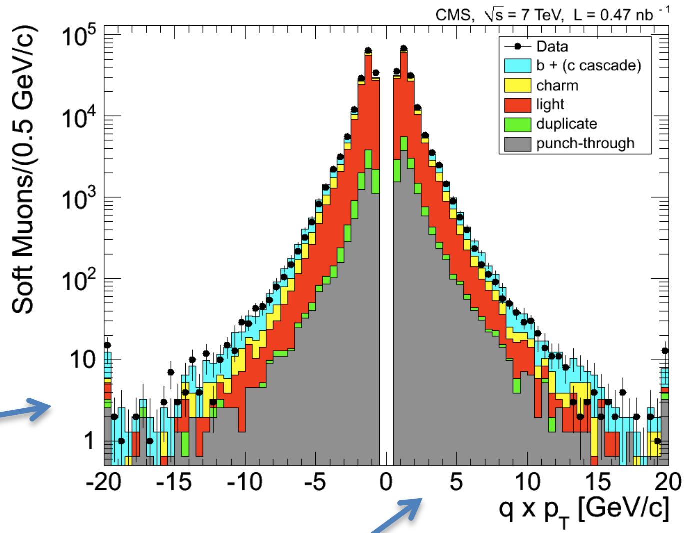
Exercise/Lesson #1

Scientific Data Analysis Lab course

Alexis Pompili - UniBA

The aim of this first exercise



In this exercise we will learn to:

1) execute the main **ROOT** commands,
2) execute a *macro* written in C (within the **ROOT** framework)
3) make a comparison between real & simulated (Monte Carlo) data with an absolute normalization
4) prepare/configure a plot (output of ROOT) with features and quality of a scientific pubblication
5) represent the various simulated components by means of ***stacked plots*** .

The plot that represent our target is the one that appears at page 11 (fig.4) of the CMS paper "*Performance of CMS muon reconstruction in pp collisions at sqrt{s}= 7TeV*" published by ***Journal of Instrumentation (JINST) 7, P10002 (2012).***

I suggest to preliminarily study the pages 6-12 of this CMS paper in order to understand the physical meaning of the following concepts :

- ***Soft Muons, Tight Muons***
- ***Prompt Muons, Muons from Beauty, Muons from Charm, Muons from Light Hadrons***
- ***Fake Muons ("hadronic punch-through"), Duplicates ("Ghost" Muons)***

Files with real and simulated data to begin with …

Create the subdirectory /home/username/Esercitazione-1: `mkdir Esercitazione-1`
Go into this subdirectory and create another one: `mkdir Step0`
In the same sub-dir create another one: `mkdir rootfiles`

In the last subdir I'll copy the following rootfiles (the 1st contains real data, the 2nd simulated):

```
-rw-r-xr-- 1 pompili cms 272774 Nov  6 16:35 Histos_Data_ZeroBias_1aprnew_goodZB_last_OK.root
-rw-r-xr-- 1 pompili cms 322854 Nov  6 16:35 Histos_Mc_MinBias_1aprnew_goodZB_last.root
```

To run ROOT

In the subdir /home/username/Esercitazione-1/Step0/ … I will copy the macro ***main0.C***
Start ROOT from this working subdirectory: `$ root`

```
   ------------------------------------------------------------
  | Welcome to ROOT 6.14/09                http://root.cern.ch |
  |                               (c) 1995-2018, The ROOT Team |
  | Built for linuxx8664gcc                                    |
  | From tag , 22 November 2018                                |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
   ------------------------------------------------------------

root [0]
```

→ ROOT Prompt : you are inside ROOT

Files with real and simulated data to begin with …

Create the subdirectory /home/username/Esercitazione-1: **`mkdir Esercitazione-1`**
Go into this subdirectory and create another one: **`mkdir Step0`**
In the same sub-dir create another one: **`mkdir rootfiles`**

In the last subdir I'll copy the following rootfiles (the 1st contains real data, the 2nd simulated):

-rw-r-xr-- 1 pompili cms 272774 Nov 6 16:35 ***Histos_Data_ZeroBias_1aprnew_goodZB_last_OK.root***
-rw-r-xr-- 1 pompili cms 322854 Nov 6 16:35 ***Histos_Mc_MinBias_1aprnew_goodZB_last.root***

To run ROOT

In the subdir /home/username/Esercitazione-1/Step0/ … I will copy the macro ***main0.C***
Start ROOT from this working subdirectory: $ **`root -l`**

```
  ------------------------------------------------------------
 | Welcome to ROOT 6.14/09                http://root.cern.ch |
 |                               (c) 1995-2018, The ROOT Team |
 | Built for linuxx8664gcc                                    |
 | From tag , 22 November 2018                                |
 | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
  ------------------------------------------------------------

root [0]
```

ROOT Prompt : you are inside ROOT

Opening and inspecting a ROOT file - 1

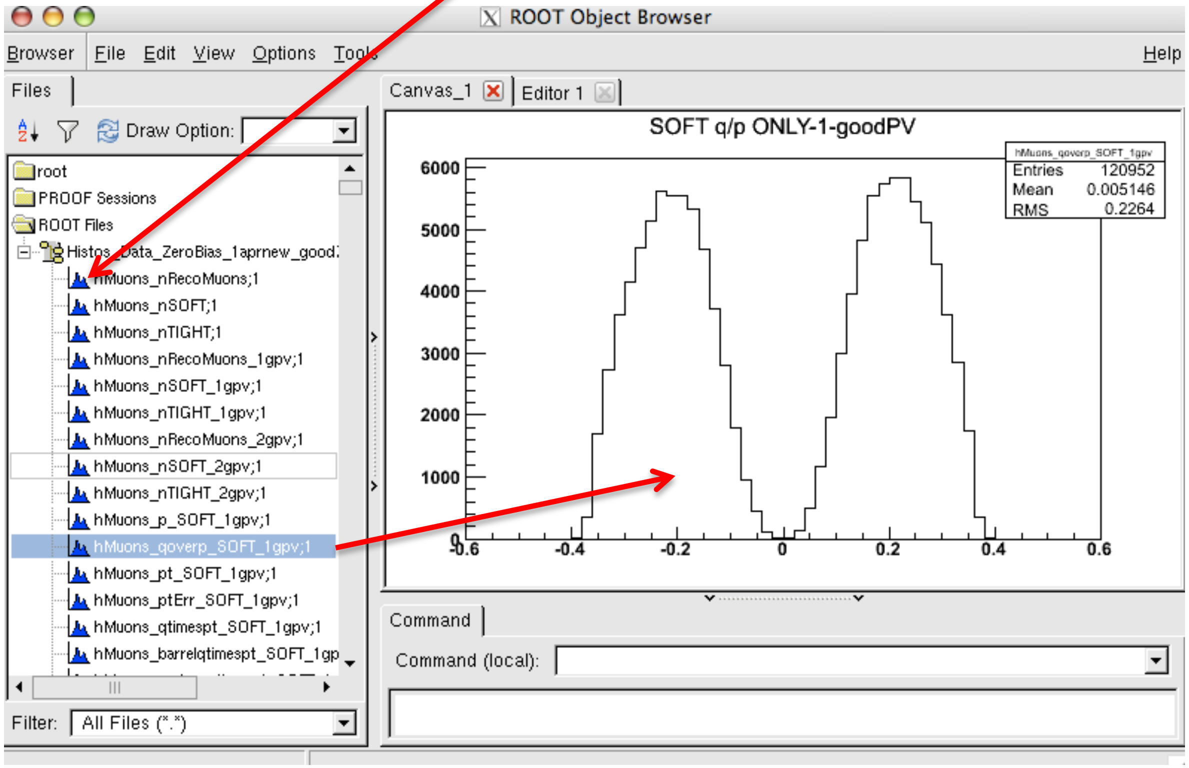
Per aprire la seguente rootupla con ROOT, Histos_Mc_MinBias_1aprnew_goodZB_last.root, fare:

```
$ root -l  Histos_Mc_MinBias_1aprnew_goodZB_last.root
```

Once the ROOT application is opened you have to "call" the "TBrowser" (namely the ROOT **GUI -** ROOT Graphical User Interface) **…** to inspect the file (in this case the file contains only histograms):

```
root [0] TBrowser a
```

This command launches the interactive panel of the GUI:



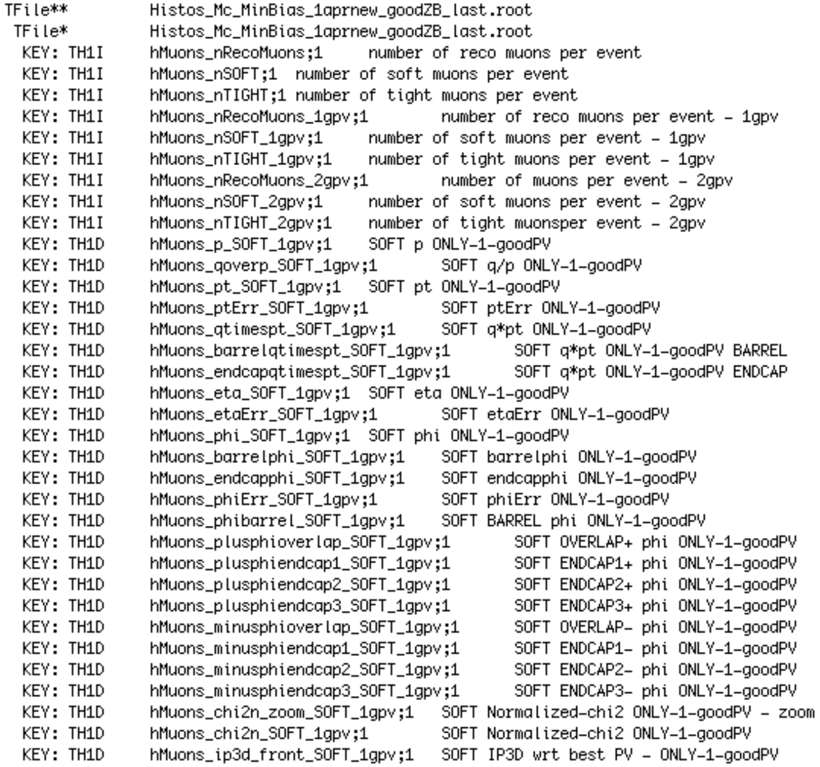
Opening and inspecting a ROOT file - 2

Alternatively:

```
$ root -l
```

```
root[0] Tfile f(“Histos_Mc_MinBias_1aprnew_goodZB_last.root”)
```

Inspect the list of the contained histograms with: root[1] `f->ls()`



```
TFile**         Histos_Mc_MinBias_1aprnew_goodZB_last.root
 TFile*         Histos_Mc_MinBias_1aprnew_goodZB_last.root
  KEY: TH1I     hMuons_nRecoMuons;1      number of reco muons per event
  KEY: TH1I     hMuons_nSOFT;1  number of soft muons per event
  KEY: TH1I     hMuons_nTIGHT;1 number of tight muons per event
  KEY: TH1I     hMuons_nRecoMuons_1gpv;1        number of reco muons per event - 1gpv
  KEY: TH1I     hMuons_nSOFT_1gpv;1      number of soft muons per event - 1gpv
  KEY: TH1I     hMuons_nTIGHT_1gpv;1     number of tight muons per event - 1gpv
  KEY: TH1I     hMuons_nRecoMuons_2gpv;1        number of muons per event - 2gpv
  KEY: TH1I     hMuons_nSOFT_2gpv;1      number of soft muons per event - 2gpv
  KEY: TH1I     hMuons_nTIGHT_2gpv;1     number of tight muonsper event - 2gpv
  KEY: TH1D     hMuons_p_SOFT_1gpv;1     SOFT p ONLY-1-goodPV
  KEY: TH1D     hMuons_qoverp_SOFT_1gpv;1       SOFT q/p ONLY-1-goodPV
  KEY: TH1D     hMuons_pt_SOFT_1gpv;1    SOFT pt ONLY-1-goodPV
  KEY: TH1D     hMuons_ptErr_SOFT_1gpv;1        SOFT ptErr ONLY-1-goodPV
  KEY: TH1D     hMuons_qtimespt_SOFT_1gpv;1     SOFT q*pt ONLY-1-goodPV
  KEY: TH1D     hMuons_barrelqtimespt_SOFT_1gpv;1       SOFT q*pt ONLY-1-goodPV BARREL
  KEY: TH1D     hMuons_endcapqtimespt_SOFT_1gpv;1       SOFT q*pt ONLY-1-goodPV ENDCAP
  KEY: TH1D     hMuons_eta_SOFT_1gpv;1  SOFT eta ONLY-1-goodPV
  KEY: TH1D     hMuons_etaErr_SOFT_1gpv;1       SOFT etaErr ONLY-1-goodPV
  KEY: TH1D     hMuons_phi_SOFT_1gpv;1  SOFT phi ONLY-1-goodPV
  KEY: TH1D     hMuons_barrelphi_SOFT_1gpv;1    SOFT barrelphi ONLY-1-goodPV
  KEY: TH1D     hMuons_endcapphi_SOFT_1gpv;1    SOFT endcapphi ONLY-1-goodPV
  KEY: TH1D     hMuons_phiErr_SOFT_1gpv;1       SOFT phiErr ONLY-1-goodPV
  KEY: TH1D     hMuons_phibarrel_SOFT_1gpv;1    SOFT BARREL phi ONLY-1-goodPV
  KEY: TH1D     hMuons_plusphioverlap_SOFT_1gpv;1       SOFT OVERLAP+ phi ONLY-1-goodPV
  KEY: TH1D     hMuons_plusphiendcap1_SOFT_1gpv;1       SOFT ENDCAP1+ phi ONLY-1-goodPV
  KEY: TH1D     hMuons_plusphiendcap2_SOFT_1gpv;1       SOFT ENDCAP2+ phi ONLY-1-goodPV
  KEY: TH1D     hMuons_plusphiendcap3_SOFT_1gpv;1       SOFT ENDCAP3+ phi ONLY-1-goodPV
  KEY: TH1D     hMuons_minusphioverlap_SOFT_1gpv;1      SOFT OVERLAP- phi ONLY-1-goodPV
  KEY: TH1D     hMuons_minusphiendcap1_SOFT_1gpv;1      SOFT ENDCAP1- phi ONLY-1-goodPV
  KEY: TH1D     hMuons_minusphiendcap2_SOFT_1gpv;1      SOFT ENDCAP2- phi ONLY-1-goodPV
  KEY: TH1D     hMuons_minusphiendcap3_SOFT_1gpv;1      SOFT ENDCAP3- phi ONLY-1-goodPV
  KEY: TH1D     hMuons_chi2n_zoom_SOFT_1gpv;1   SOFT Normalized-chi2 ONLY-1-goodPV - zoom
  KEY: TH1D     hMuons_chi2n_SOFT_1gpv;1        SOFT Normalized-chi2 ONLY-1-goodPV
  KEY: TH1D     hMuons_ip3d_front_SOFT_1gpv;1   SOFT IP3D wrt best PV - ONLY-1-goodPV
```

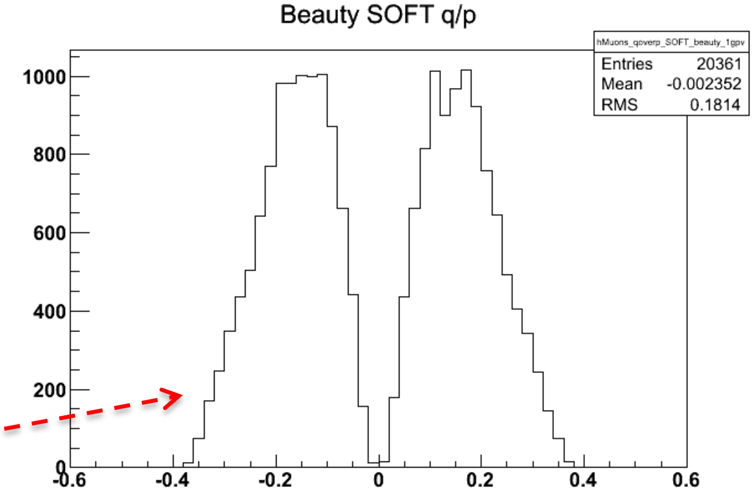
……………………………………

And I can open either the interactive browser

```
root [2] TBrowser a
```

…or look at the single histogram from the command line:

```
root[2] hMuons_qoverp_SOFT_beauty_1gpv->Draw()
```



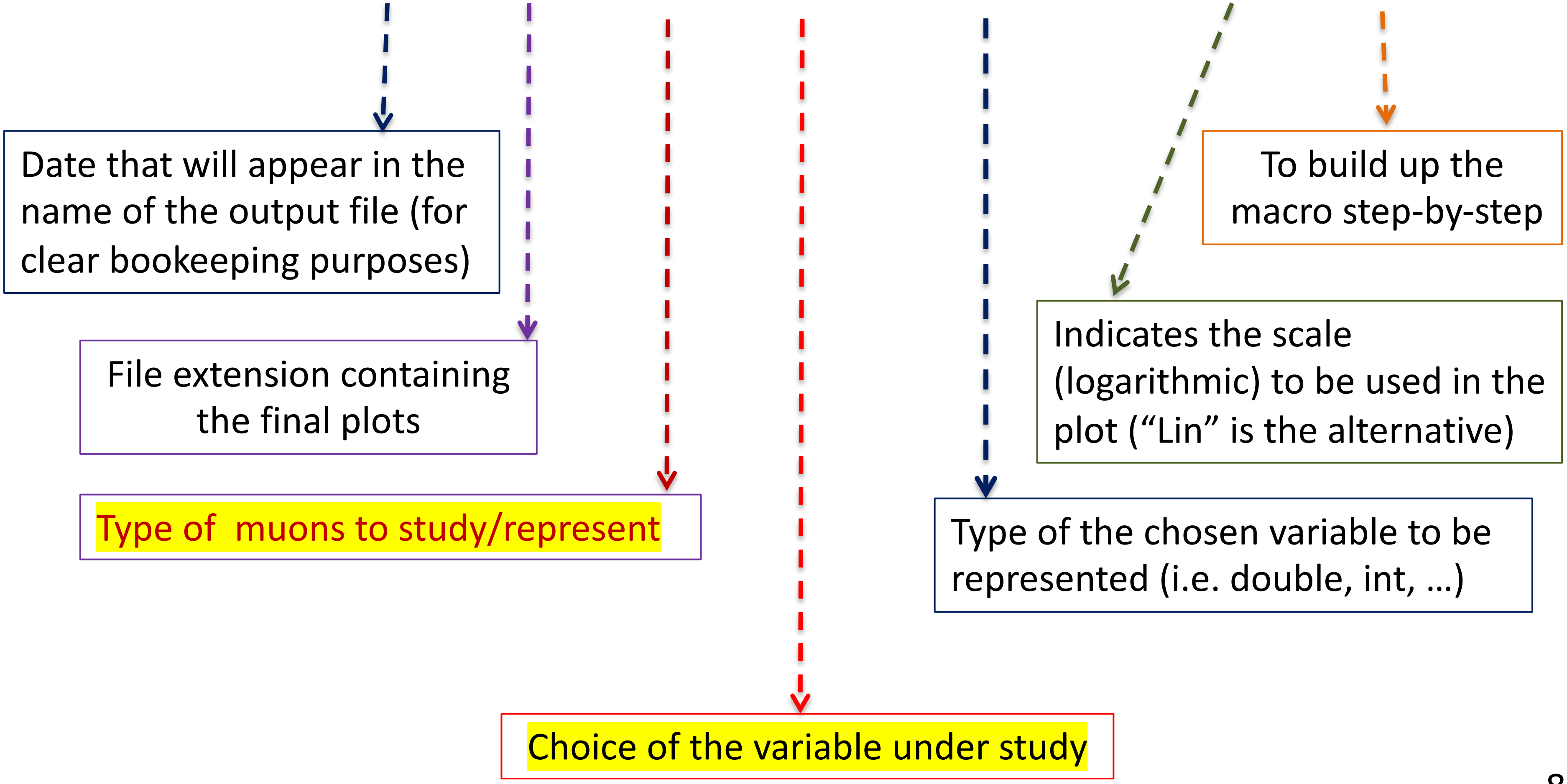
Step 0

STEP-0 : how-to-execute the macro

In the sub-directory ***/home/username/Esercitazione-1/Step0/*** you can find the *macro file* ***main0.C*** to be executed and provide the plots as output.

To execute the macro written in C/C^{++} language, please issue the command:

```
root [0] .x main0.C(”19oct","png","SOFT","qtimespt","doub","1gpv","Log","Step0")
```



Date that will appear in the name of the output file (for clear bookeeping purposes)

File extension containing the final plots

Type of muons to study/represent

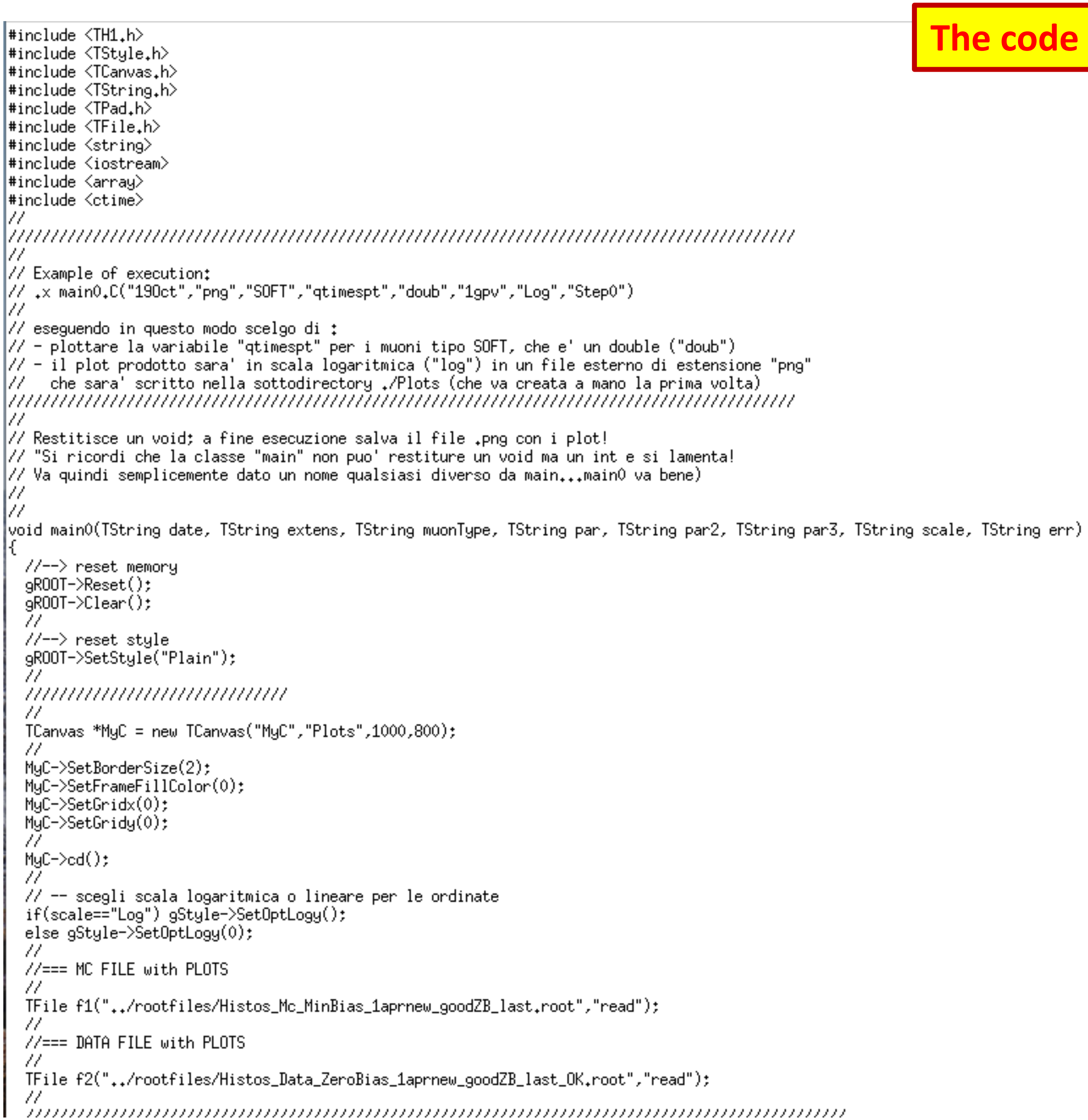
Choice of the variable under study

Type of the chosen variable to be represented (i.e. double, int, …)

Indicates the scale (logarithmic) to be used in the plot (“Lin” is the alternative)

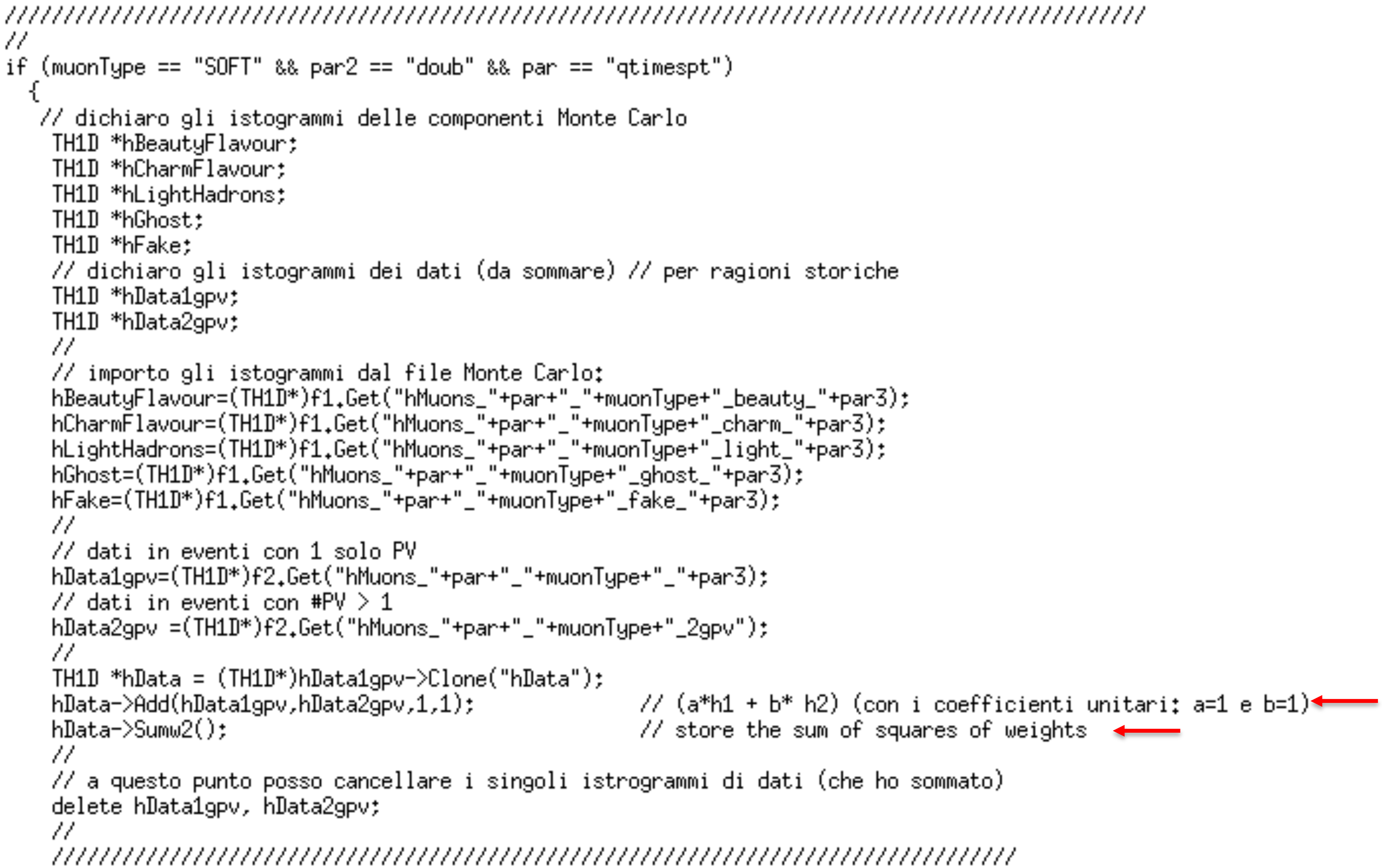
To build up the macro step-by-step

The code in the macro *main0.C* - I



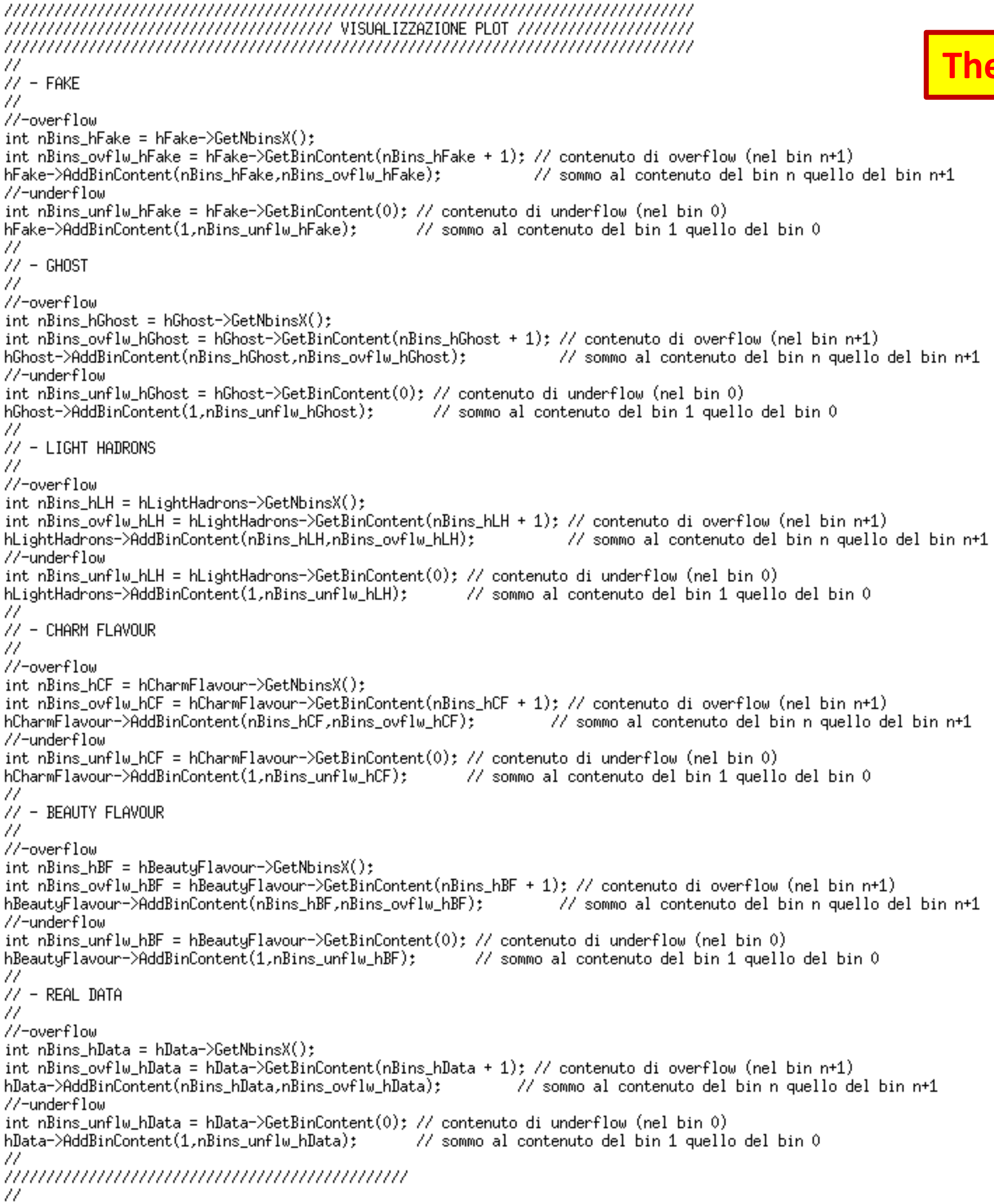
```
#include <TH1.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TString.h>
#include <TPad.h>
#include <TFile.h>
#include <string>
#include <iostream>
#include <array>
#include <ctime>
//
////////////////////////////////////////////////////////////////////////////////////////////
//
// Example of execution:
// .x main0.C("19Oct","png","SOFT","qtimespt","doub","1gpv","Log","Step0")
//
// eseguendo in questo modo scelgo di :
// - plottare la variabile "qtimespt" per i muoni tipo SOFT, che e' un double ("doub")
// - il plot prodotto sara' in scala logaritmica ("log") in un file esterno di estensione "png"
//   che sara' scritto nella sottodirectory ./Plots (che va creata a mano la prima volta)
////////////////////////////////////////////////////////////////////////////////////////////
//
// Restitisce un void; a fine esecuzione salva il file .png con i plot!
// "Si ricordi che la classe "main" non puo' restiture un void ma un int e si lamenta!
// Va quindi semplicemente dato un nome qualsiasi diverso da main...main0 va bene)
//
//
void main0(TString date, TString extens, TString muonType, TString par, TString par2, TString par3, TString scale, TString err)
{
  //--> reset memory
  gROOT->Reset();
  gROOT->Clear();
  //
  //--> reset style
  gROOT->SetStyle("Plain");
  //
  ///////////////////////////////
  //
  TCanvas *MyC = new TCanvas("MyC","Plots",1000,800);
  //
  MyC->SetBorderSize(2);
  MyC->SetFrameFillColor(0);
  MyC->SetGridx(0);
  MyC->SetGridy(0);
  //
  MyC->cd();
  //
  // -- scegli scala logaritmica o lineare per le ordinate
  if(scale=="Log") gStyle->SetOptLogy();
  else gStyle->SetOptLogy(0);
  //
  //=== MC FILE with PLOTS
  //
  TFile f1("../rootfiles/Histos_Mc_MinBias_1aprnew_goodZB_last.root","read");
  //
  //=== DATA FILE with PLOTS
  //
  TFile f2("../rootfiles/Histos_Data_ZeroBias_1aprnew_goodZB_last_OK.root","read");
  //
  ////////////////////////////////////////////////////////////////////////////////////////////////
```

The code in the macro *main0.C* - II



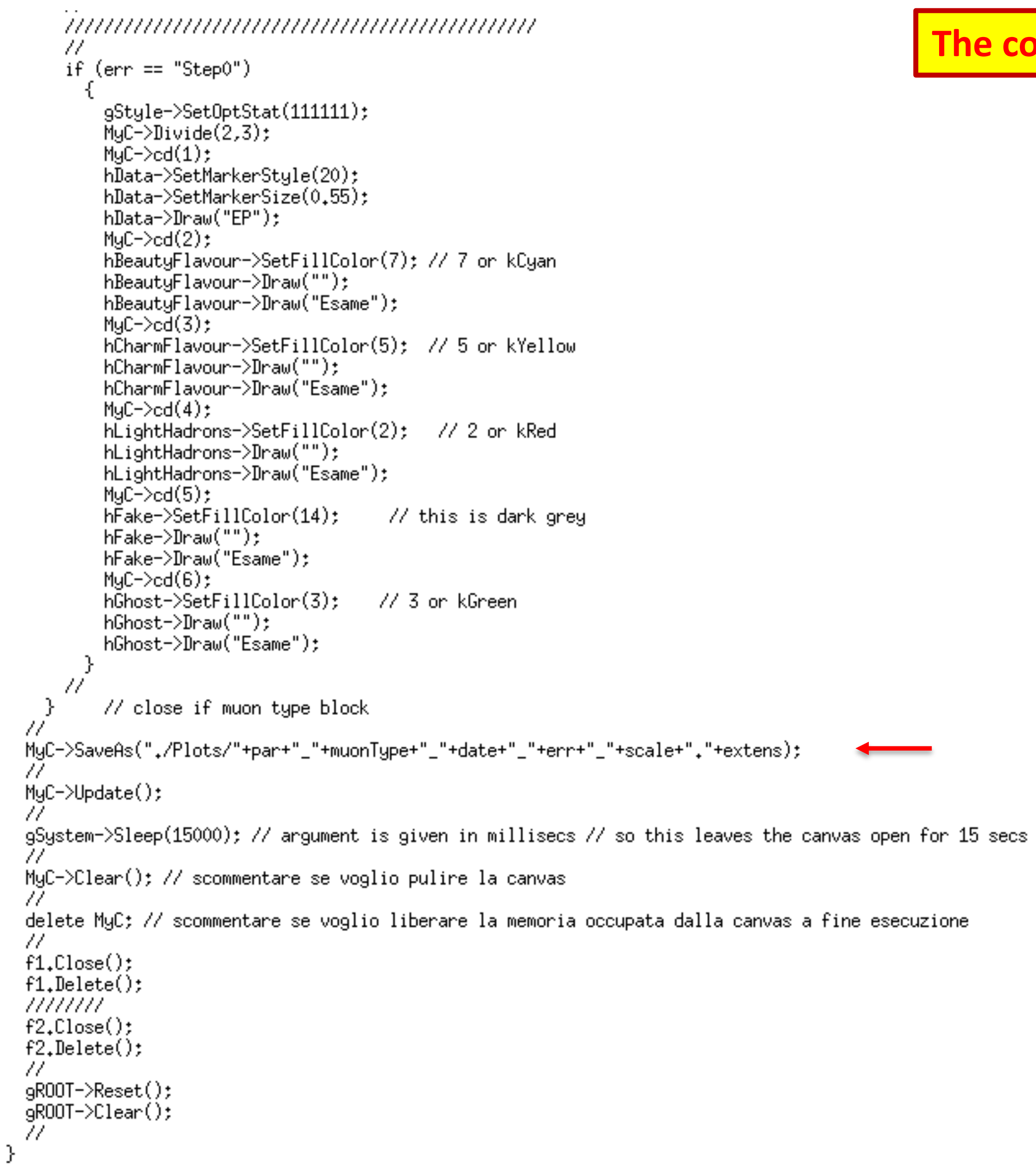
```
//////////////////////////////////////////////////////////////////////////////////////////////////
//
if (muonType == "SOFT" && par2 == "doub" && par == "qtimespt")
  {
   // dichiaro gli istogrammi delle componenti Monte Carlo
    TH1D *hBeautyFlavour;
    TH1D *hCharmFlavour;
    TH1D *hLightHadrons;
    TH1D *hGhost;
    TH1D *hFake;
    // dichiaro gli istogrammi dei dati (da sommare) // per ragioni storiche
    TH1D *hData1gpv;
    TH1D *hData2gpv;
    //
    // importo gli istogrammi dal file Monte Carlo:
    hBeautyFlavour=(TH1D*)f1.Get("hMuons_"+par+"_"+muonType+"_beauty_"+par3);
    hCharmFlavour=(TH1D*)f1.Get("hMuons_"+par+"_"+muonType+"_charm_"+par3);
    hLightHadrons=(TH1D*)f1.Get("hMuons_"+par+"_"+muonType+"_light_"+par3);
    hGhost=(TH1D*)f1.Get("hMuons_"+par+"_"+muonType+"_ghost_"+par3);
    hFake=(TH1D*)f1.Get("hMuons_"+par+"_"+muonType+"_fake_"+par3);
    //
    // dati in eventi con 1 solo PV
    hData1gpv=(TH1D*)f2.Get("hMuons_"+par+"_"+muonType+"_"+par3);
    // dati in eventi con #PV > 1
    hData2gpv =(TH1D*)f2.Get("hMuons_"+par+"_"+muonType+"_2gpv");
    //
    TH1D *hData = (TH1D*)hData1gpv->Clone("hData");
    hData->Add(hData1gpv,hData2gpv,1,1);                // (a*h1 + b* h2) (con i coefficienti unitari: a=1 e b=1)
    hData->Sumw2();                                     // store the sum of squares of weights
    //
    // a questo punto posso cancellare i singoli istrogrammi di dati (che ho sommato)
    delete hData1gpv, hData2gpv;
    //
    ///////////////////////////////////////////////////////////////////////////////////
```

The code in the macro *main0.C* - III



```
//////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////// VISUALIZZAZIONE PLOT /////////////////////
//////////////////////////////////////////////////////////////////////////////////
//
// - FAKE
//
//-overflow
int nBins_hFake = hFake->GetNbinsX();
int nBins_ovflw_hFake = hFake->GetBinContent(nBins_hFake + 1); // contenuto di overflow (nel bin n+1)
hFake->AddBinContent(nBins_hFake,nBins_ovflw_hFake);           // sommo al contenuto del bin n quello del bin n+1
//-underflow
int nBins_unflw_hFake = hFake->GetBinContent(0); // contenuto di underflow (nel bin 0)
hFake->AddBinContent(1,nBins_unflw_hFake);       // sommo al contenuto del bin 1 quello del bin 0
//
// - GHOST
//
//-overflow
int nBins_hGhost = hGhost->GetNbinsX();
int nBins_ovflw_hGhost = hGhost->GetBinContent(nBins_hGhost + 1); // contenuto di overflow (nel bin n+1)
hGhost->AddBinContent(nBins_hGhost,nBins_ovflw_hGhost);           // sommo al contenuto del bin n quello del bin n+1
//-underflow
int nBins_unflw_hGhost = hGhost->GetBinContent(0); // contenuto di underflow (nel bin 0)
hGhost->AddBinContent(1,nBins_unflw_hGhost);       // sommo al contenuto del bin 1 quello del bin 0
//
// - LIGHT HADRONS
//
//-overflow
int nBins_hLH = hLightHadrons->GetNbinsX();
int nBins_ovflw_hLH = hLightHadrons->GetBinContent(nBins_hLH + 1); // contenuto di overflow (nel bin n+1)
hLightHadrons->AddBinContent(nBins_hLH,nBins_ovflw_hLH);           // sommo al contenuto del bin n quello del bin n+1
//-underflow
int nBins_unflw_hLH = hLightHadrons->GetBinContent(0); // contenuto di underflow (nel bin 0)
hLightHadrons->AddBinContent(1,nBins_unflw_hLH);       // sommo al contenuto del bin 1 quello del bin 0
//
// - CHARM FLAVOUR
//
//-overflow
int nBins_hCF = hCharmFlavour->GetNbinsX();
int nBins_ovflw_hCF = hCharmFlavour->GetBinContent(nBins_hCF + 1); // contenuto di overflow (nel bin n+1)
hCharmFlavour->AddBinContent(nBins_hCF,nBins_ovflw_hCF);         // sommo al contenuto del bin n quello del bin n+1
//-underflow
int nBins_unflw_hCF = hCharmFlavour->GetBinContent(0); // contenuto di underflow (nel bin 0)
hCharmFlavour->AddBinContent(1,nBins_unflw_hCF);       // sommo al contenuto del bin 1 quello del bin 0
//
// - BEAUTY FLAVOUR
//
//-overflow
int nBins_hBF = hBeautyFlavour->GetNbinsX();
int nBins_ovflw_hBF = hBeautyFlavour->GetBinContent(nBins_hBF + 1); // contenuto di overflow (nel bin n+1)
hBeautyFlavour->AddBinContent(nBins_hBF,nBins_ovflw_hBF);         // sommo al contenuto del bin n quello del bin n+1
//-underflow
int nBins_unflw_hBF = hBeautyFlavour->GetBinContent(0); // contenuto di underflow (nel bin 0)
hBeautyFlavour->AddBinContent(1,nBins_unflw_hBF);       // sommo al contenuto del bin 1 quello del bin 0
//
// - REAL DATA
//
//-overflow
int nBins_hData = hData->GetNbinsX();
int nBins_ovflw_hData = hData->GetBinContent(nBins_hData + 1); // contenuto di overflow (nel bin n+1)
hData->AddBinContent(nBins_hData,nBins_ovflw_hData);         // sommo al contenuto del bin n quello del bin n+1
//-underflow
int nBins_unflw_hData = hData->GetBinContent(0); // contenuto di underflow (nel bin 0)
hData->AddBinContent(1,nBins_unflw_hData);       // sommo al contenuto del bin 1 quello del bin 0
//
////////////////////////////////////////////////
//
```

The code in the macro *main0.C* - IV

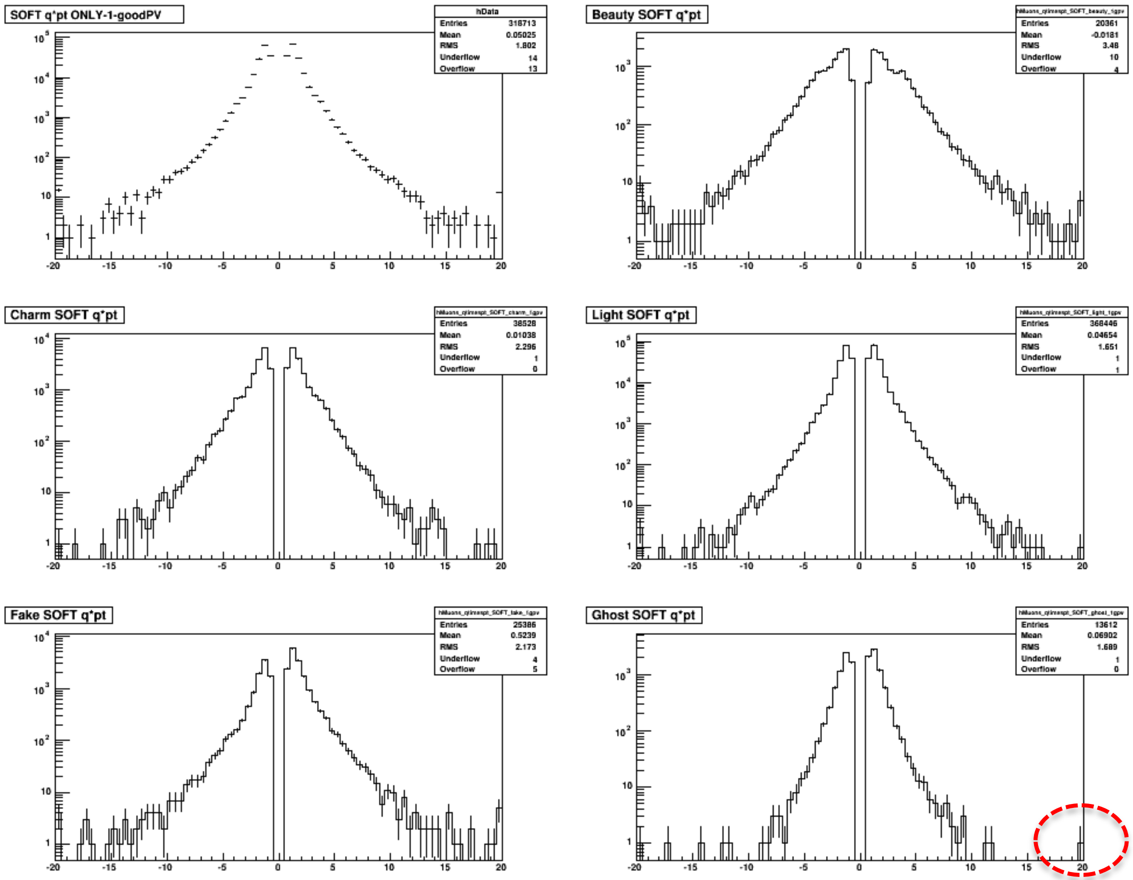


```
      ..
      ////////////////////////////////////////////////
      //
      if (err == "Step0")
        {
          gStyle->SetOptStat(111111);
          MyC->Divide(2,3);
          MyC->cd(1);
          hData->SetMarkerStyle(20);
          hData->SetMarkerSize(0.55);
          hData->Draw("EP");
          MyC->cd(2);
          hBeautyFlavour->SetFillColor(7); // 7 or kCyan
          hBeautyFlavour->Draw("");
          hBeautyFlavour->Draw("Esame");
          MyC->cd(3);
          hCharmFlavour->SetFillColor(5);  // 5 or kYellow
          hCharmFlavour->Draw("");
          hCharmFlavour->Draw("Esame");
          MyC->cd(4);
          hLightHadrons->SetFillColor(2);   // 2 or kRed
          hLightHadrons->Draw("");
          hLightHadrons->Draw("Esame");
          MyC->cd(5);
          hFake->SetFillColor(14);     // this is dark grey
          hFake->Draw("");
          hFake->Draw("Esame");
          MyC->cd(6);
          hGhost->SetFillColor(3);    // 3 or kGreen
          hGhost->Draw("");
          hGhost->Draw("Esame");
        }
      //
    }     // close if muon type block
  //
  MyC->SaveAs("./Plots/"+par+"_"+muonType+"_"+date+"_"+err+"_"+scale+"."+extens);
  //
  MyC->Update();
  //
  gSystem->Sleep(15000); // argument is given in millisecs // so this leaves the canvas open for 15 secs
  //
  MyC->Clear(); // scommentare se voglio pulire la canvas
  //
  delete MyC; // scommentare se voglio liberare la memoria occupata dalla canvas a fine esecuzione
  //
  f1.Close();
  f1.Delete();
  ////////
  f2.Close();
  f2.Delete();
  //
  gROOT->Reset();
  gROOT->Clear();
  //
}
```

STEP-0 : simple histograms visualization

Once the *macro* is executed, we can visualize the plot directly (on screen) or stored in the output file (in sub-dir Step0/Plots) that can be scrutinized by the command:

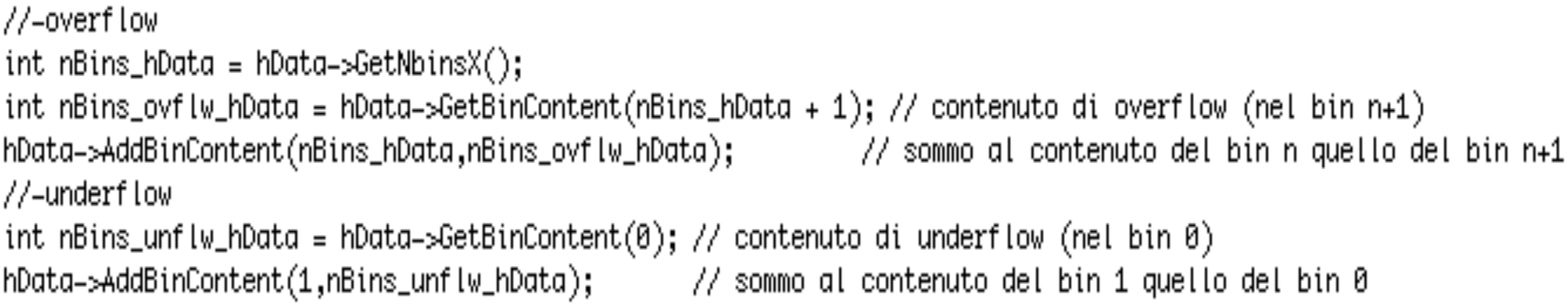
```
$ display ./Plots/qtimespt_SOFT_19oct_Step0_Log.png
```



Note: the **overflows/underflows** can be appreciated typically when using the LOG scale

STEP-0 : underflows & overflows in histograms

The visualization of the overflow/underflow in the histograms it is not a **ROOT** default. To obtain this feature the following code lines were added and inserted for each histogram (see above - slide 11):



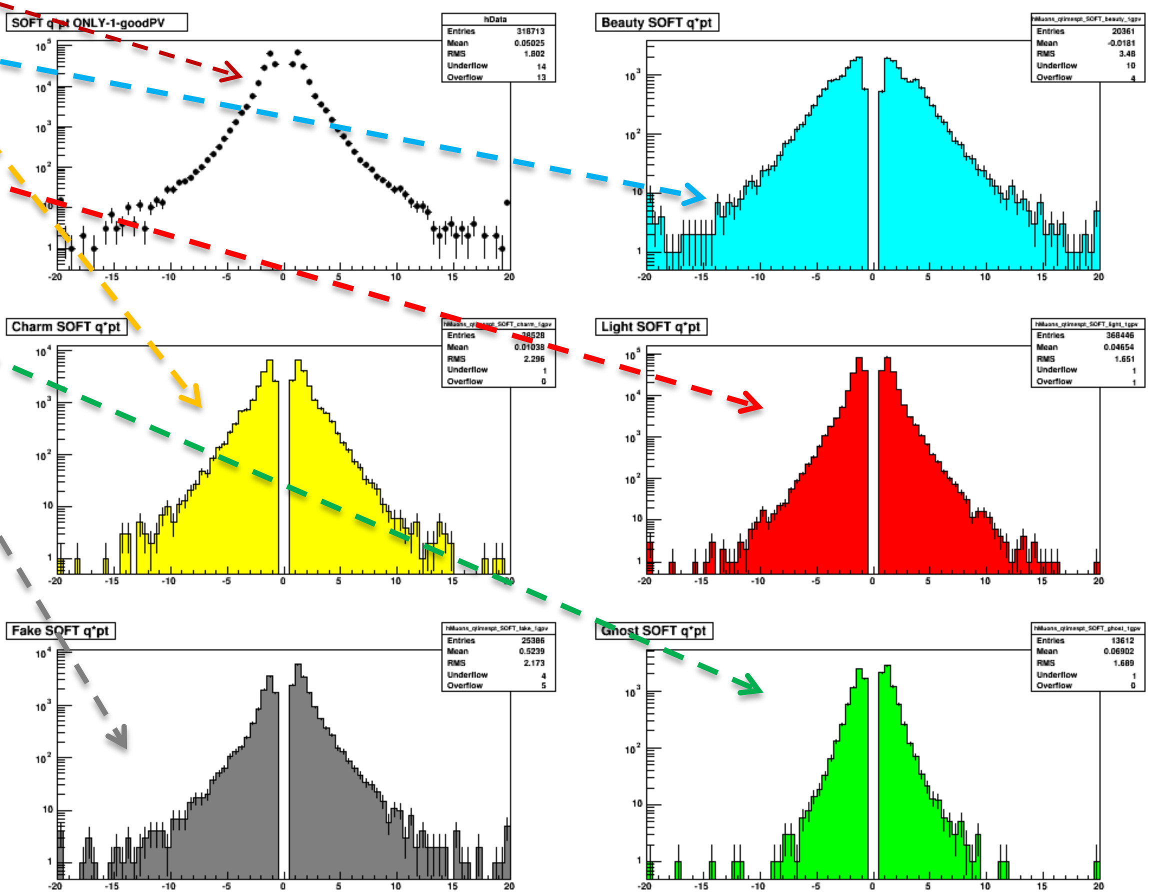
```
//-overflow
int nBins_hData = hData->GetNbinsX();
int nBins_ovflw_hData = hData->GetBinContent(nBins_hData + 1); // contenuto di overflow (nel bin n+1)
hData->AddBinContent(nBins_hData,nBins_ovflw_hData);          // sommo al contenuto del bin n quello del bin n+1
//-underflow
int nBins_unflw_hData = hData->GetBinContent(0); // contenuto di underflow (nel bin 0)
hData->AddBinContent(1,nBins_unflw_hData);       // sommo al contenuto del bin 1 quello del bin 0
```

STEP-0 : options for histograms' visualization

To make appear the data entries as small black circles instead of crosses we need to add: **hData->SetMarkerStyle(20);** to properly set their size: **hData->SetMarkerSize(0.55);**

To provide filling colours to the histograms for the MC components :

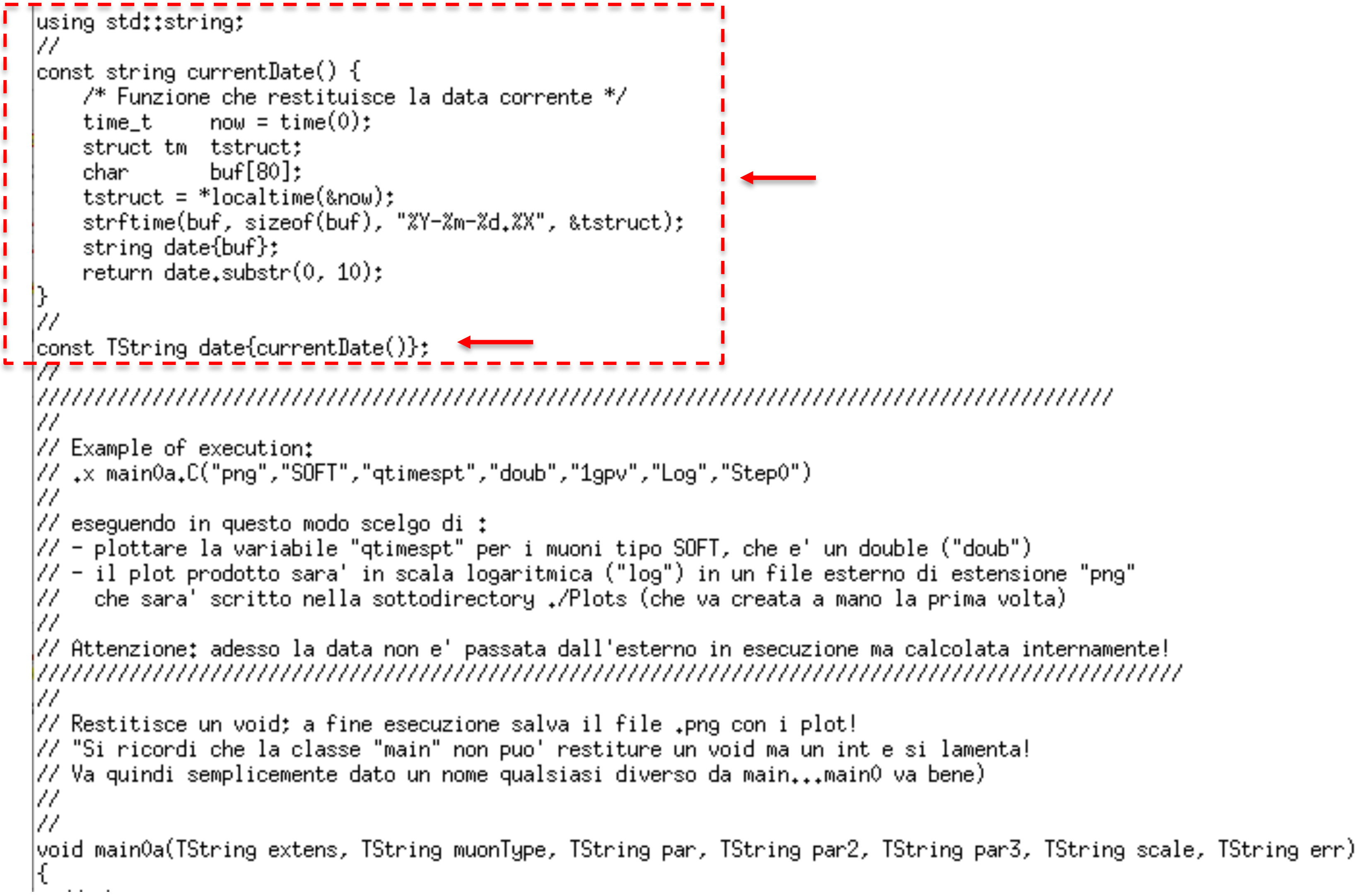
hBeautyFlavour->SetFillColor(7);
hCharmFlavour->SetFillColor(5);
hLightHadrons->SetFillColor(2);
hFake->SetFillColor(14);
hGhost->SetFillColor(3);



Step 0a

Getting date from the machine instead of passing it as parameter…

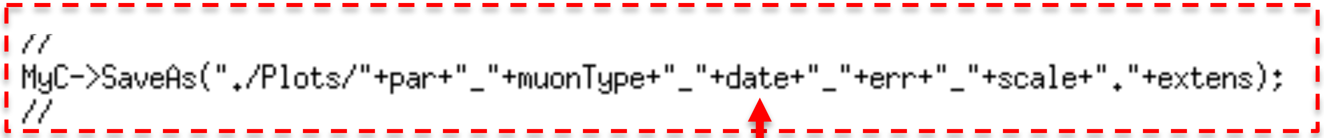
Have a look at a modified macro now called **main0a.C** ;
date is not passed anymore from outside through the interface (now I have 1 parameter less) but is defined inside the macro and taken from the machine clock.



```
using std::string;
//
const string currentDate() {
    /* Funzione che restituisce la data corrente */
    time_t     now = time(0);
    struct tm  tstruct;
    char       buf[80];
    tstruct = *localtime(&now);
    strftime(buf, sizeof(buf), "%Y-%m-%d.%X", &tstruct);
    string date{buf};
    return date.substr(0, 10);
}
//
const TString date{currentDate()};
//
/////////////////////////////////////////////////////////////////////////////////////////////
//
// Example of execution:
// .x main0a.C("png","SOFT","qtimespt","doub","1gpv","Log","Step0")
//
// eseguendo in questo modo scelgo di :
// - plottare la variabile "qtimespt" per i muoni tipo SOFT, che e' un double ("doub")
// - il plot prodotto sara' in scala logaritmica ("log") in un file esterno di estensione "png"
//   che sara' scritto nella sottodirectory ./Plots (che va creata a mano la prima volta)
//
// Attenzione: adesso la data non e' passata dall'esterno in esecuzione ma calcolata internamente!
///////////////////////////////////////////////////////////////////////////////////////////////////
//
// Restitisce un void; a fine esecuzione salva il file .png con i plot!
// "Si ricordi che la classe "main" non puo' restiture un void ma un int e si lamenta!
// Va quindi semplicemente dato un nome qualsiasi diverso da main...main0 va bene)
//
//
void main0a(TString extens, TString muonType, TString par, TString par2, TString par3, TString scale, TString err)
{
```

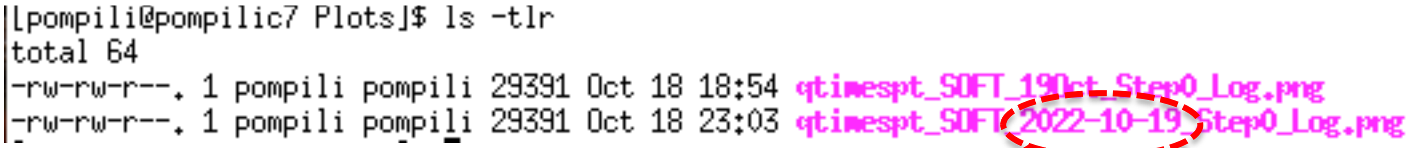
Getting date from the machine instead of passing it as parameter…

The *const Tstring date* is now used in the name of the output file



```
//
MyC->SaveAs("./Plots/"+par+"_"+muonType+"_"+date+"_"+err+"_"+scale+"."+extens);
//
```

As a result, there is a new output file in the subdir Plots:

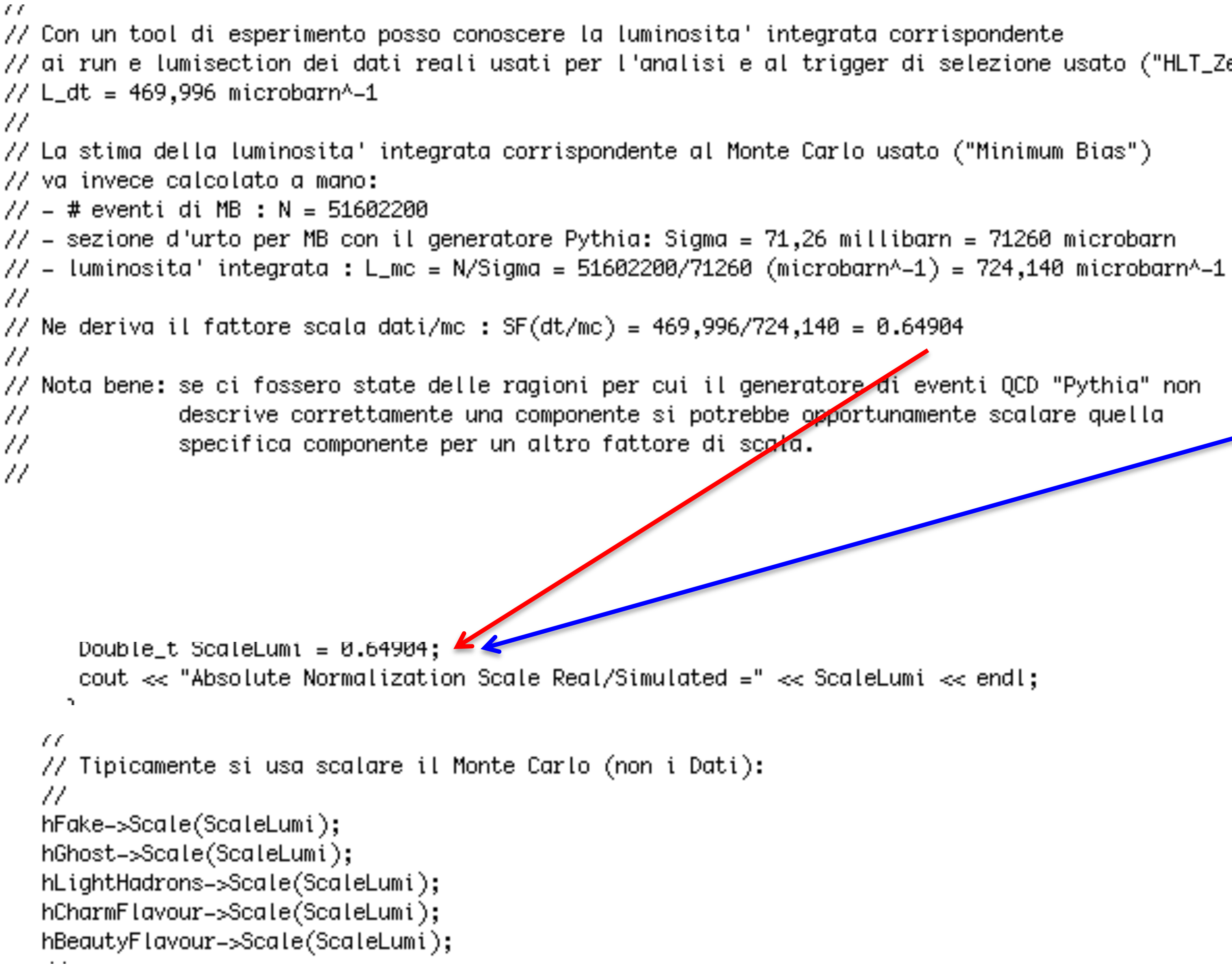


```
[pompili@pompilic7 Plots]$ ls -tlr
total 64
-rw-rw-r--. 1 pompili pompili 29391 Oct 18 18:54 qtimespt_SOFT_19Oct_Step0_Log.png
-rw-rw-r--. 1 pompili pompili 29391 Oct 18 23:03 qtimespt_SOFT_2022-10-19_Step0_Log.png
```

… so the ***const TString*** has value: 2022-10-19

Step 1

STEP-1 : absolute normalization of MC to Data – 1



```
//
// Con un tool di esperimento posso conoscere la luminosita' integrata corrispondente
// ai run e lumisection dei dati reali usati per l'analisi e al trigger di selezione usato ("HLT_ZeroBias"):
// L_dt = 469,996 microbarn^-1
//
// La stima della luminosita' integrata corrispondente al Monte Carlo usato ("Minimum Bias")
// va invece calcolato a mano:
// - # eventi di MB : N = 51602200
// - sezione d'urto per MB con il generatore Pythia: Sigma = 71,26 millibarn = 71260 microbarn
// - luminosita' integrata : L_mc = N/Sigma = 51602200/71260 (microbarn^-1) = 724,140 microbarn^-1
//
// Ne deriva il fattore scala dati/mc : SF(dt/mc) = 469,996/724,140 = 0.64904
//
// Nota bene: se ci fossero state delle ragioni per cui il generatore di eventi QCD "Pythia" non
//            descrive correttamente una componente si potrebbe opportunamente scalare quella
//            specifica componente per un altro fattore di scala.
//
```

```
    Double_t ScaleLumi = 0.64904;
    cout << "Absolute Normalization Scale Real/Simulated =" << ScaleLumi << endl;
  }

  //
  // Tipicamente si usa scalare il Monte Carlo (non i Dati):
  //
  hFake->Scale(ScaleLumi);
  hGhost->Scale(ScaleLumi);
  hLightHadrons->Scale(ScaleLumi);
  hCharmFlavour->Scale(ScaleLumi);
  hBeautyFlavour->Scale(ScaleLumi);
```

L_{int}^{DATA}

$$L_{\text{int}}^{MC} = \frac{N_{evt}^{GEN}}{\sigma_{Pythia}^{MinBias}}$$

$$f_{SCALE} = \frac{L_{int}^{MC}}{L_{\text{int}}^{DATA}}$$

STEP-1 : absolute normalization of MC to Data – 2

Create the sub-directory ***/home/username/Esercitazione-1/Step1/*** ,
where now I copy the *macro* ***main1.C***, ***while*** you create the sub-dir ***Plots***,
Where the macro at execution will write the plots' file.

Now the macro is executed with :

```
root [0] .x main1.C("png","SOFT","qtimespt","doub","1gpv","Log","Step1”)
```

Once the *macro* has been executed you can visualize the plots:

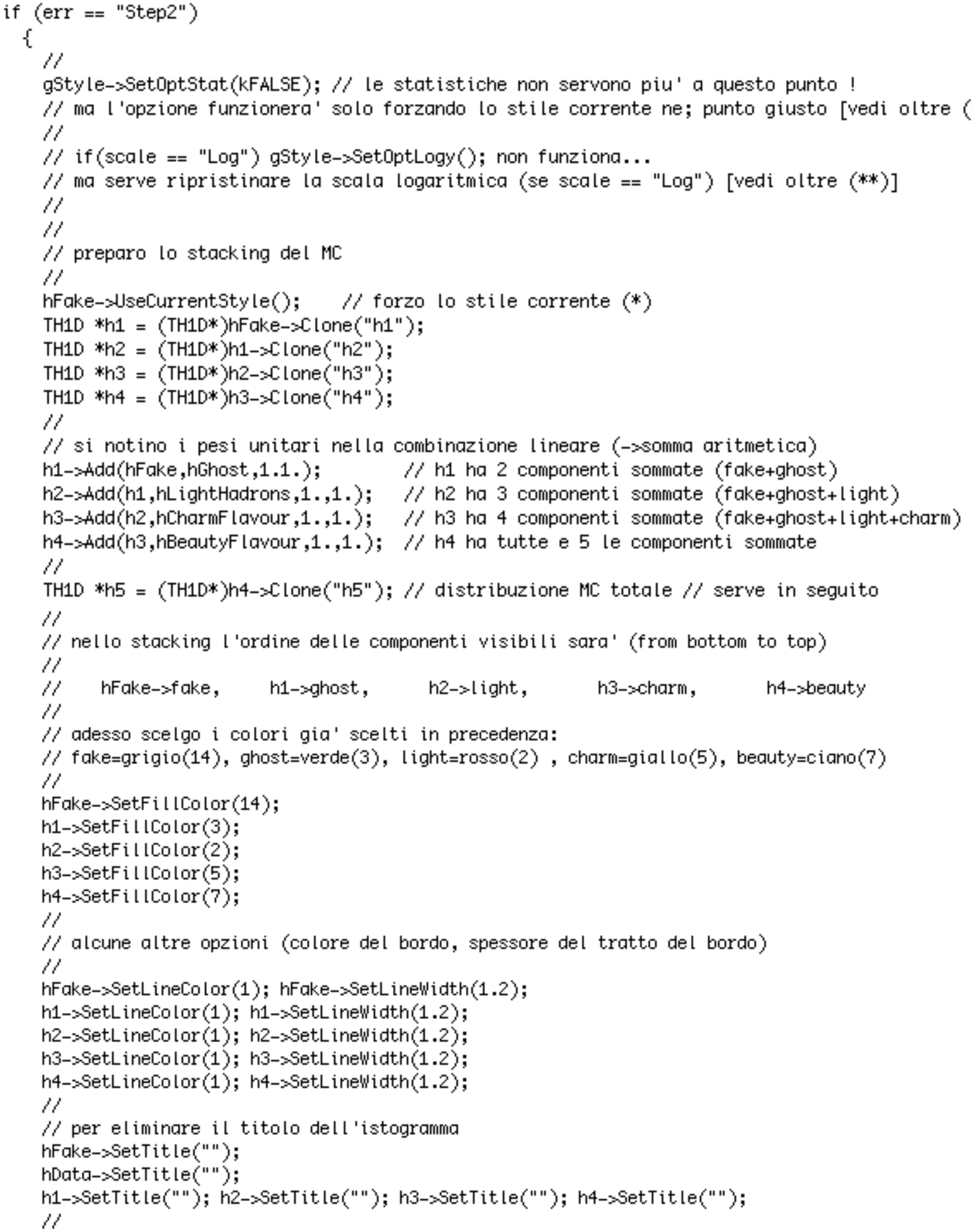
```
$ display ./Plots/qtimespt_SOFT_7nov_Step1_Log_scaleLumi.png
```

You can appreciate that the 5 simulated distributions are effectively scaled (note the change of scale in the y-axis with respect to the previous plot). Note that the number of *entries* has remined the same in the statistics box !! (this may generate some confusion … but it is enough to check the overflows/underflows that are no more integers in order to be sure of the scaling).

EXERCISE for home : try to apply a **relative normalization (to obtain a shape comparison)**

Step 2

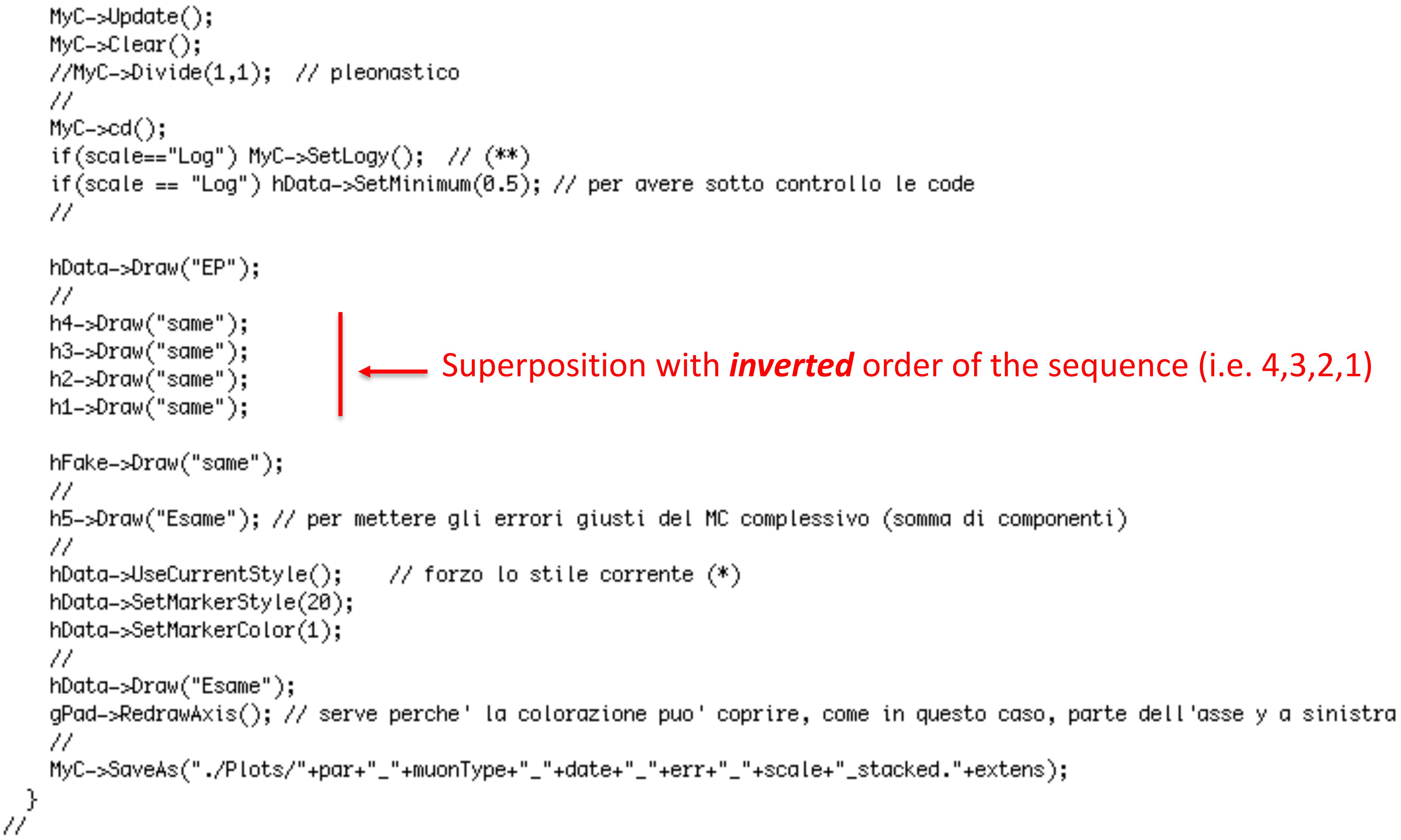
STEP-2 : histograms' *stacking* - 1



```
if (err == "Step2")
  {
    //
    gStyle->SetOptStat(kFALSE); // le statistiche non servono piu' a questo punto !
    // ma l'opzione funzionera' solo forzando lo stile corrente ne; punto giusto [vedi oltre (*)]
    //
    // if(scale == "Log") gStyle->SetOptLogy(); non funziona...
    // ma serve ripristinare la scala logaritmica (se scale == "Log") [vedi oltre (**)]
    //
    //
    // preparo lo stacking del MC
    //
    hFake->UseCurrentStyle();    // forzo lo stile corrente (*)
    TH1D *h1 = (TH1D*)hFake->Clone("h1");
    TH1D *h2 = (TH1D*)h1->Clone("h2");
    TH1D *h3 = (TH1D*)h2->Clone("h3");
    TH1D *h4 = (TH1D*)h3->Clone("h4");
    //
    // si notino i pesi unitari nella combinazione lineare (->somma aritmetica)
    h1->Add(hFake,hGhost,1.1.);        // h1 ha 2 componenti sommate (fake+ghost)
    h2->Add(h1,hLightHadrons,1.,1.);   // h2 ha 3 componenti sommate (fake+ghost+light)
    h3->Add(h2,hCharmFlavour,1.,1.);   // h3 ha 4 componenti sommate (fake+ghost+light+charm)
    h4->Add(h3,hBeautyFlavour,1.,1.);  // h4 ha tutte e 5 le componenti sommate
    //
    TH1D *h5 = (TH1D*)h4->Clone("h5"); // distribuzione MC totale // serve in seguito
    //
    // nello stacking l'ordine delle componenti visibili sara' (from bottom to top)
    //
    //     hFake->fake,     h1->ghost,      h2->light,       h3->charm,       h4->beauty
    //
    // adesso scelgo i colori gia' scelti in precedenza:
    // fake=grigio(14), ghost=verde(3), light=rosso(2) , charm=giallo(5), beauty=ciano(7)
    //
    hFake->SetFillColor(14);
    h1->SetFillColor(3);
    h2->SetFillColor(2);
    h3->SetFillColor(5);
    h4->SetFillColor(7);
    //
    // alcune altre opzioni (colore del bordo, spessore del tratto del bordo)
    //
    hFake->SetLineColor(1); hFake->SetLineWidth(1.2);
    h1->SetLineColor(1); h1->SetLineWidth(1.2);
    h2->SetLineColor(1); h2->SetLineWidth(1.2);
    h3->SetLineColor(1); h3->SetLineWidth(1.2);
    h4->SetLineColor(1); h4->SetLineWidth(1.2);
    //
    // per eliminare il titolo dell'istogramma
    hFake->SetTitle("");
    hData->SetTitle("");
    h1->SetTitle(""); h2->SetTitle(""); h3->SetTitle(""); h4->SetTitle("");
    //
```

histograms as ***partial*** sums
in a sequence (1,2,3,4)

STEP-2 : histograms' *stacking* - 2



```
    MyC->Update();
    MyC->Clear();
    //MyC->Divide(1,1);  // pleonastico
    //
    MyC->cd();
    if(scale=="Log") MyC->SetLogy();  // (**)
    if(scale == "Log") hData->SetMinimum(0.5); // per avere sotto controllo le code
    //

    hData->Draw("EP");
    //
    h4->Draw("same");
    h3->Draw("same");
    h2->Draw("same");
    h1->Draw("same");

    hFake->Draw("same");
    //
    h5->Draw("Esame"); // per mettere gli errori giusti del MC complessivo (somma di componenti)
    //
    hData->UseCurrentStyle();    // forzo lo stile corrente (*)
    hData->SetMarkerStyle(20);
    hData->SetMarkerColor(1);
    //
    hData->Draw("Esame");
    gPad->RedrawAxis(); // serve perche' la colorazione puo' coprire, come in questo caso, parte dell'asse y a sinistra
    //
    MyC->SaveAs("./Plots/"+par+"_"+muonType+"_"+date+"_"+err+"_"+scale+"_stacked."+extens);
  }
//
```

Superposition with ***inverted*** order of the sequence (i.e. 4,3,2,1)

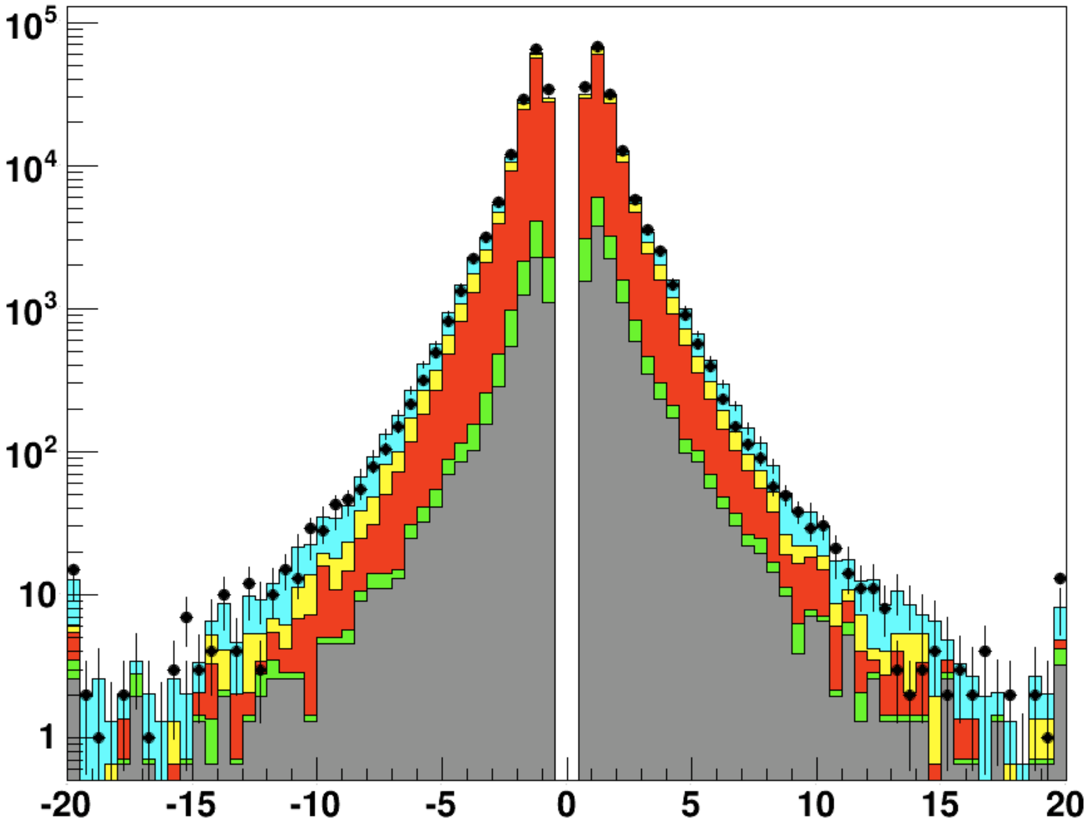
STEP-2 : histograms' *stacking* - 3

Create the sub-dir ***/home/username/esercitazione-1/step2/*** ,
where the *macro* ***main2.C*** is copied, while you create the sub-dir ***Plots.***
To execute the macro:

```
root [0] .x main2.C("png","SOFT","qtimespt","doub","1gpv","Log","Step2”)
```

To visualize the output:

```
$ display ./Plots/qtimespt_SOFT_2022-10-20_Step2_Log_stacked.png
```



Step 2a

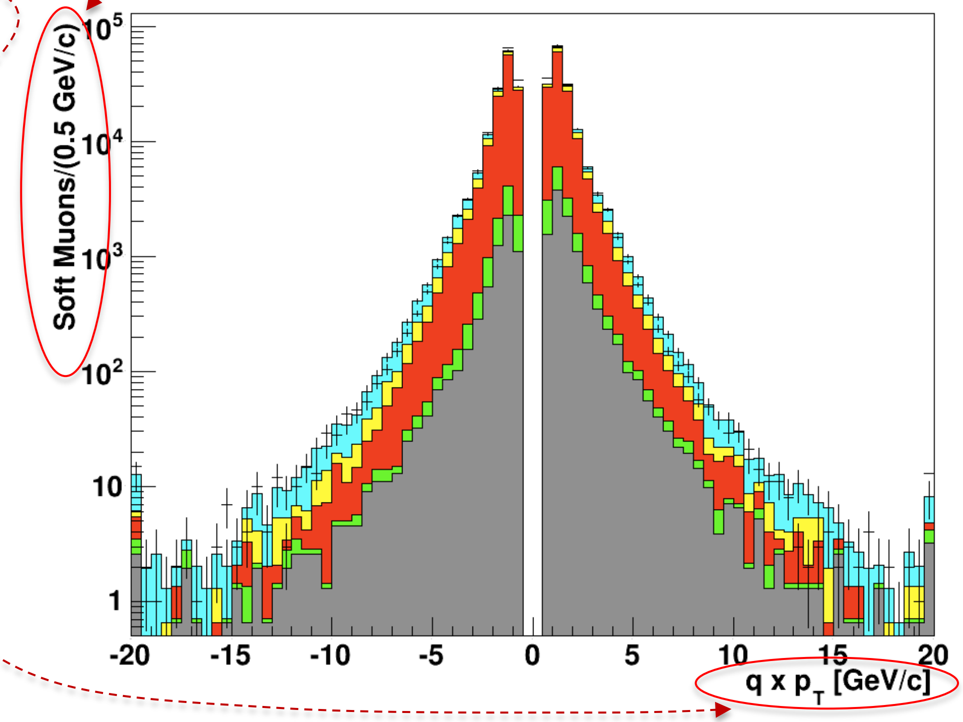
STEP-2a : solutions for the final graphics - 1

Under the sub-directory /***step2*** , find now the macro ***main2a.C.*** Create also ***./Plots .*** Now:

root [0] **.x main2a.C("png","SOFT","qtimespt","doub","1gpv","Log","Step2”)**

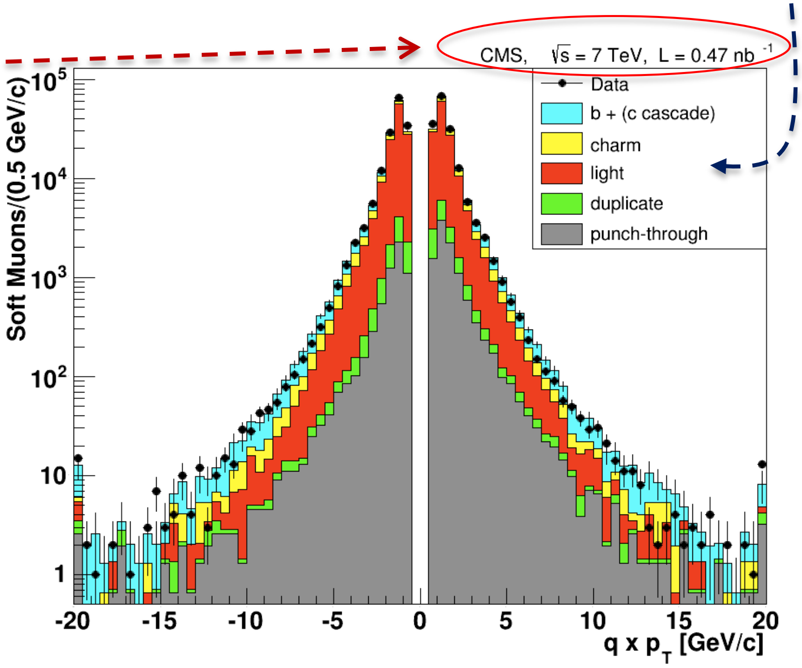
Additional code:

```
///////////////// code to enhance the graphical appearence and provide a publishible plot
//
char nuovotitolo[255];
float bin_width=hData->GetBinWidth(1.);                    //O quello che ti pare
sprintf(nuovotitolo, "Soft Muons/(%.1f GeV/c)", bin_width);
hData->GetYaxis()->SetTitle(nuovotitolo);
hData->GetXaxis()->SetTitle("q x p_{T} [GeV/c]");
```



STEP-2a : solutions for the final graphics - 2

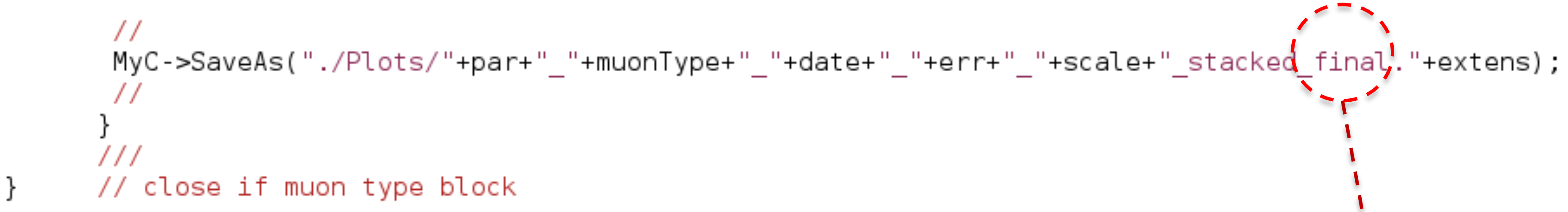
```
double xminleg = .63; double yminleg = .63; double xmaxleg = .90; double ymaxleg = .90;
TLegend *txt = new TLegend(xminleg,yminleg,xmaxleg,ymaxleg);
txt->SetTextSize(0.03);
txt->SetTextAlign(12);
txt->SetTextFont(42);
txt->SetShadowColor(0);
txt->SetFillStyle(0);
//
txt->AddEntry(hData, "Data", "LP");
txt->AddEntry(h4, "b + (c cascade)", "F");
txt->AddEntry(h3, "charm", "F");
txt->AddEntry(h2, "light", "F");
txt->AddEntry(h1, "duplicate", "F");
txt->AddEntry(hFake, "punch-through", "F");
//
txt->Draw();
//
TLatex* mylatex4dx = new TLatex (3.,150000.,"CMS,   #sqrt{s} = 7 TeV,  L = 0.47 nb^{ - 1}");
mylatex4dx->SetTextSize(0.032);
mylatex4dx->SetTextFont(42);
mylatex4dx->Draw("same");
//
```



STEP-2a : solutions for the final graphics - 3

Finally ...

...



```
      //
      MyC->SaveAs("./Plots/"+par+"_"+muonType+"_"+date+"_"+err+"_"+scale+"_stacked_final."+extens);
      //
     }
     ///
}    // close if muon type block
```

```
$ display ./Plots/qtimespt_SOFT_2022-10-20_Step2_Log_stacked_final.png
```