Jupyter Notebook in ALICE MasterClass

Piotr Nowakowski

10/15/2020

Jupyter Notebook in ALICE MasterClass

- Origin of idea
- Implementation
- Presentation
- Encountered problems

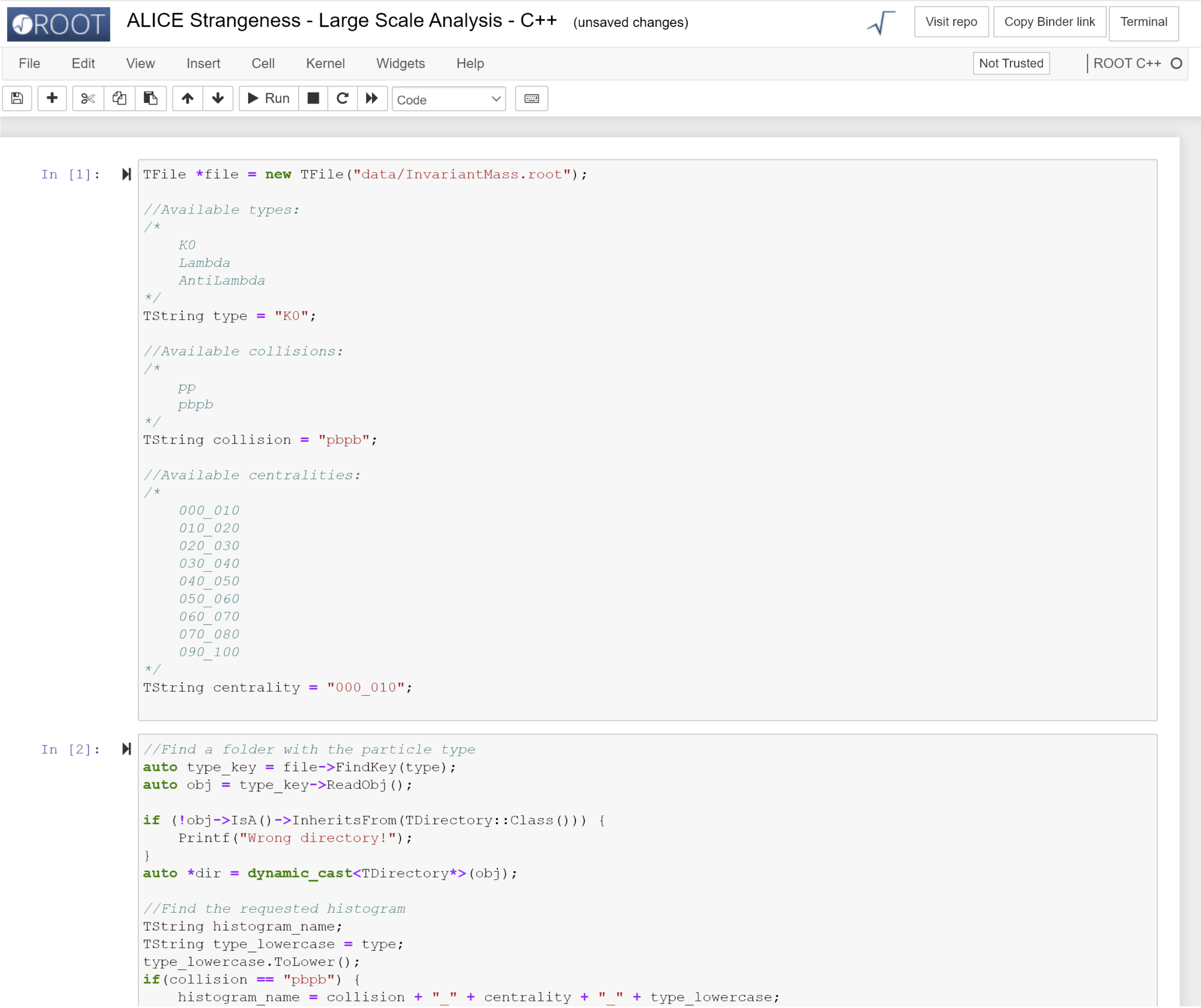
Origin of idea

- **It has been brought to my attention that the University of Münster prepared an online version of Nuclear Modification Factor (“RAA”) exercise – the Large Scale Analysis part**
- **This version is based on Jupyter Notebook, in which a student can write Python code in their browser and immediately see results “live”**
  - This technology is frequently used in university lectures when teaching programming
- **The data was converted from ROOT to a different format, readable by standard data-science Python libraries (i.e. Pandas); The exercise itself was also reimplemented to be doable in Python**
- **Python is fine, but is conversion really necessary?**

Implementation

- **C++ ROOT can also be used in Jupyter Notebook instead of Python**
  - See 2016 CHEP: *"The new ROOT interface: Jupyter Notebooks"*
- **This can be hosted in CERN SWAN (requires CERN account) or via *mybinder.org* website**
- **The example of setting up ROOT for *mybinder.org* in the mentioned presentation is outdated and no longer works :(**
- ***mybinder.org* uses Docker containers to create Jupyter Notebooks – I have managed to create from scratch a working configuration which runs ROOT Jupyter**
- **I adapted the code from MasterClass Strangeness exercise (Large Scale Analysis part) to test this approach**

Presentation



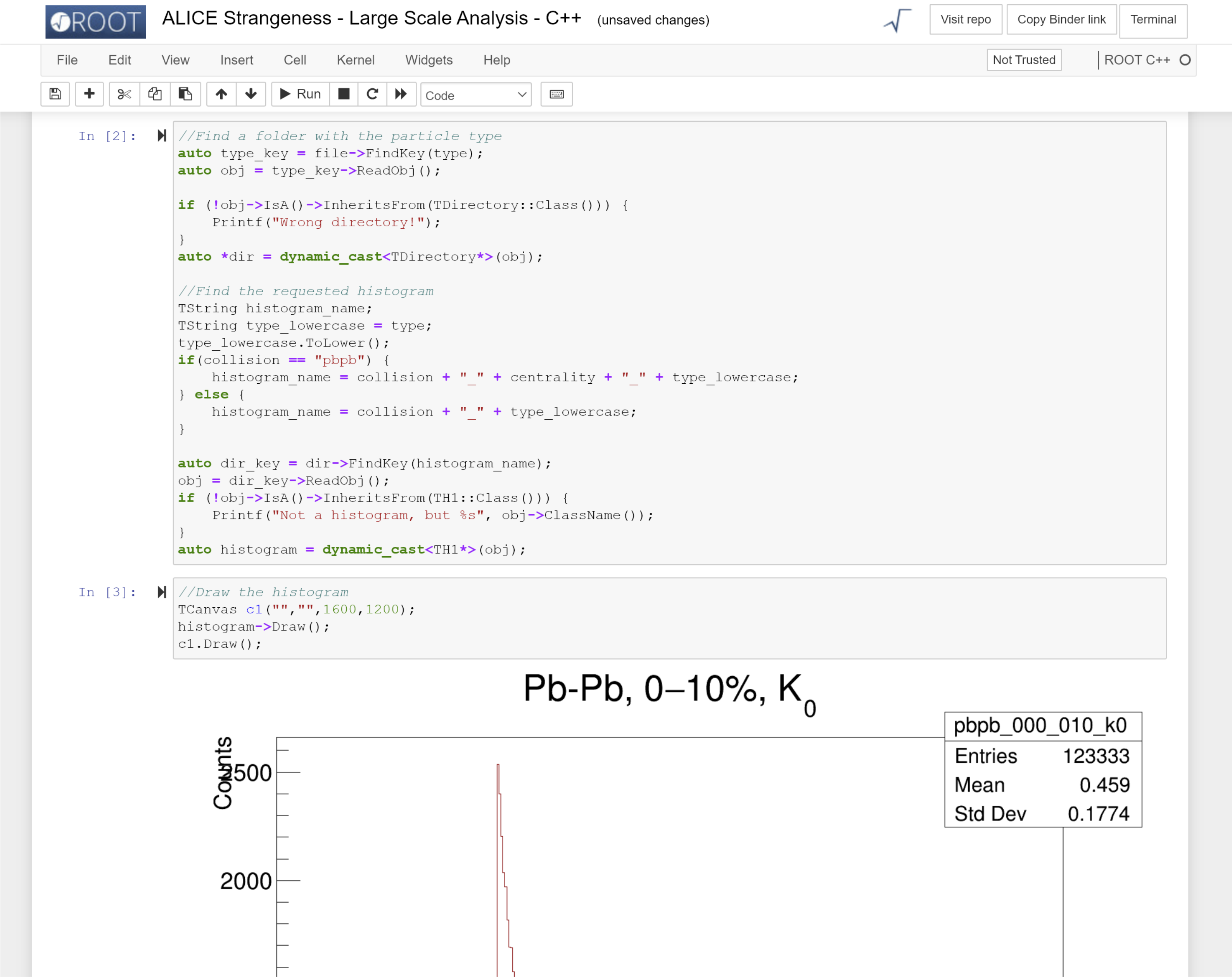
In [1]:

```
TFile *file = new TFile("data/InvariantMass.root");

//Available types:
/*
    K0
    Lambda
    AntiLambda
*/
TString type = "K0";

//Available collisions:
/*
    pp
    pbpb
*/
TString collision = "pbpb";

//Available centralities:
/*
    000_010
    010_020
    020_030
    030_040
    040_050
    050_060
    060_070
    070_080
    090_100
*/
TString centrality = "000_010";
```

In [2]:

```
//Find a folder with the particle type
auto type_key = file->FindKey(type);
auto obj = type_key->ReadObj();

if (!obj->IsA()->InheritsFrom(TDirectory::Class())) {
    Printf("Wrong directory!");
}
auto *dir = dynamic_cast<TDirectory*>(obj);

//Find the requested histogram
TString histogram_name;
TString type_lowercase = type;
type_lowercase.ToLower();
if(collision == "pbpb") {
    histogram_name = collision + "_" + centrality + "_" + type_lowercase;
```

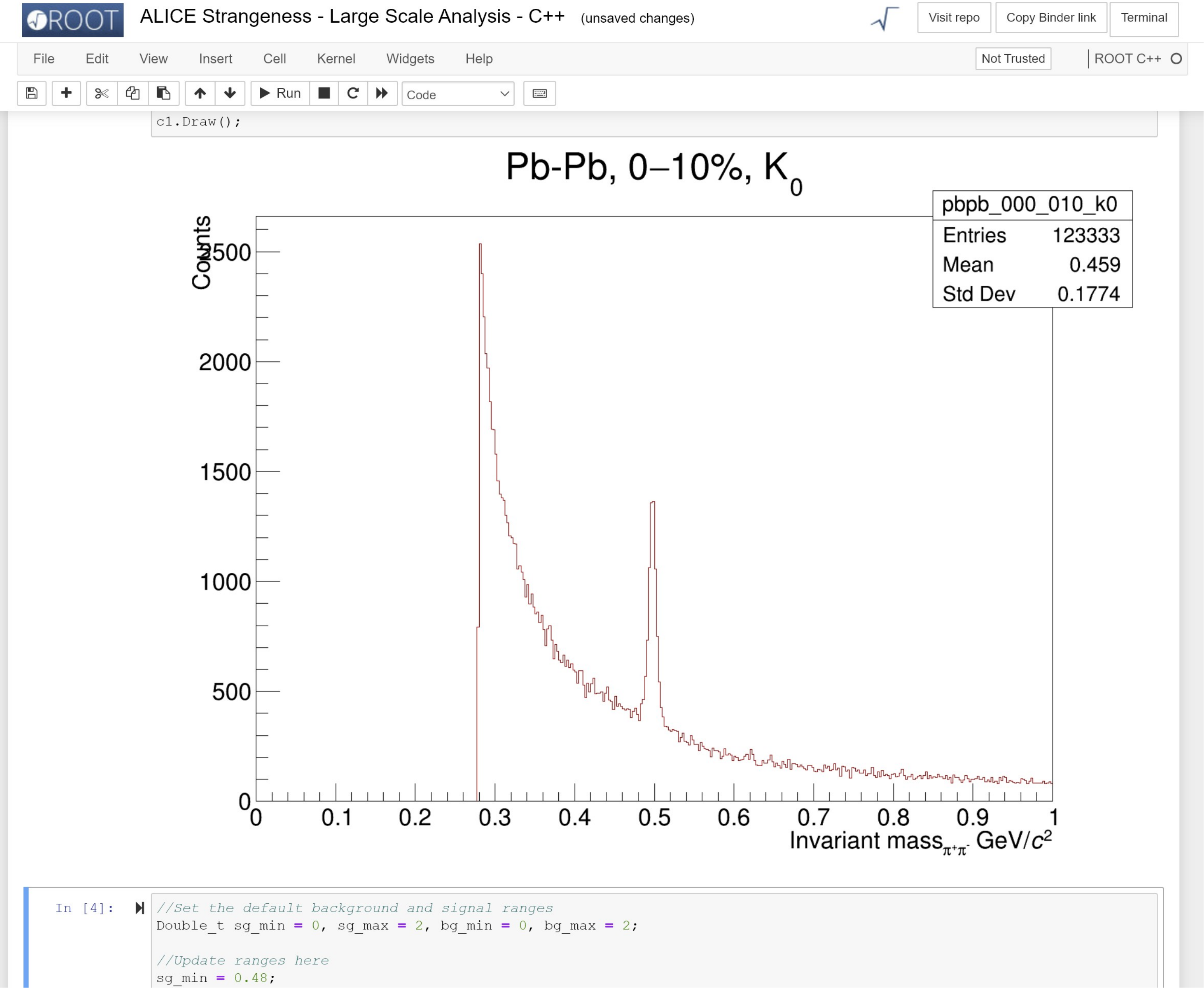
Presentation



```
//Find a folder with the particle type
auto type_key = file->FindKey(type);
auto obj = type_key->ReadObj();

if (!obj->IsA()->InheritsFrom(TDirectory::Class())) {
    Printf("Wrong directory!");
}
auto *dir = dynamic_cast<TDirectory*>(obj);

//Find the requested histogram
TString histogram_name;
TString type_lowercase = type;
type_lowercase.ToLower();
if(collision == "pbpb") {
    histogram_name = collision + "_" + centrality + "_" + type_lowercase;
} else {
    histogram_name = collision + "_" + type_lowercase;
}

auto dir_key = dir->FindKey(histogram_name);
obj = dir_key->ReadObj();
if (!obj->IsA()->InheritsFrom(TH1::Class())) {
    Printf("Not a histogram, but %s", obj->ClassName());
}
auto histogram = dynamic_cast<TH1*>(obj);
```

```
//Draw the histogram
TCanvas c1("","",1600,1200);
histogram->Draw();
c1.Draw();
```

Presentation



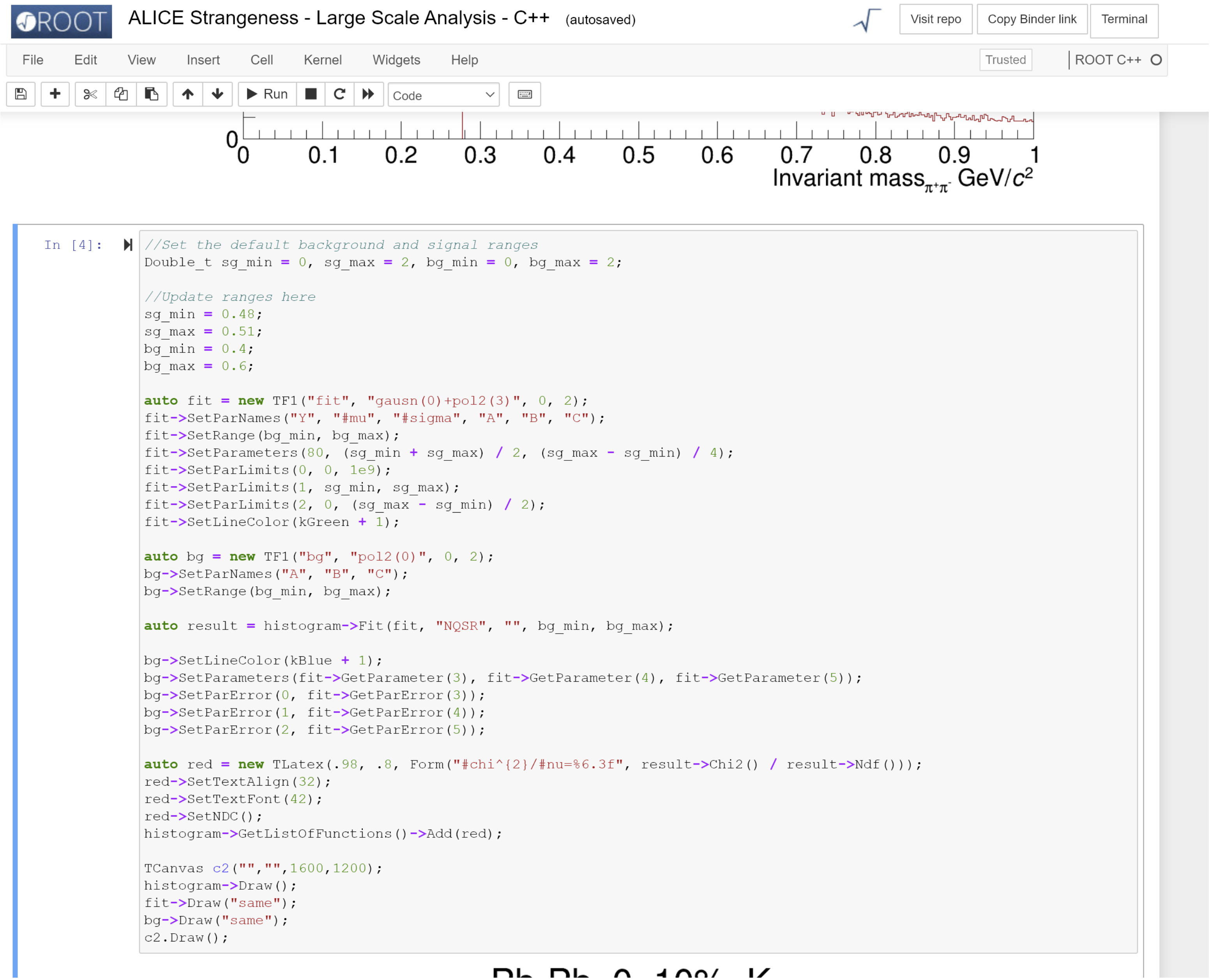
```
c1.Draw();
```

```
pbpb_000_010_k0
Entries      123333
Mean          0.459
Std Dev      0.1774
```

In [4]:

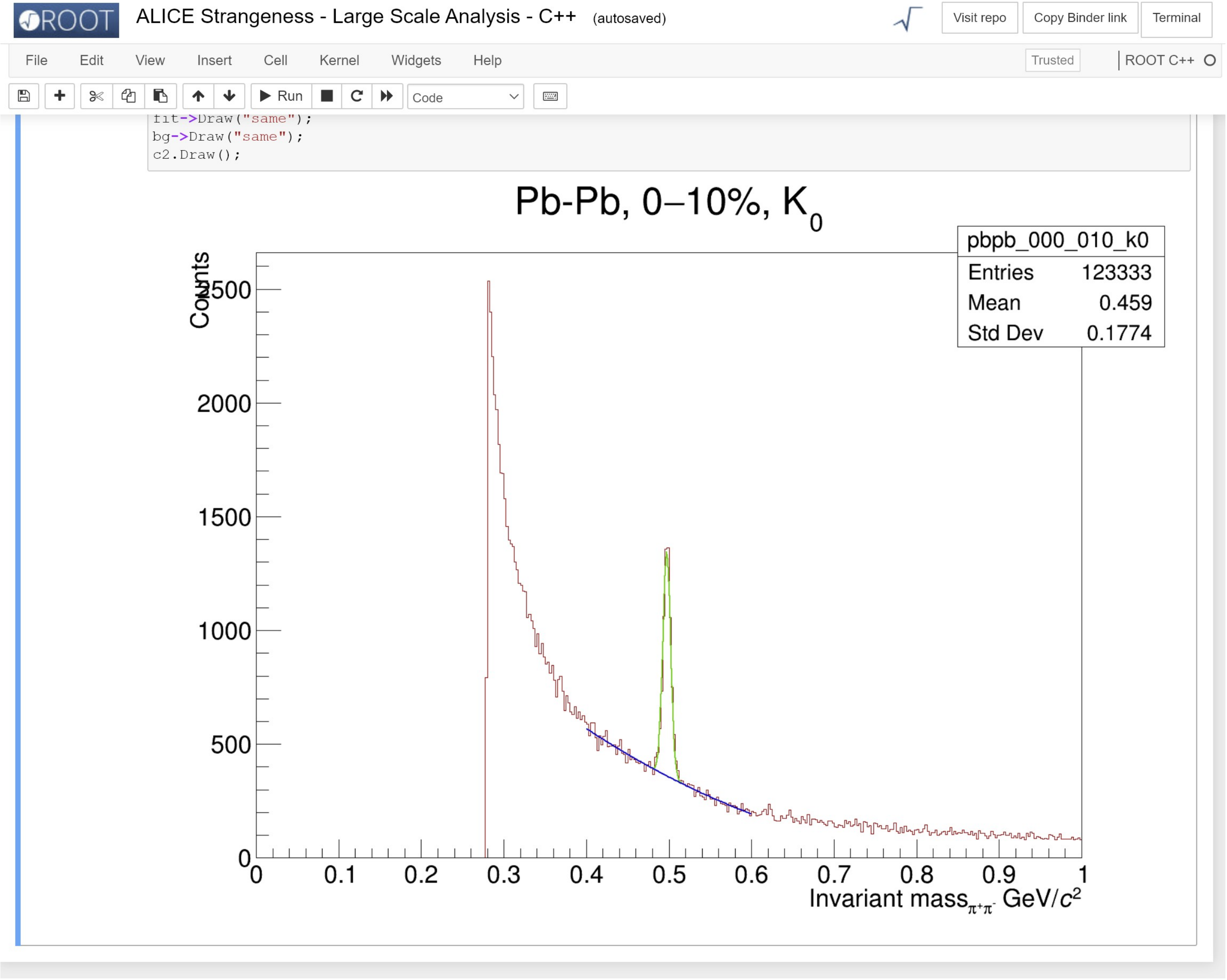
```
//Set the default background and signal ranges
Double_t sg_min = 0, sg_max = 2, bg_min = 0, bg_max = 2;

//Update ranges here
sg_min = 0.48;
```

Presentation



```
//Set the default background and signal ranges
Double_t sg_min = 0, sg_max = 2, bg_min = 0, bg_max = 2;

//Update ranges here
sg_min = 0.48;
sg_max = 0.51;
bg_min = 0.4;
bg_max = 0.6;

auto fit = new TF1("fit", "gausn(0)+pol2(3)", 0, 2);
fit->SetParNames("Y", "#mu", "#sigma", "A", "B", "C");
fit->SetRange(bg_min, bg_max);
fit->SetParameters(80, (sg_min + sg_max) / 2, (sg_max - sg_min) / 4);
fit->SetParLimits(0, 0, 1e9);
fit->SetParLimits(1, sg_min, sg_max);
fit->SetParLimits(2, 0, (sg_max - sg_min) / 2);
fit->SetLineColor(kGreen + 1);

auto bg = new TF1("bg", "pol2(0)", 0, 2);
bg->SetParNames("A", "B", "C");
bg->SetRange(bg_min, bg_max);

auto result = histogram->Fit(fit, "NQSR", "", bg_min, bg_max);

bg->SetLineColor(kBlue + 1);
bg->SetParameters(fit->GetParameter(3), fit->GetParameter(4), fit->GetParameter(5));
bg->SetParError(0, fit->GetParError(3));
bg->SetParError(1, fit->GetParError(4));
bg->SetParError(2, fit->GetParError(5));

auto red = new TLatex(.98, .8, Form("#chi^{2}/#nu=%6.3f", result->Chi2() / result->Ndf()));
red->SetTextAlign(32);
red->SetTextFont(42);
red->SetNDC();
histogram->GetListOfFunctions()->Add(red);

TCanvas c2("","",1600,1200);
histogram->Draw();
fit->Draw("same");
bg->Draw("same");
c2.Draw();
```

Presentation



```
fit->Draw("same");
bg->Draw("same");
c2.Draw();
```

Encountered problems

- **By default, graphics generated by ROOT in Jupyter are static images**
  - In theory it is possible to make them interactive by enabling JavaScript ROOT feature with magic line *“%%jsroot on”*
  - It doesn’t work in my case (some paths for JS scripts are invalid) – I think ROOT doesn’t expect to be run inside *mybinder.org*
  - Need to check this with experts, maybe I’m missing something simple like an additional config option
- **In Python version of Jupyter Notebook it is possible to create widgets (buttons, drop-down menus etc.) via code that appear in the Notebook**
  - This looks nice and prevents typos in e.g. selection of file for analysis
  - I have no idea if C++ ROOT also supports this feature

home.cern