Exercise/Hands-on #1

Scientific Data Analysis Lab course

Alexis Pompili - UniBA

alexis.pompili@ba.infn.it

The aim of this 1st exercise

In this exercise we will learn to:

1a) execute the main **ROOT** commands

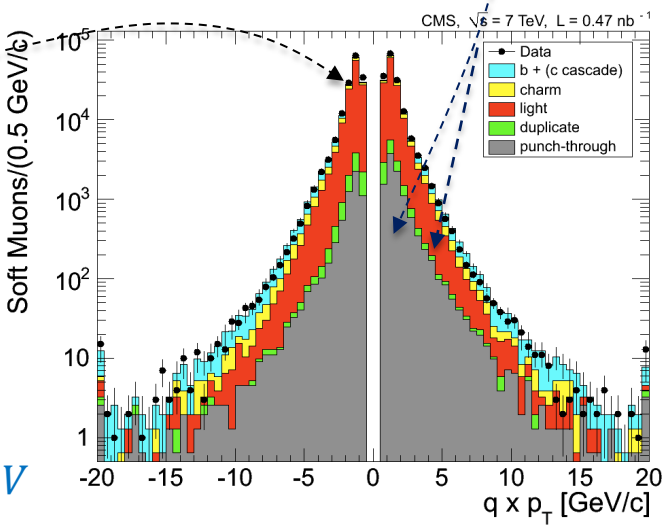
1b) execute a *macro* written in C (within the **ROOT** framework))

2) manipulate histograms

3) make a comparison between *real* & *simulated* data with an absolute normalization

4) represent the various simulated components by means of ***stacked plots***

5) prepare a plot (output of **ROOT**) with features & quality of a scientific publication.



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)***.

alexis.pompili@ba.infn.it

INTRODUCTION

alexis.pompili@ba.infn.it

Preliminary knowledge

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

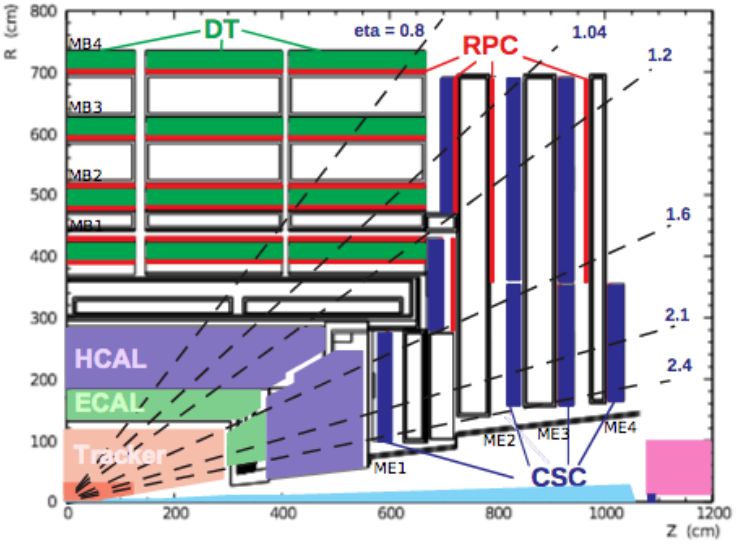
- ***Soft*** Muons***, Tight*** Muons
- ***Prompt*** Muons***,***
- Muons from ***Beauty*** *decays* ***,***
- Muons from ***Charm,***
- Muons from ***Light Hadrons decays***
- ***Fake*** Muons ("***hadronic punch-through***"), ***Duplicates*** ("***Ghost***" Muons)

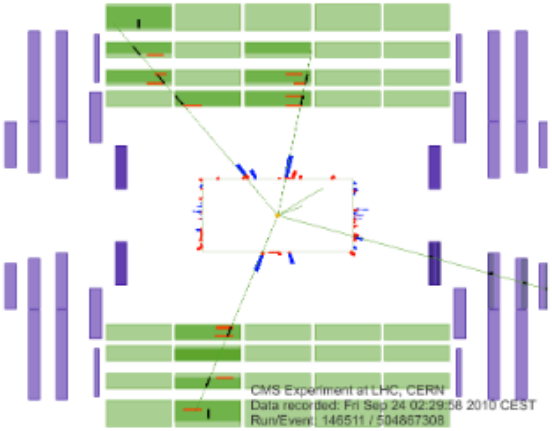
In the next slides I provide some "quick&dirty" preliminary info but - please - read the paper!

alexis.pompili@ba.infn.it

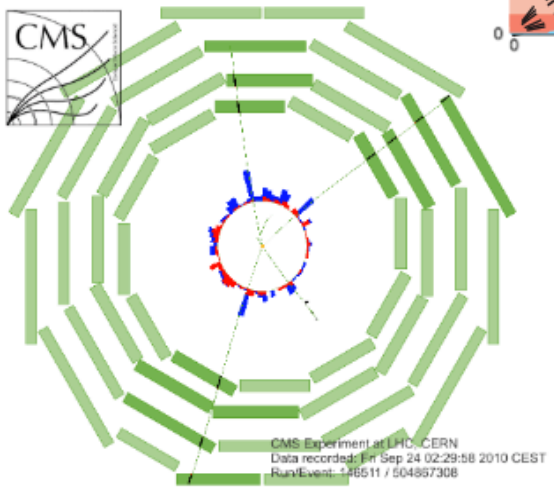
Preliminary knowledge : CMS muon chambers system

Longitudinal layout of one quadrant





(a)

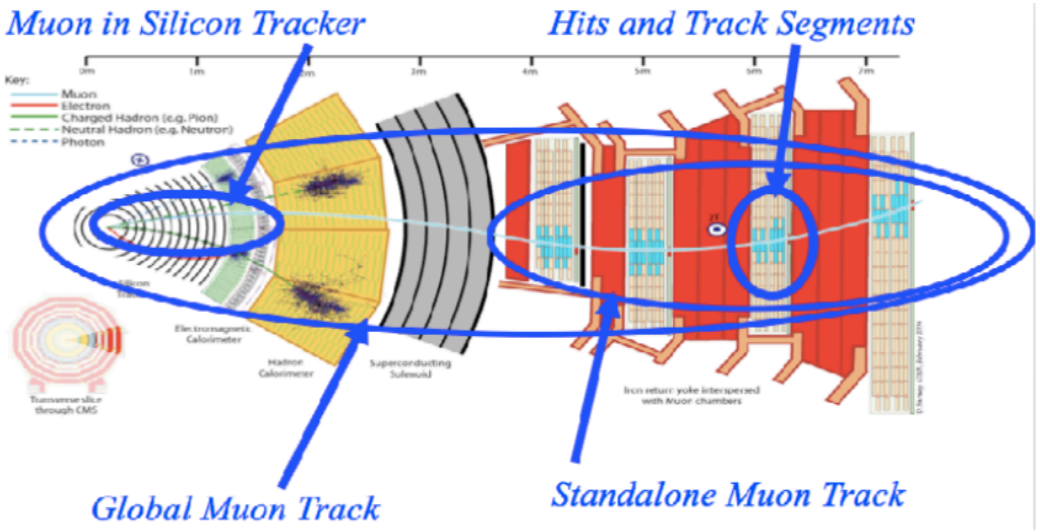


(b)

Figure 2. (a) The longitudinal (r-z) and b) the transverse (r-ϕ) views of a collision event in which four muons were reconstructed. The green (thin) curves in the inner cylinder represent tracks of charged particles reconstructed in the inner tracker with transverse momentum $p_T > 1\,\text{GeV}/c$; those extending to the muon system represent tracks of muons reconstructed using hits in both inner tracker and the muon system. Three muons were identified by the DTs and RPCs, the fourth one by the CSCs. Short black stubs in the muon system show fitted muon-track segments; as the z position is not measured in the outer barrel station, the segments in it are drawn at the z centre of the wheel, with their directions perpendicular to the chamber. Short red (light) horizontal lines in the r-z view indicate positions of RPC hits; energy depositions in the ECAL and HCAL are shown as red (light) and blue (dark) bars, respectively.

Preliminary knowledge : muon reconstruction

The muon reconstruction in CMS is obtained mainly with two different (complementary) approaches:



1) ***Global*** muon: the standalone muon track, built in the muon chambers, is extrapolated *inwards* and a tracker track is considered if satisfying the matching requirements; an overall fit provides the ***global muon***.

2) ***Tracker*** muon: a tracker track is extrapolated *outwards* and if at least a track segment in the muon chambers satisfies the matching requirements this is added to the tracker track to build a ***tracker muon***.

Adding few **identification** requirements … (quality)

- the ***Tracker*** **muon** is the so-called ***Soft*** Muon
- the ***Global*** **muon** is the so-called ***Tight*** Muon

Note: for low values of the transverse momentum ($p_T \lessapprox 5GeV$) of the muon candidate the second approach is more efficient since it requires just 1 track segment in the muon chambers; instead, the first approach becomes efficient with 2 or more track segments.

alexis.pompili@ba.infn.it

Preliminary knowledge : muons' source in simulation - I

- In the range of $p_T \lessapprox 30GeV$ the most abundant source of muons is the semi-leptonic decays of heavy flavour hadron (i.e. b-hadrons & c-hadrons): $B \to D\mu\nu, D \to X\mu\nu$ ($X =$ light hadron)

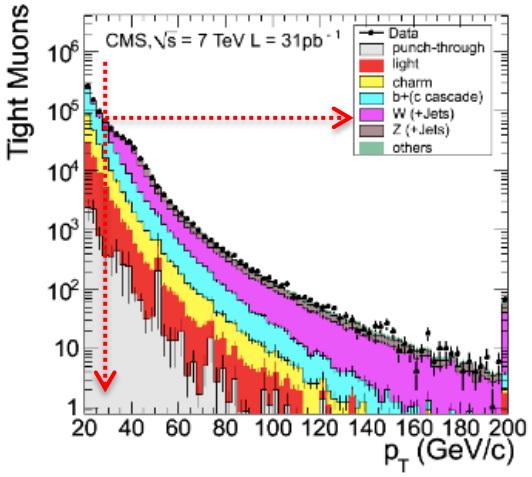
  This contribution is accompanied by a high rate of …

  (a) muon candidates produced by light-flavour hadron decays,

  (b) fake muons mainly from hadron showers not fully contained in the calorimeters.

  The relative weights of these background contributions are sensitive to the detrials of muon selection.

- In the range of $p_T > 30GeV$ the muons from W+Z bosons dominate the p_T -spectrum:



- In the simulation, for each reconstructed muon, the hits in the muon system can be associated unambiguously with the simulated particle that produced them (by means of the so-called *Monte Carlo truth*) and this allows the classification into categories as discussed in the next slide!

Preliminary knowledge : muons' source in simulation - II

alexis.pompili@ba.infn.it

- *Prompt muons.* Here the majority of muon chamber hits associated with the reconstructed muon candidate were produced by a muon, arising either from decays of W, Z, and promptly produced quarkonia states, or other sources such as Drell-Yan processes or top quark production. These individual sources are shown separately where appropriate.

- *Muons from heavy flavour.* Here the majority of muon chamber hits of the muon candidate were again produced by a muon, but the muon's parent particle was a beauty or charmed hadron, or a τ lepton. This class of events has been split according to the heaviest flavour generated in the event. Hence, *beauty* includes muons from direct b-hadron decays, from cascade b $\rightarrow$ c hadron decays, as well as cascade decays of τ leptons from b hadrons.

- *Muons from light flavour.* In this category, the majority of muon chamber hits of the muon candidate were produced by a muon arising from a decay in flight of light hadrons (π and K) or, less frequently, from the decay of particles produced in nuclear interactions in the detector material. This category includes hadrons whose tracks reconstructed in the tracker were mistakenly matched to the muon chamber hits.

real muons

---

fake muons

- *Hadron punch-through.* Here the majority of muon chamber hits of the misidentified muon candidate were produced by a particle that was not a muon. "Punch-through" (i.e., hadron shower remnants penetrating through the calorimeters and reaching the muon system) is the most common source of these candidates, although "sail-through" (i.e., particles not undergoing nuclear interactions upstream of the muon system) is present as well.

- *Duplicate.* If one simulated particle gives rise to more than one reconstructed muon candidate, that with the largest number of matched hits is assigned to one of the above categories, and any others are labeled as "duplicate". Duplicate candidates can arise either from failures of the pattern recognition of the reconstruction software, or from patterns that mimic multiple candidates.

alexis.pompili@ba.infn.it

Preliminary knowledge : muons' source in simulation - III

The data sample used in this exercise (about data-to-simulation comparison) was collected in the earliest LHC run (2010) with the so-called ***zero-bias trigger***. The latter, skipping the details, provides a **fully inclusive sample of low-p_T muons** (its simple trigger logic selects 99% of the events from pp collisions having at least one reconstructed muon of some kind).
The corresponding simulated sample consists of ***minimum-bias events*** generated with PYTHIA.

Table 1. Composition by source of the low-p_T muon candidates reconstructed in zero-bias events, according to simulation for the Soft and Tight Muon selections.

Muon source	Soft Muons [%]	Tight Muons [%]
beauty	4.4	22.2
charm	8.3	21.9
light flavour	79.0	55.7
hadron punch-through	5.4	0.2
duplicate	2.9	<0.01
prompt	$\lesssim$0.1	$\lesssim$0.1

Table 1 lists the sources of muons according to simulation. The majority of reconstructed muon candidates originate from decays in flight of pions and kaons ("light flavour"). This is particularly evident for Soft Muons, while Tight Muons have larger heavy-flavour components. For both selections the contribution of muons from heavy-flavour decays increases with p_T. The Tight Muon selection reduces the hadron punch-through contribution to 0.2% while it is about 5% in Soft Muons. The measurements of muon misidentification probabilities presented in section 5.3 confirm that the simulation correctly estimates the probability for light hadrons to be misidentified as muons.

EXERCISE: HOW-TO

Remember that you can access the virtual machine dedicated to this course by doing:

`ssh -Y yourUserName@212.189.202.110`

and then : **`source login_corso21.sh`**

alexis.pompili@ba.infn.it

Files with real and simulated data to begin with …

Create the subdirectory Esercitazione-1 in /home/username/: **`mkdir Esercitazione-1`**
Within /home/username/Esercitazione-1 create, two sub-directories:
`mkdir rootfiles` , **`mkdir Plots`**

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

- ***Histos_Data_ZeroBias_1aprnew_goodZB_last_OK.root***
- ***Histos_Mc_MinBias_1aprnew_goodZB_last.root***

To run ROOT

In the subdir /home/username/Esercitazione-1/ … I will copy the macro ***main1.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 - I

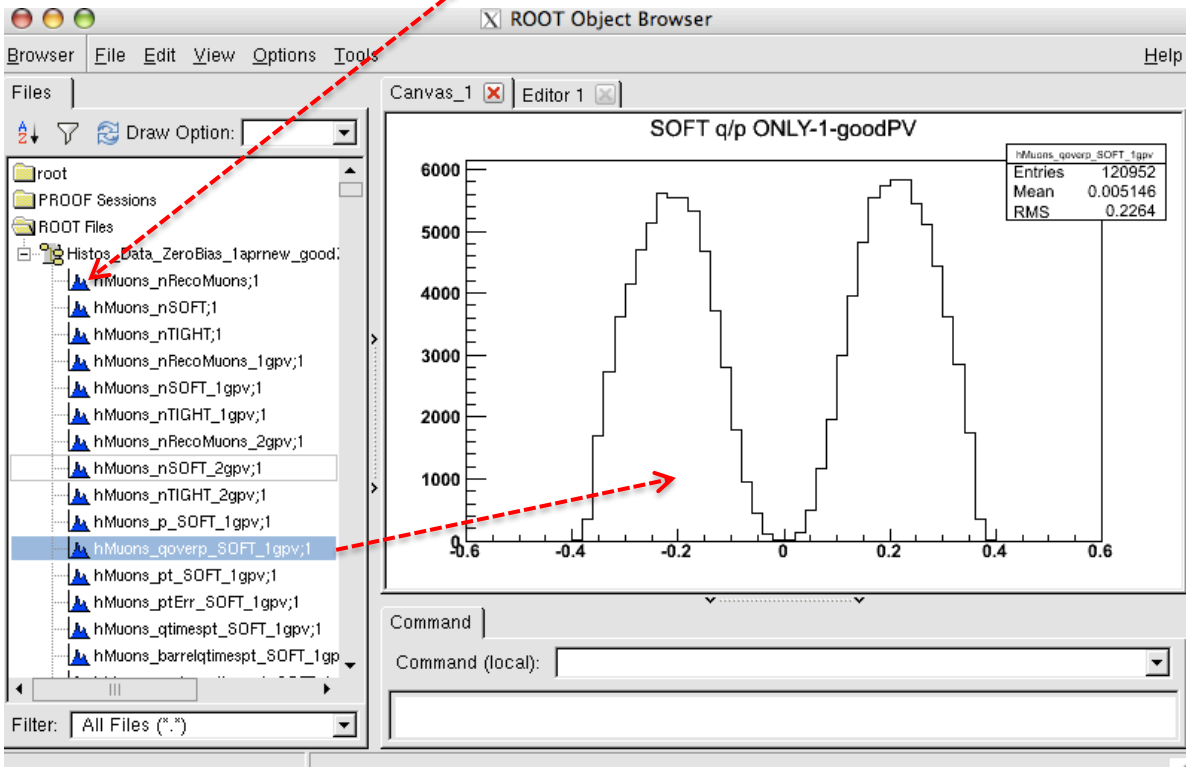
To open & inspect the rootuple *Histos_Mc_MinBias_1aprnew_goodZB_last.root* with ROOT, do:

```
$ 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 - II

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()`

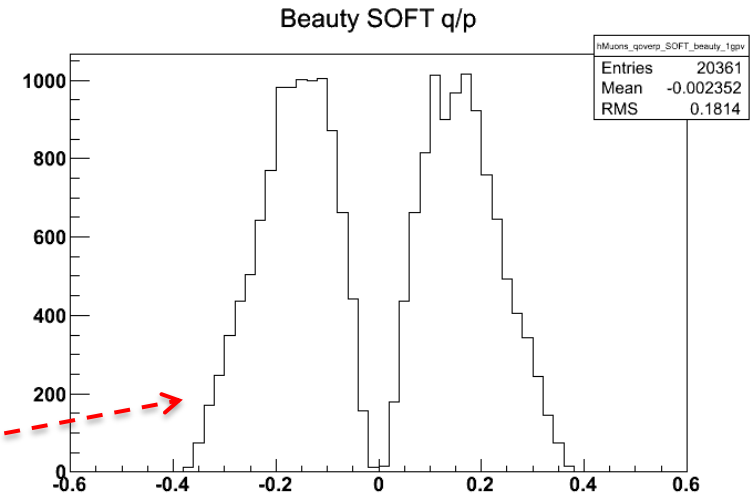
```
TFile**         Histos_Mc_MinBias_1aprnew_goodZB_last.root
 TFile*         Histos_Mc_MinBias_1aprnew_goodZB_last.root
  KEY: TH1I     hMuons_nRecoMuons;1     number of reco muons per event
  KEY: TH1I     hMuons_nSOFT;1  number of soft muons per event
  KEY: TH1I     hMuons_nTIGHT;1 number of tight 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()`



How-to-execute the macro

In the sub-directory ***/home/username/Esercitazione-1/*** you can find the *macro file* ***main1.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 main1.C("4Oct")`

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

Example:

```
[pompili@vm-pompili Esercitazione1]$ root -l
[root [0] .x main1.C("4Oct")
 Info in <TCanvas::Print>: file ./Plots/qtimespt_SOFT_4Oct_log.png has been created
 Absolute Normalization Scale Real/Simulated =0.64904
 Info in <TCanvas::Print>: file ./Plots/qtimespt_SOFT_4Oct_log_norm.png has been created
 Info in <TCanvas::Print>: file ./Plots/qtimespt_SOFT_4Oct_log_norm_stacked.png has been created
 Info in <TCanvas::Print>: file ./Plots/qtimespt_SOFT_4Oct_log_norm_stacked_ok.png has been created
 root [1]
```

Note: the *.png files (carrying the output plots) will be stored in the sub-dir */home/username/Esercitazione-1/Plots/* ; to visualize them do:

```
[pompili@vm-pompili Esercitazione1]$ display ./Plots/qtimespt_SOFT_4Oct_log_norm_stacked_ok.png &
```

EXERCISE: the CODE in the macro

alexis.pompili@ba.infn.it

The code : setup - I

First of all the includes to link the classes of the libraries useful for the code:

```
#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>
//
```

it expects to be executed passing it a String (in double brackets, i.e. "…"):

```
.x main1.C("4Oct")
```

The core part is :

```
void main(TString date)
{
…
…… here  the implemented code enters …
…
}
```

Let's inspect the code now :

The code : setup - I alternative

Alternatively :

```
void main(TString date)
{
…
…… here  the implemented code enters …
…
}
```

```
.x main1.C()
```

Let's instead use this function :

```
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()};
//
```

… in this way … the date is not passed anymore from outside through the interface but is defined inside the macro and taken from the machine clock!

As a result the ***const TString*** would take value: 2022-10-4

The code : setup - II

First part of the code is devoted to “preparation” :

```
//--> reset memory
gROOT->Reset();
gROOT->Clear();
//
//--> reset style
gROOT->SetStyle("Plain");
//gROOT->ForceStyle(); // (*)
//
// -- choose log
gStyle->SetOptLogy();
// --- gStyle->SetOptLogy(0); // to go back to linear scale
//
//--> prepare your canvas (the graphical object that will host your plots)
//
TCanvas *MyC = new TCanvas("MyC","Plots",1000,800);
//
MyC->SetBorderSize(2);
MyC->SetFrameFillColor(0);
MyC->SetGridx(0);
MyC->SetGridy(0);
//
MyC->cd();
//
//======================================
//
//=== MC FILE with Histograms
//
TFile f1("./rootfiles/Histos_Mc_MinBias_1aprnew_goodZB_last.root","read"); // open only in READ mode !
//
//=== DATA FILE with Histograms
//
TFile f2("./rootfiles/Histos_Data_ZeroBias_1aprnew_goodZB_last_OK.root","read");
//
////////////////////////////////////////////////////////////////////////////////////////////////////
//
```

The code : starting histograms (data &MC)

Load/import the content from the histograms hosted in the 2 external root files;
here we choose the variable under study (**`qtimespt`**) for which type of muon (**`SOFT`**)

[of course there are many other variables (*) and also the **`TIGHT`** type of muon]:

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//
//--- declare the histograms I will use and import the content of the histograms in the Monte Carlo file
TH1D *hBeautyFlavour;
TH1D *hCharmFlavour;
TH1D *hLightHadrons;
TH1D *hGhost;
TH1D *hFake;
//
hBeautyFlavour=(TH1D*)f1.Get("hMuons_qtimespt_SOFT_beauty_1gpv");
hCharmFlavour=(TH1D*)f1.Get("hMuons_qtimespt_SOFT_charm_1gpv");
hLightHadrons=(TH1D*)f1.Get("hMuons_qtimespt_SOFT_light_1gpv");
hGhost=(TH1D*)f1.Get("hMuons_qtimespt_SOFT_ghost_1gpv");
hFake=(TH1D*)f1.Get("hMuons_qtimespt_SOFT_fake_1gpv");
//
//--- these two real data histograms need to be summed (please do not care about that:it is for just historical reasons)
//
TH1D *hData1gpv;
TH1D *hData2gpv;
//
hData1gpv=(TH1D*)f2.Get("hMuons_qtimespt_SOFT_1gpv");
hData2gpv =(TH1D*)f2.Get("hMuons_qtimespt_SOFT_2gpv");
//
TH1D *hData = (TH1D*)hData1gpv->Clone("hData");
hData->Add(hData1gpv,hData2gpv,1,1);   // (a*h1 + b* h2) (with unitary coefficients in the linear combination: a=1 e b=1)
hData->Sumw2();                        // store the sum of squares of weights
//
// now I can delete the single histograms that i have added:
delete hData1gpv, hData2gpv;
//
```

histograms' addition

(*) just be careful that variables have different type (`double`, `int`, …)

alexis.pompili@ba.infn.it

The code - visualization of starting histograms - I

Let's prepare for visualization all the 6 histograms (5 for MC components/sources and 1 for data) that will be used later (here we take care to visualize also underflows and overflows):

```
//
//////////////////////////////////////////////////////////////////////////////
//////////////////////////// Let's just visualize the plots //////////////////////
//////////////////////////////////////////////////////////////////////////////
//
// need the prepare the histograms first (handle underflows and overflows)
//
// - FAKE
//
//-overflow
int nBins_hFake = hFake->GetNbinsX();
int nBins_ovflw_hFake = hFake->GetBinContent(nBins_hFake + 1); // the overflow content is in bin n+1
hFake->AddBinContent(nBins_hFake,nBins_ovflw_hFake);      // add the content of bin n+1 to that of bin n (last regular bin)
//-underflow
int nBins_unflw_hFake = hFake->GetBinContent(0); // underflow content in bin 0
hFake->AddBinContent(1,nBins_unflw_hFake);       // add the content of bin 0 to that of bin 1 (first regular bin)
//
// - GHOST
//
//-overflow
int nBins_hGhost = hGhost->GetNbinsX();
int nBins_ovflw_hGhost = hGhost->GetBinContent(nBins_hGhost + 1);
hGhost->AddBinContent(nBins_hGhost,nBins_ovflw_hGhost);
//-underflow
int nBins_unflw_hGhost = hGhost->GetBinContent(0);
hGhost->AddBinContent(1,nBins_unflw_hGhost);
//
// - LIGHT HADRONS
//
//-overflow
int nBins_hLH = hLightHadrons->GetNbinsX();
int nBins_ovflw_hLH = hLightHadrons->GetBinContent(nBins_hLH + 1);
hLightHadrons->AddBinContent(nBins_hLH,nBins_ovflw_hLH);
//-underflow
int nBins_unflw_hLH = hLightHadrons->GetBinContent(0);
hLightHadrons->AddBinContent(1,nBins_unflw_hLH);
//
```

The visualization of the overflow/underflow in the histograms it is not a **ROOT** default and must be done by hand! (for each of the 6 histograms)

The code - visualization of starting histograms - II

```
//
// - CHARM FLAVOUR
//
//-overflow
int nBins_hCF = hCharmFlavour->GetNbinsX();
int nBins_ovflw_hCF = hCharmFlavour->GetBinContent(nBins_hCF + 1);
hCharmFlavour->AddBinContent(nBins_hCF,nBins_ovflw_hCF);
//-underflow
int nBins_unflw_hCF = hCharmFlavour->GetBinContent(0);
hCharmFlavour->AddBinContent(1,nBins_unflw_hCF);
//
// - BEAUTY FLAVOUR
//
//-overflow
int nBins_hBF = hBeautyFlavour->GetNbinsX();
int nBins_ovflw_hBF = hBeautyFlavour->GetBinContent(nBins_hBF + 1);
hBeautyFlavour->AddBinContent(nBins_hBF,nBins_ovflw_hBF);
//-underflow
int nBins_unflw_hBF = hBeautyFlavour->GetBinContent(0);
hBeautyFlavour->AddBinContent(1,nBins_unflw_hBF);
//
// - REAL DATA
//
//-overflow
int nBins_hData = hData->GetNbinsX();
int nBins_ovflw_hData = hData->GetBinContent(nBins_hData + 1);
hData->AddBinContent(nBins_hData,nBins_ovflw_hData);
//-underflow
int nBins_unflw_hData = hData->GetBinContent(0);
hData->AddBinContent(1,nBins_unflw_hData);
//
```

alexis.pompili@ba.infn.it

The code - visualization of starting histograms - III

Now we can visualize by plotting:

```
//
////////////////////////////// plot now
//
gStyle->SetOptStat(111111);
MyC->Divide(2,3);
MyC->cd(1);
hData->SetMarkerStyle(20);
hData->SetMarkerSize(0.55);
hData->Draw("EP");   // option to get error and point-polymarker
MyC->cd(2);
hBeautyFlavour->SetFillColor(7); // 7 or kCyan
hBeautyFlavour->Draw("");
hBeautyFlavour->Draw("Esame");  // Error + superimpose
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");
//
MyC->SaveAs("./Plots/qtimespt_SOFT_"+date+"_log.png");
gSystem->Sleep(15000); // argument is given in millisecs // so this leaves the canvas open for 15 secs
//
MyC->Update();   // decomment to update the canvas
MyC->Clear();    // decomment to clean the canvas
```

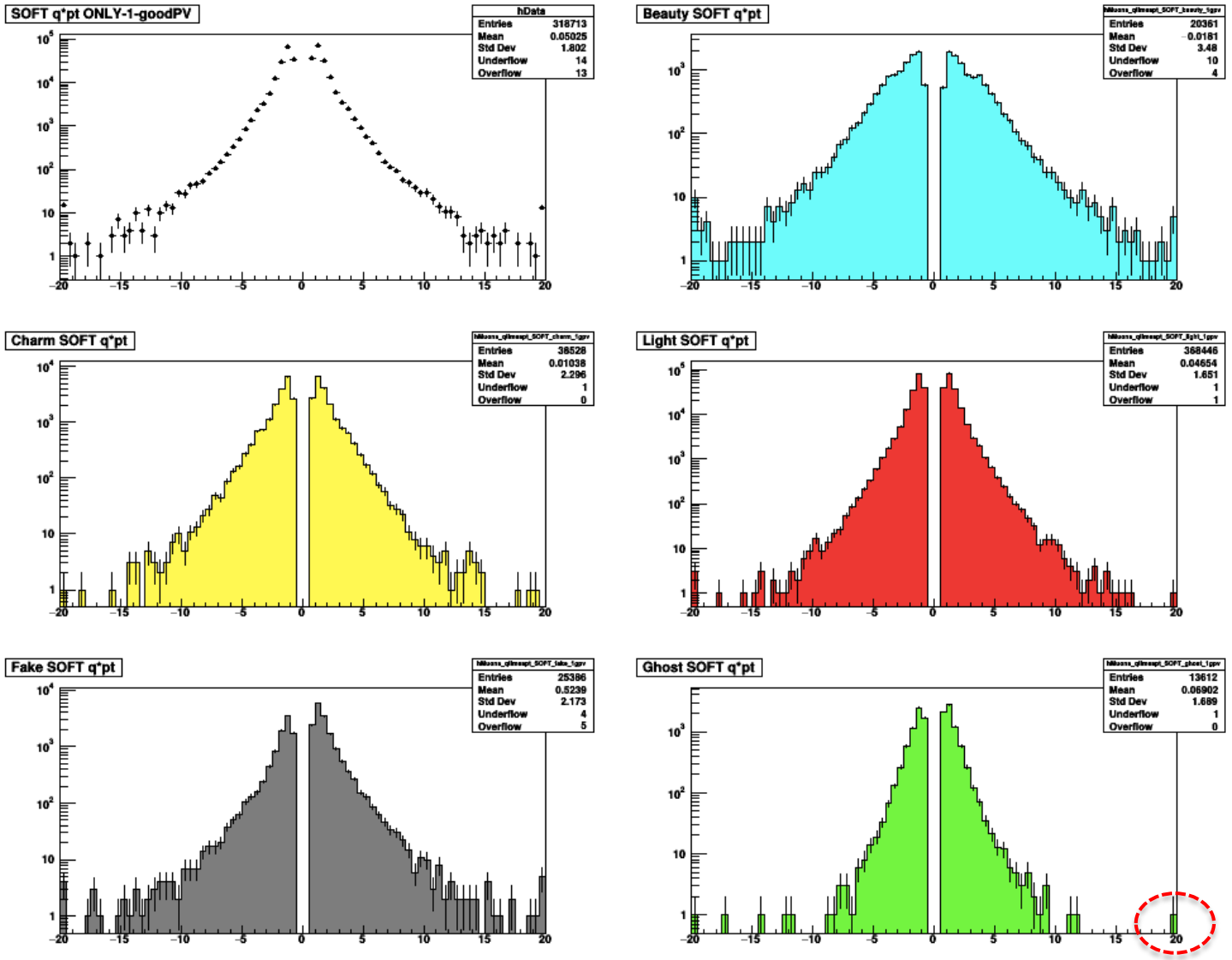
output file will be: **qtimespt_SOFT_4Oct_log.png**

alexis.pompili@ba.infn.it

alexis.pompili@ba.infn.it

Visualization of starting histograms - I

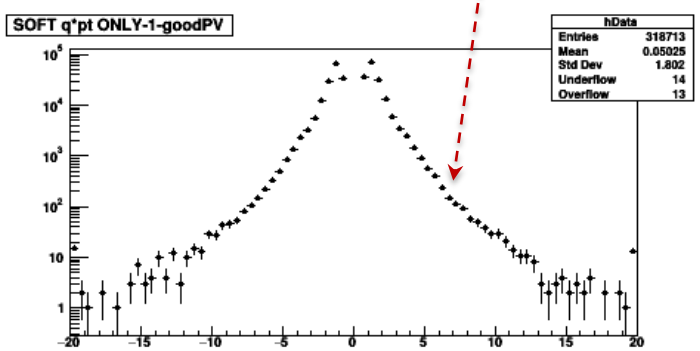
…and we get (with … `display ./Plots/qtimespt_SOFT_4Oct_log.png &`):

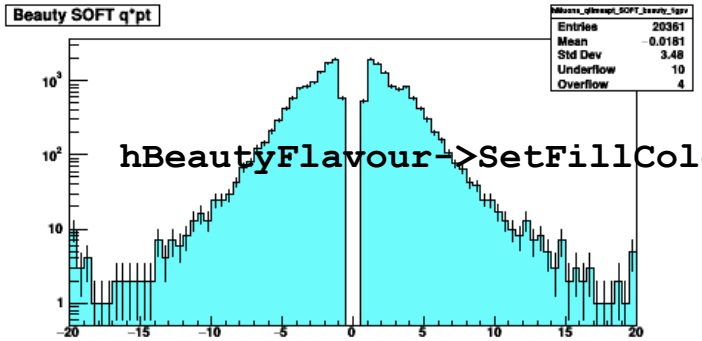


Note: the **overflows/underflows** can be appreciated typically when using the LOG scale

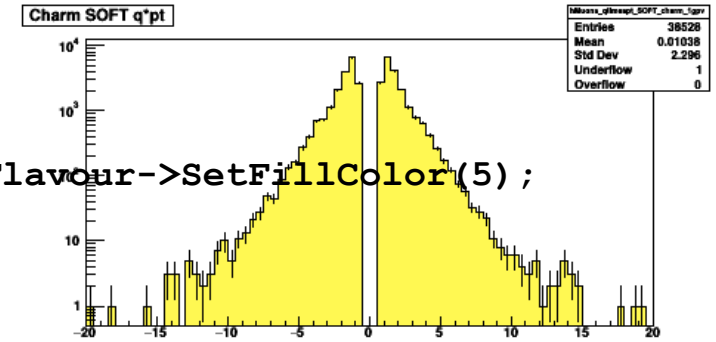
Visualization of starting histograms - II

To make appear the entries in the data histogram as little round spots: `hData->SetMarkerStyle(20);`
To give the spots the right size: `hData->SetMarkerSize(0.55);`

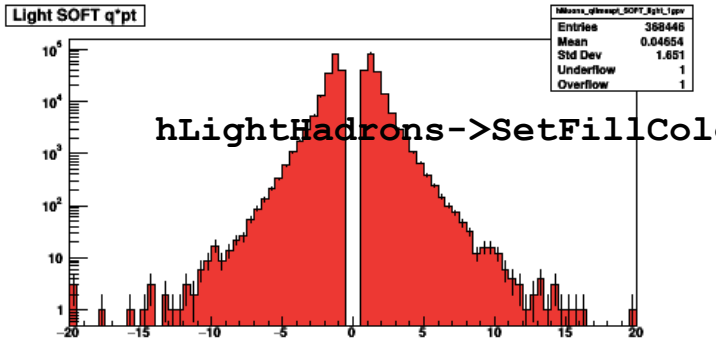




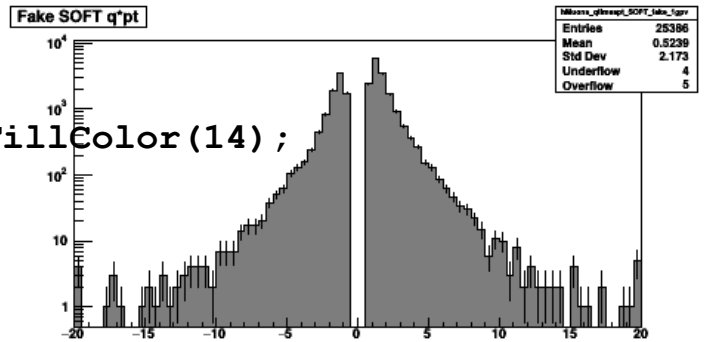
`hBeautyFlavour->SetFillColor(7);`



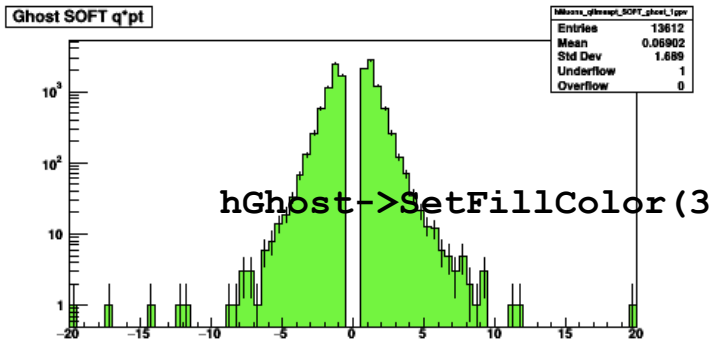
`hCharmFlavour->SetFillColor(5);`



`hLightHadrons->SetFillColor(2);`



`hFake->SetFillColor(14);`



`hGhost->SetFillColor(3);`

Data-MC comparison : absolute normalization - I

If we want to fairly compare Data and MC we can more correctly scale the MC components in order that the simulated events are scaled at the corresponding amount of events in the data:

```
//
//////////////////////////////////////////////
//
// DEFINE THE ABSOLUTE NORMALIZATION
//
// With a tool given by my experiment (CMS) I can know the integrated luminosity
// corresponding to the Runs and Lumisections of the collected data that are triggered
// by the High Level Trigger called "ZeroBias":  L_dt = 469,996 microbarn^-1
//
// I have to estimate by hand the luminosity correspndent to the used Monte Carlo samples
// (called "Minimum Bias" sample):
//
// - n. of events di MB : N = 51602200
// - cross section for Minimum Bias with the Pythia QCD Generator: Sigma = 71,26 millibarn = 71260 microbarn
// - integrated luminosity : L_mc = N/Sigma = 51602200/71260 (microbarn^-1) = 724,140 microbarn^-1
//
// In this way we derive the SCALE FACTOR DATA/MC : SF(dt/mc) = 469,996/724,140 = 0.64904
//
Double_t ScaleLumi = 0.64904;
//
cout << "Absolute Normalization Scale Real/Simulated =" << ScaleLumi << endl;
//
hFake->Scale(ScaleLumi);
hGhost->Scale(ScaleLumi);
hLightHadrons->Scale(ScaleLumi);
hCharmFlavour->Scale(ScaleLumi);
hBeautyFlavour->Scale(ScaleLumi);
//
```

$$L_{\text{int}}^{DATA}$$

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

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

the **`scale`** method scales the histogram by the given coefficient !

EXERCISE@HOME: try to apply a ***relative*** **normalization (to obtain a *shape comparison*)**

Data-MC comparison : absolute normalization - II

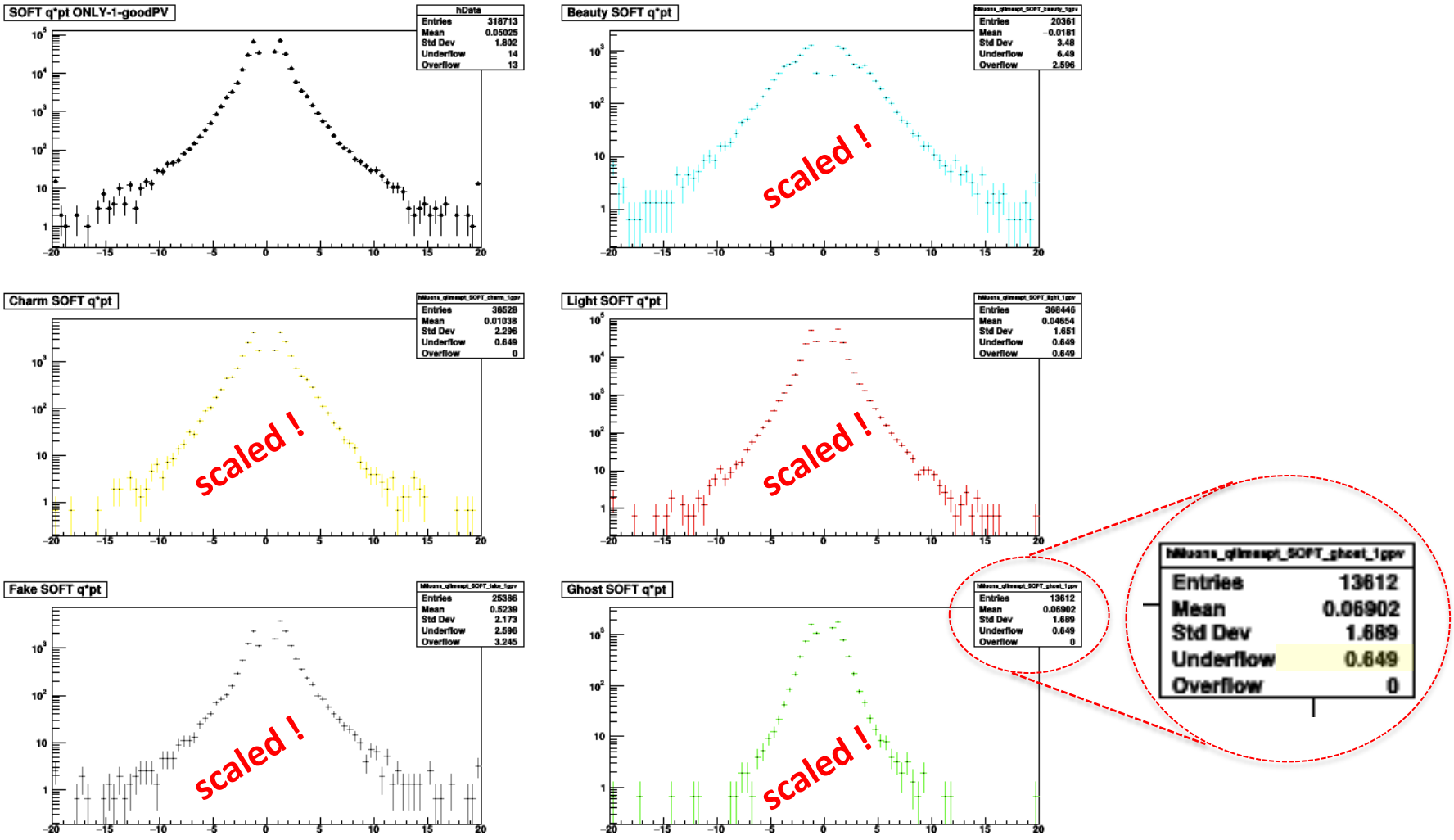
Let's now visualize the result of this "normalization scaling" by writing the plotting code:

```
//
gStyle->SetOptStat(111111);   // write the main features of the histogram
MyC->Divide(2,3);
MyC->cd(1);
hData->SetMarkerStyle(20);
hData->SetMarkerSize(0.55);
hData->Draw("EP");
MyC->cd(2);
hBeautyFlavour->SetLineColor(7); // 7 or kCyan
hBeautyFlavour->Draw("EL");
MyC->cd(3);
hCharmFlavour->SetLineColor(5);  // 5 or kYellow
hCharmFlavour->Draw("EL");
MyC->cd(4);
hLightHadrons->SetLineColor(2);   // 2 or kRed
hLightHadrons->Draw("EL");
MyC->cd(5);
hFake->SetLineColor(14);     // this is dark grey
hFake->Draw("EL");
MyC->cd(6);
hGhost->SetLineColor(3);    // 3 or kGreen
hGhost->Draw("EL");
//
MyC->SaveAs("./Plots/qtimespt_SOFT_"+date+"_log_norm.png");
//
gSystem->Sleep(15000); // argument is given in millisecs // so this leaves the canvas open for 15 secs
//
MyC->Update();
MyC->Clear();
//
```

output file will be: **qtimespt_SOFT_4Oct_log_norm.png**

… and we get:

Data-MC comparison : absolute normalization - III



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 remained the same in the statistics box !! (this may generate some confusion … but it's enough to check that overflows/underflows are no more integers to be sure of the scaling).

The code - *stacked* histograms - I

```
//////////////////////////////////////////////// Prepare STACKING for Data/MC comparison
//
TH1D *h1 = (TH1D*)hFake->Clone("h1");
TH1D *h2 = (TH1D*)h1->Clone("h2");
TH1D *h3 = (TH1D*)h2->Clone("h3");
TH1D *h4 = (TH1D*)h3->Clone("h4");
//
// Note the unitary weights in the linear combination (->arithmetic sum)
h1->Add(hFake,hGhost,1.,1.);        // h1 has 2 summed components (fake+ghost)
h2->Add(h1,hLightHadrons,1.,1.);    // h2 has 3 summed components (fake+ghost+light)
h3->Add(h2,hCharmFlavour,1.,1.);    // h3 has 4 summed components (fake+ghost+light+charm)
h4->Add(h3,hBeautyFlavour,1.,1.);   // h4 has all the 5 components summed up
//
TH1D *h5 = (TH1D*)h4->Clone("h5");  // MC total distribution // needed later
//
// In the stacking the order of the visible components will be (from bottom to top):
// hFake->fake,     h1->ghost,      h2->light,       h3->charm,       h4->beauty
//
// Now i choose the colors: same as those already chosen earlir :
// fake=gray(14), ghost=green(3), light=red(2) , charm=yellow(5), beauty=cyan(7)
//
hFake->SetFillColor(14);
h1->SetFillColor(3);
h2->SetFillColor(2);
h3->SetFillColor(5);
h4->SetFillColor(7);
//
// few other options (border color, thikness of the border)
//
hFake->SetLineColor(1); hFake->SetLineWidth(2); hFake->SetTitle("");
h1->SetLineColor(1); h1->SetLineWidth(2); h1->SetTitle("");
h2->SetLineColor(1); h2->SetLineWidth(2); h2->SetTitle("");
h3->SetLineColor(1); h3->SetLineWidth(2); h3->SetTitle("");
h4->SetLineColor(1); h4->SetLineWidth(2); h4->SetTitle("");
hData->SetTitle("");
//
//
hData->SetMinimum(0.5);   // in order to have under control the right tail (in log scale)
//
MyC->cd();
//
```

histograms as ***partial*** sums in a sequence

see next slide

The code - *stacked* histograms - II

```
//
MyC->cd();
//
hFake->SetStats(0);
h1->SetStats(0);
h2->SetStats(0);
h3->SetStats(0);
h4->SetStats(0);
h5->SetStats(0);
hData->SetStats(0);
//
//hData->UseCurrentStyle();
hData->Draw("EP");
h4->Draw("histsame");
h3->Draw("histsame");
h2->Draw("histsame");
h1->Draw("histsame");
hFake->Draw("histsame");
//
h5->Draw("Esame"); // in order to get the correct errors in the MC sum (of components)
//
hData->SetMarkerStyle(20);
//hData->SetMarkerColor(1);
hData->SetMarkerSize(0.85);
hData->Draw("Esame");
//
gPad->RedrawAxis(); // needed because the color filling covers part of the y-axis
//
MyC->SaveAs("./Plots/qtimespt_SOFT_"+date+"_log_norm_stacked.png");
//
```

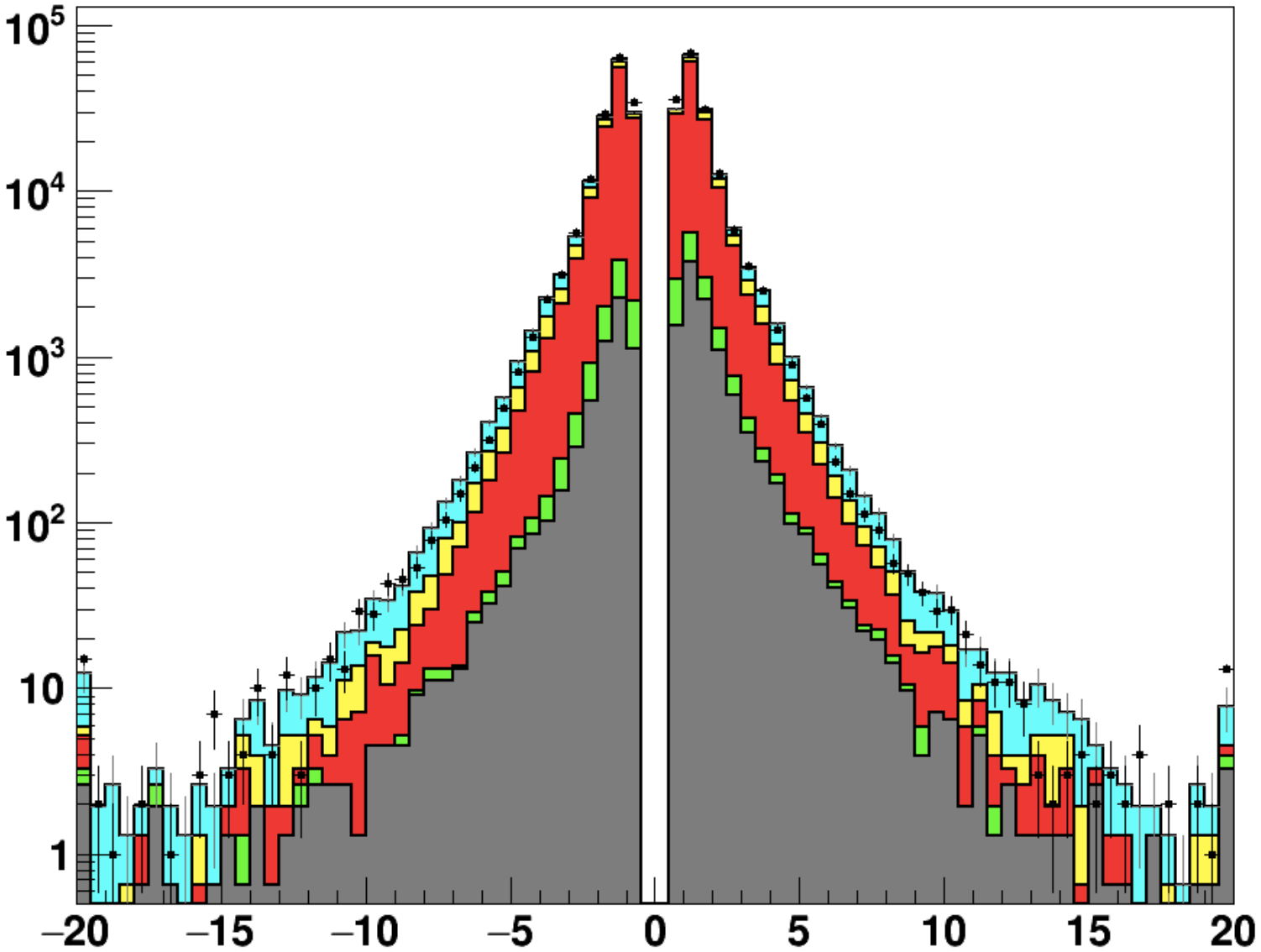
here is implemented the superposition of histograms according to the foreseen (inverted) order

output file will be: **qtimespt_SOFT_4Oct_log_norm_stacked.png**

alexis.pompili@ba.infn.it

The code - *stacked* histograms - III

… and we get:



alexis.pompili@ba.infn.it

The code - *stacked* histograms refined - I

Now we need to enhance the graphics:

```
////////////////////////////////////////////////
//
// code to enhance the graphical appearence and provide a publishible plot
//
char newtitle[255];
float bin_width=hData->GetBinWidth(1.);    // 0 ... whatever
//
sprintf(newtitle, "Soft Muons/(%.1f GeV/c)", bin_width);
hData->GetYaxis()->SetTitle(newtitle);
hData->GetXaxis()->SetTitle("q x p_{T} [GeV/c]");
//
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->SetTextFont(42);
mylatex4dx->SetTextSize(0.032);
mylatex4dx->Draw("same");
//
MyC->SaveAs("./Plots/qtimespt_SOFT_"+date+"_log_norm_stacked_ok.png");
//
////////////////////////////////////////////////
//
// \*
delete MyC; // decomment to free the memory occupied by the canvas at the end of the execution
//
f1.Close();
f1.Delete();
////////
f2.Close();
f2.Delete();
//
gROOT->Reset();
gROOT->Clear();
gROOT->ForceStyle();
//  */
```

(a)

(b)

(legend)

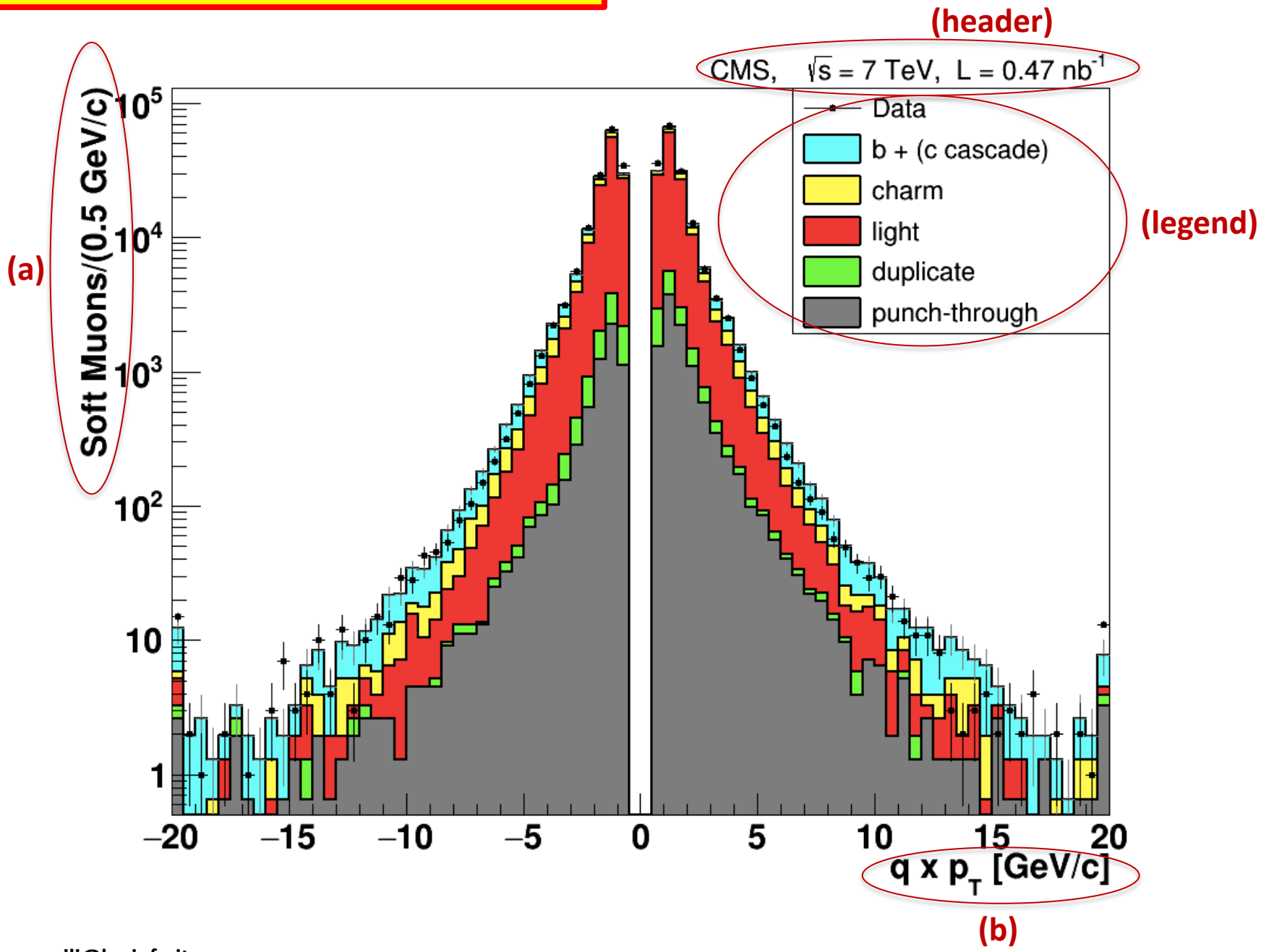
(header)

output file will be:

qtimespt_SOFT_4Oct_log_norm_stacked_ok.png

The code - *stacked* histograms refined - II

… and we get:



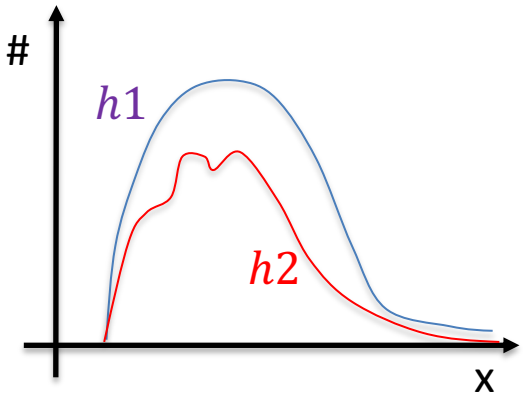
alexis.pompili@ba.infn.it

EXERCISE: compare 2 histograms by relative normalization

alexis.pompili@ba.infn.it

A data-MC comparison with absolute normalization that we have seen in the exercise is really a stringent test of both the reconstruction algorithms and the accuracy of the generation/simulation.

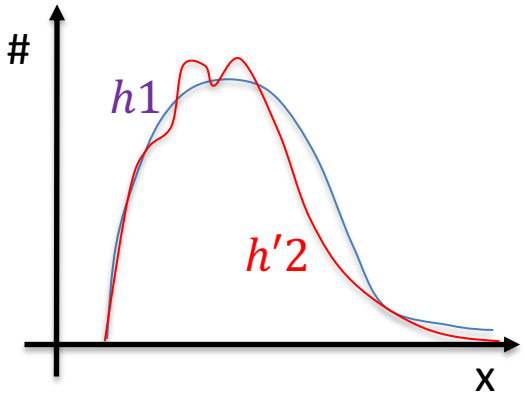
While the absolute normalization is a bin-by-bin comparison test, when doing a relative normalization only the shapes of the distributions are compared; this is because ….
in the relative normalization one histogram is normalized to the area of the other and the constraint on having the same area has some impact in the comparison.
Let me visualize with a sketch this aspect:



h_1 has $\#_1$ entries

h_2 has $\#_2$ entries

$$h'2 = h2 \cdot \frac{\#_1}{\#_2}$$



$h'2$ has $\#_1$ entries
(it is normalized to h_1)

As exercise compare two MC components !