Lecture2

July 21, 2016

1 How to make a *nice* histogram

1.1 Objectives:

- How can you compare histograms?
- What are *nice* histograms?
- How to make a *nice* histogram?

1.1.1 If everyone of you will be able to make a *nice* histogram after this lecture, I will be happy!

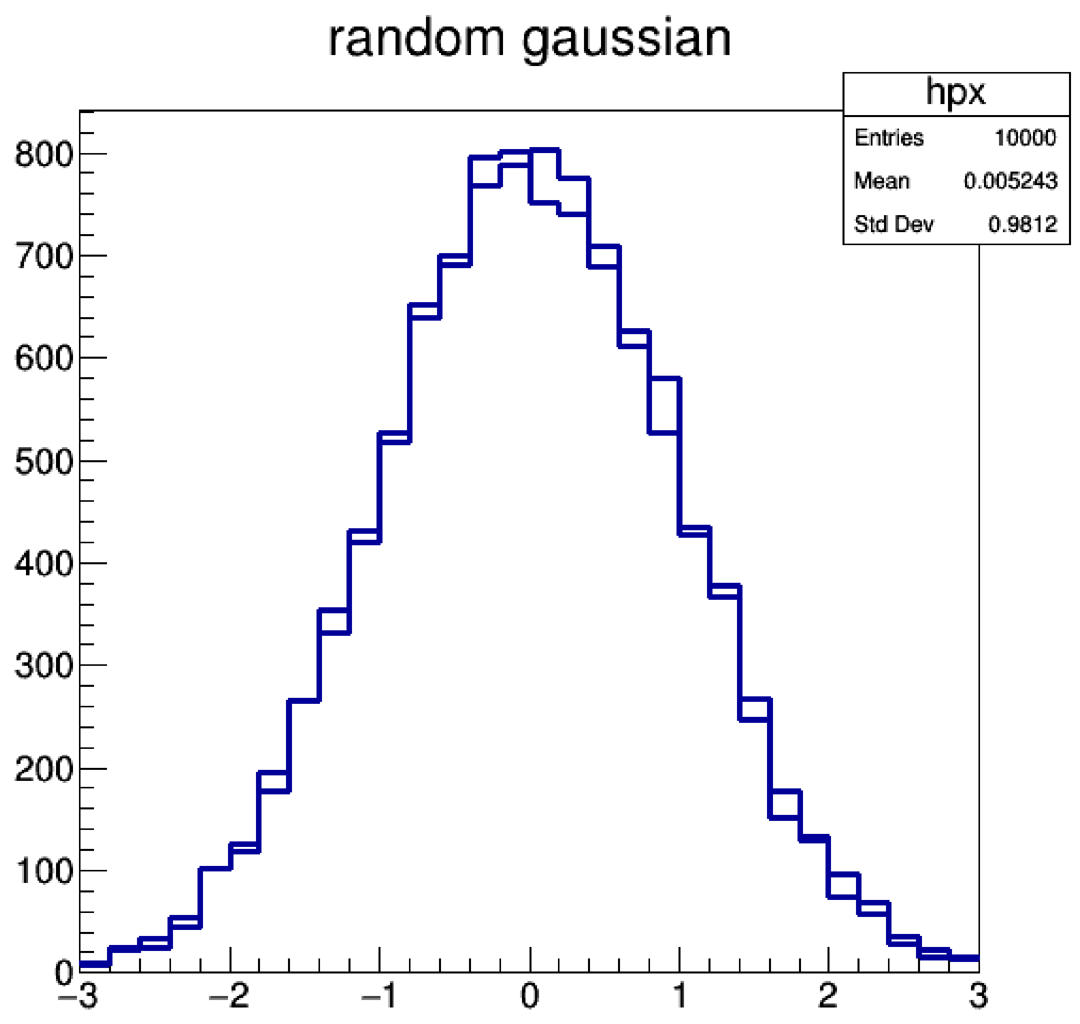
1.2 Compare histograms

Let's go back to our comparison of two randomly created Gaussian distributions:

```
In [1]: TCanvas c1("c1","Filling Example",600,600);
        TH1F hpx("hpx","random gaussian",30,-3,3);
        TH1F hpx2("hpx2","random gaussian",30,-3,3);
        TRandom3 r;
        int stats = 10000;
        Float_t px;
        for (Int_t i = 0; i < stats; i++) {
            double value = r.Gaus(0.0,1.0);
            hpx.Fill(value);
        }
        for (Int_t i = 0; i < stats; i++) {
            double value = r.Gaus(0.0,1.0);
            hpx2.Fill(value);
        }
```

And draw:

```
In [2]: hpx.Draw();
        hpx2.Draw("SAME");
        c1.Draw();
```

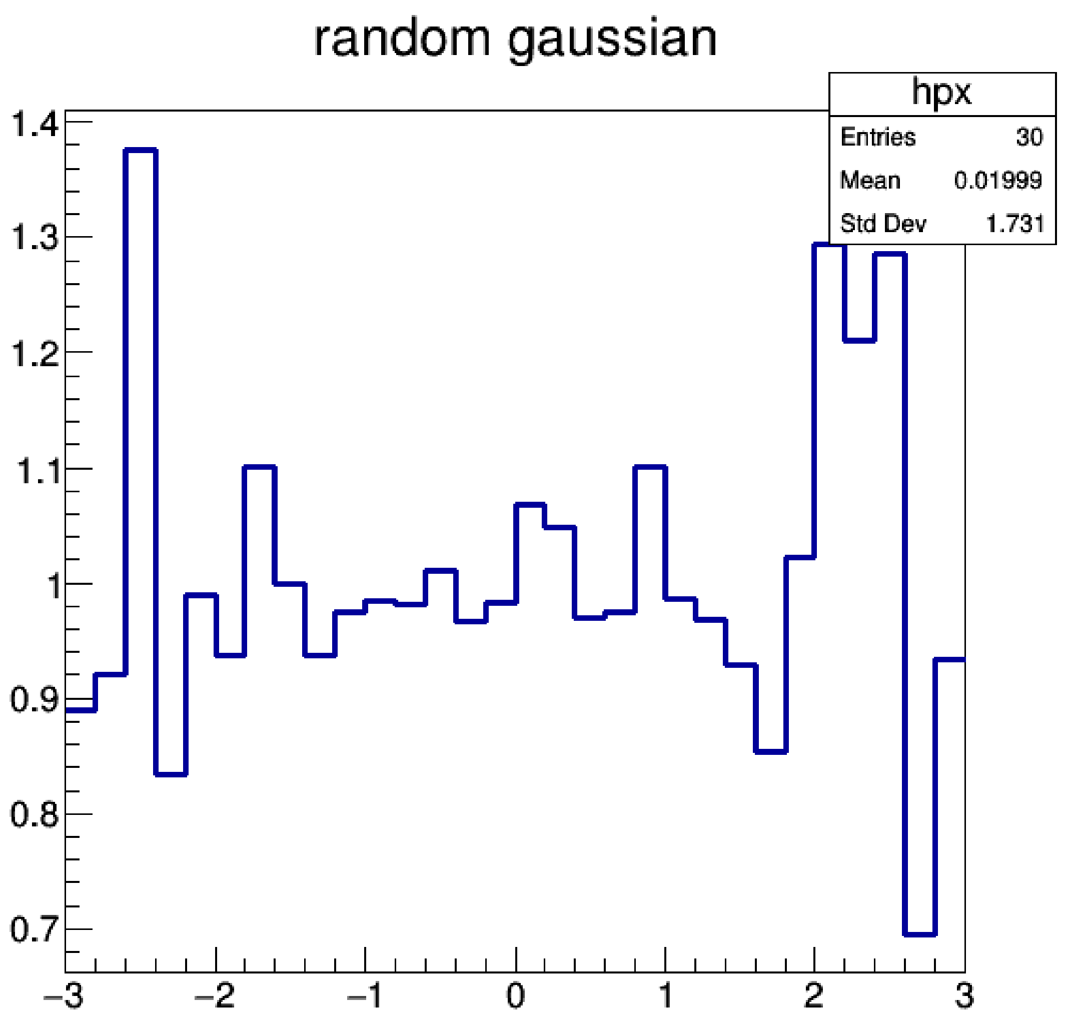


The distributions look different, but how can we compare the shapes? This is an often occuring task in HEP and there are **many ways** to do that. **One option** (not necessarily the best) is to calculate the ratio.
Let's create a new histogram, which is an exact copy of the first one:

```
In [3]: TH1F *hratio = (TH1F*)hpx.Clone();
```

Let's divide it by the second histogram.
Note that this function expects a pointer to a histogram `TH1*`, so we need to use the address of hpx2.
You can already see from this example, that it makes often sense to use pointer instead of objects.

```
In [4]: hratio->Divide(&hpx2);
        hratio->Draw();
        c1.Draw();
```

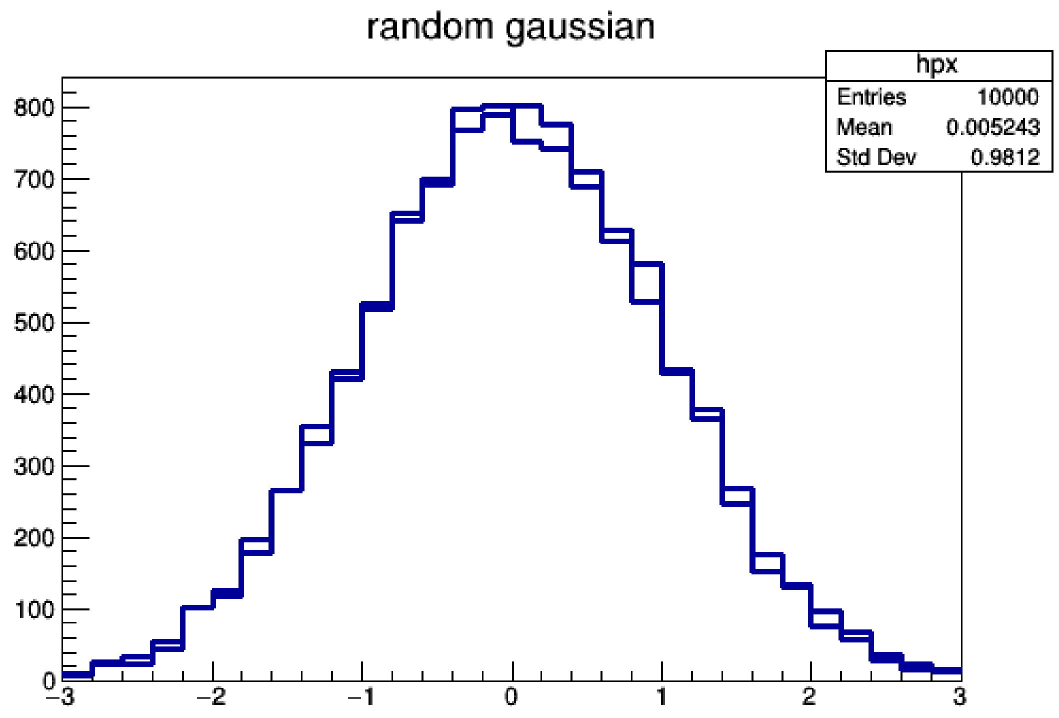


Can we also draw this comparison in the same canvas?
Yes, by dividing the canvas into different pads of different size:

```
In [5]: c1.Clear();
        TPad pad1("pad1","main",  0, 0.3, 1, 1.0);
        pad1.SetFillColor(0);
        pad1.Draw();

        TPad pad2("pad2","ratio", 0, 0.05, 1, 0.3);
        pad2.SetFillColor(0);
        pad2.Draw();
```

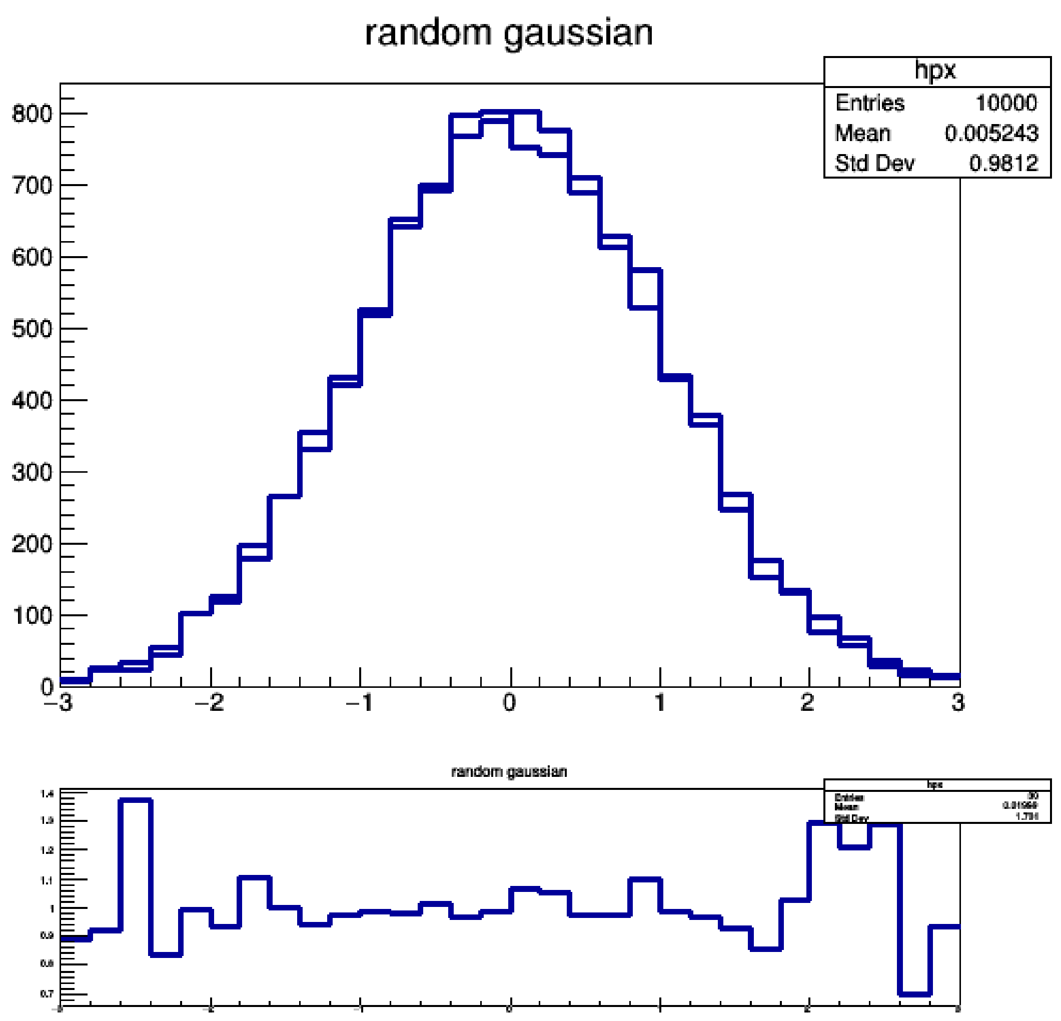
Go to the first pad and draw the two histograms:

```
In [6]: pad1.cd();
        hpx.Draw();
        hpx2.Draw("SAME");
        c1.Draw();
```



Now go to the second pad and draw the ratio:

```
In [7]: pad2.cd();
        hratio->Draw();
        c1.Draw();
```



1.2.1 Great, are you happy with this plot?

1.3 What is a nice plot?

- **It should contain all necessary information**
  - Axis titles and labels including units
  - Legend: What is shown and in which histogram/graph
  - Additional information about the data, experiment, selection, etc.
- **It should not contain unneccassary information**
  - Statistic box

  - Titles (most cases)
  - Anything not required to understand the plot

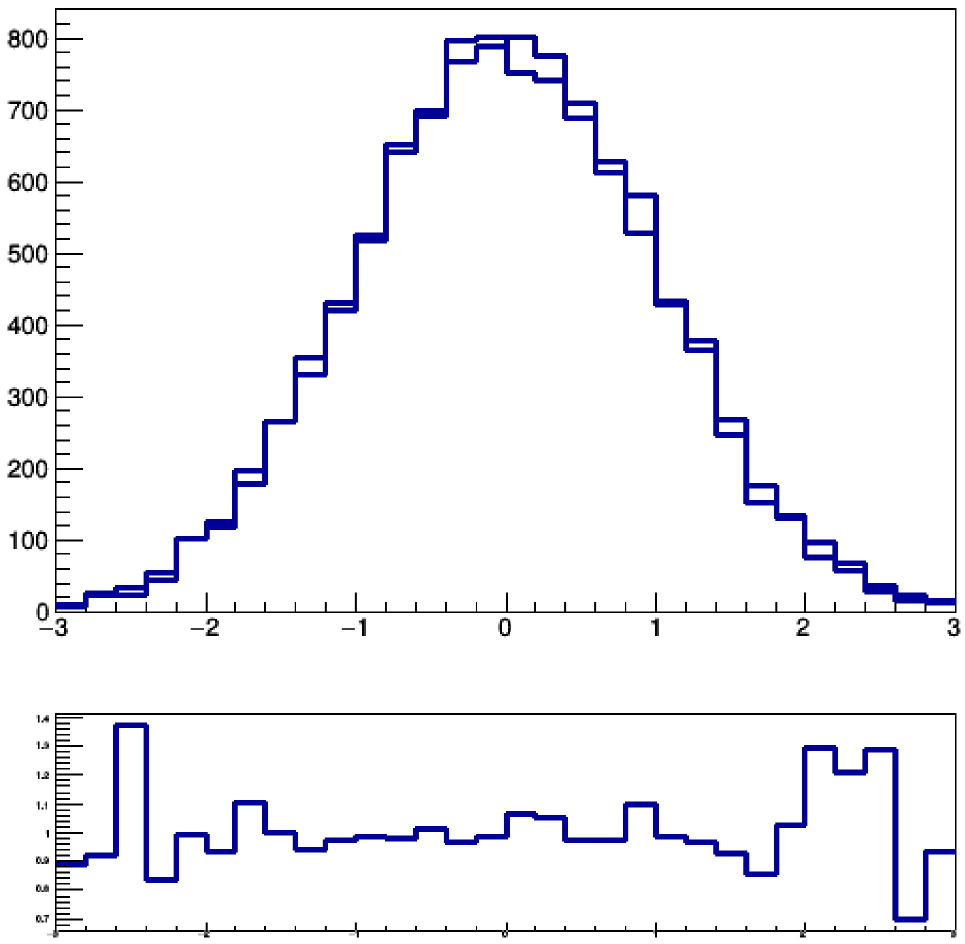
- **It should be easy to read**
  - Not too crowded, but complete
  - Each histogram/graph should be uniquely identifiable (also in black & white)
  - Sufficiently large text sizes for legend, axis label, axis title, etc

- **It should contain meaningful content**
  - Statistical & systematic uncertainties
  - Choose axis ranges properly (show everything, don't hide anything)
  - Data should make sense

1.3.1 This is not meant to be a complete list but a good orientation!

Let's make the plot *nicer* (not perfect)!
Start with easiest thing: Remove statistic box and histogram title globally!

```
In [8]: gStyle->SetOptStat(0);
        gStyle->SetOptTitle(0);
        c1.Draw();
```

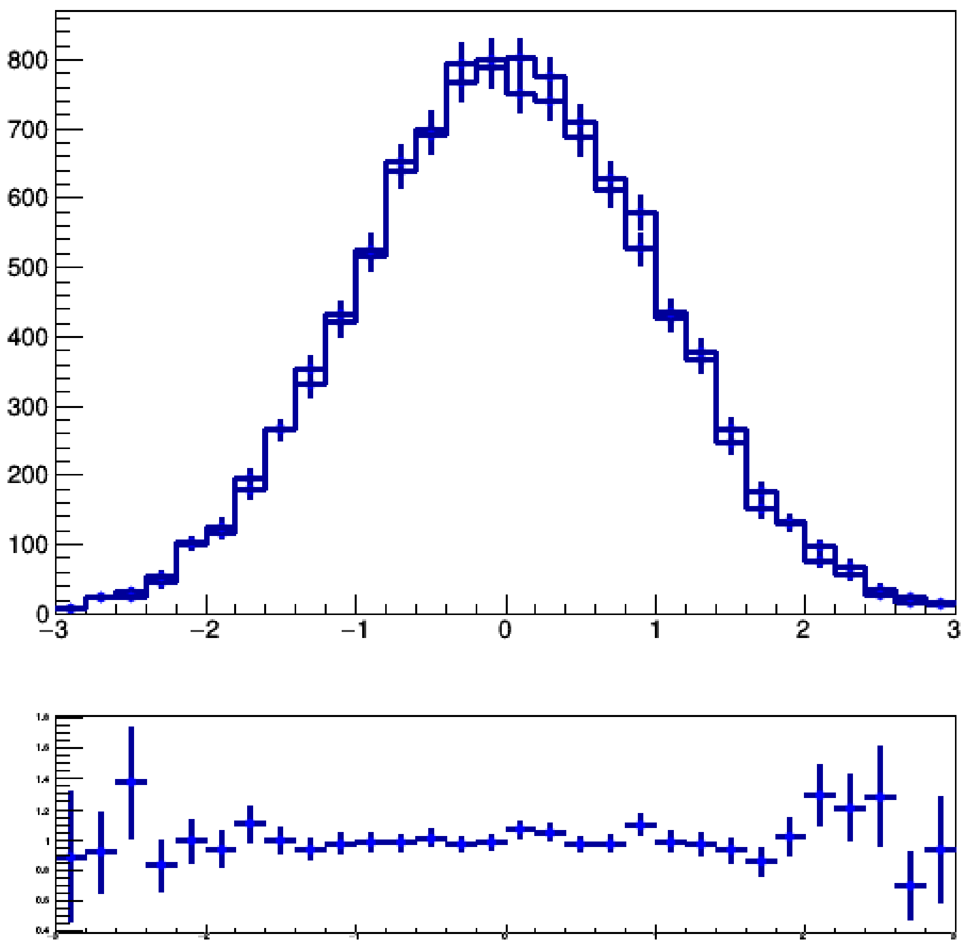


Store and show statistical uncertainties (sum of squares of weights):

```
In [9]: hpx.Sumw2();
        hpx2.Sumw2();
        pad1.cd();
        hpx.Draw("HE");
        hpx2.Draw("HESAME");
```

We need to recalculate the ratio including the weights for the error:

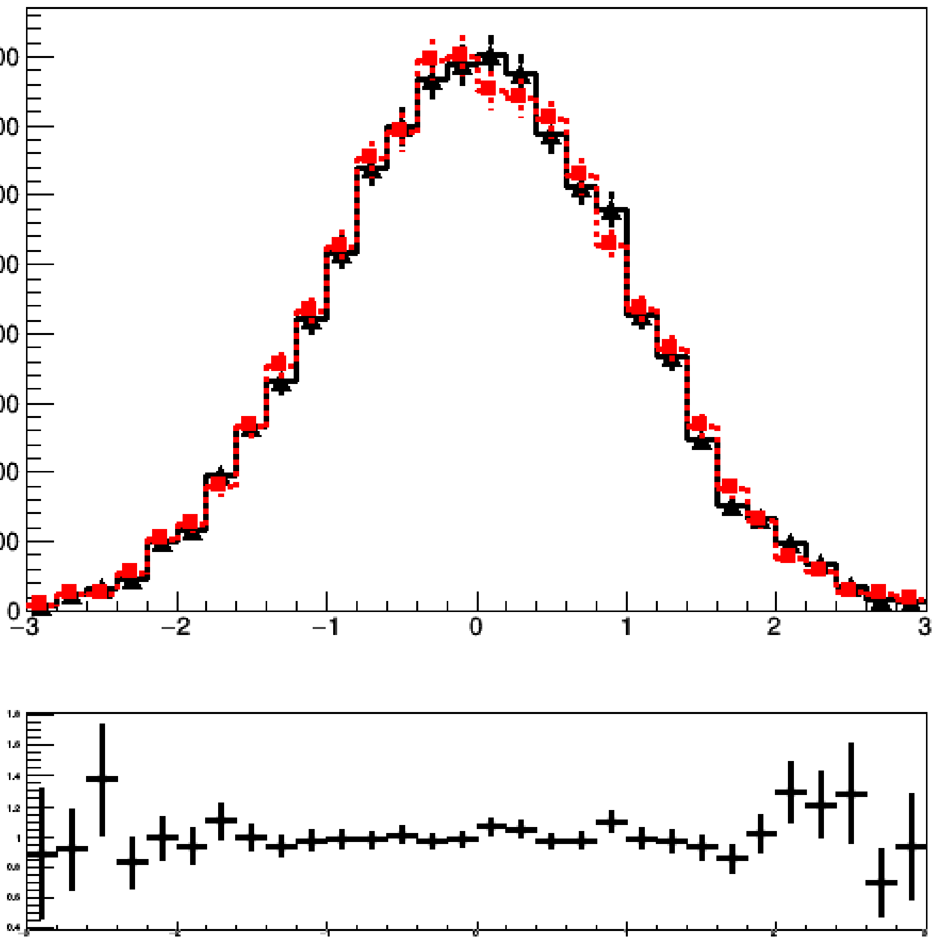
```
In [10]: hratio = (TH1F*)hpx.Clone();
         hratio->Divide(&hpx2);
         pad2.cd();
         hratio->Draw("EP");
         c1.Draw();
```



Make histograms unambiguous by using different color, line and marker style (black & white):

```
In [11]: hpx.SetLineColor(kBlack);
         hpx.SetMarkerColor(kBlack);
         hpx.SetMarkerStyle(22);
         hpx.SetMarkerSize(1.5);

         hpx2.SetLineColor(kRed);
         hpx2.SetLineStyle(2);
         hpx2.SetMarkerColor(kRed);
         hpx2.SetMarkerStyle(21);
         hpx2.SetMarkerSize(1.1);

         hratio->SetLineColor(kBlack);
         hratio->SetMarkerColor(kBlack);
```

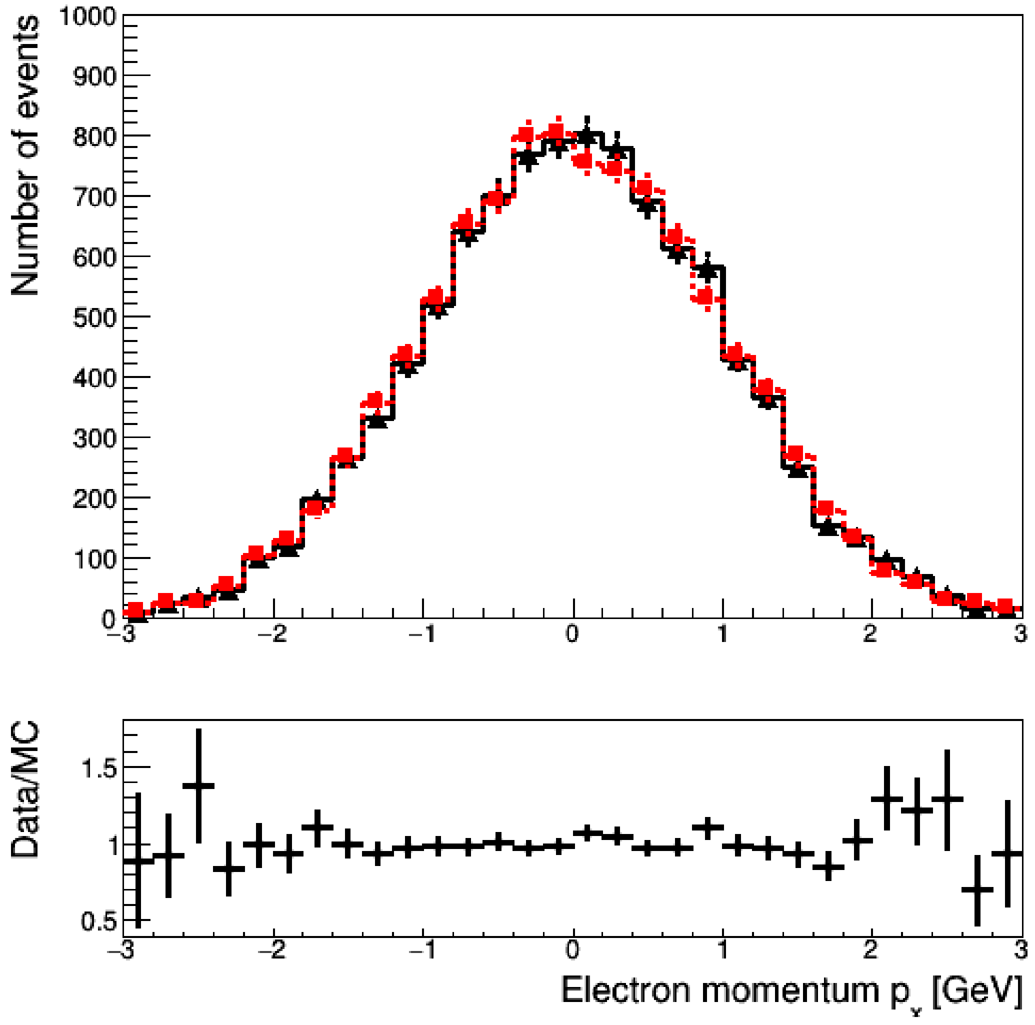
```
c1.Draw();
```



There are many other ways to make histograms/graphs distinguishable, the best to use depends on the plot!
Now add axis title and adjust the axis label and title font, size, offset:

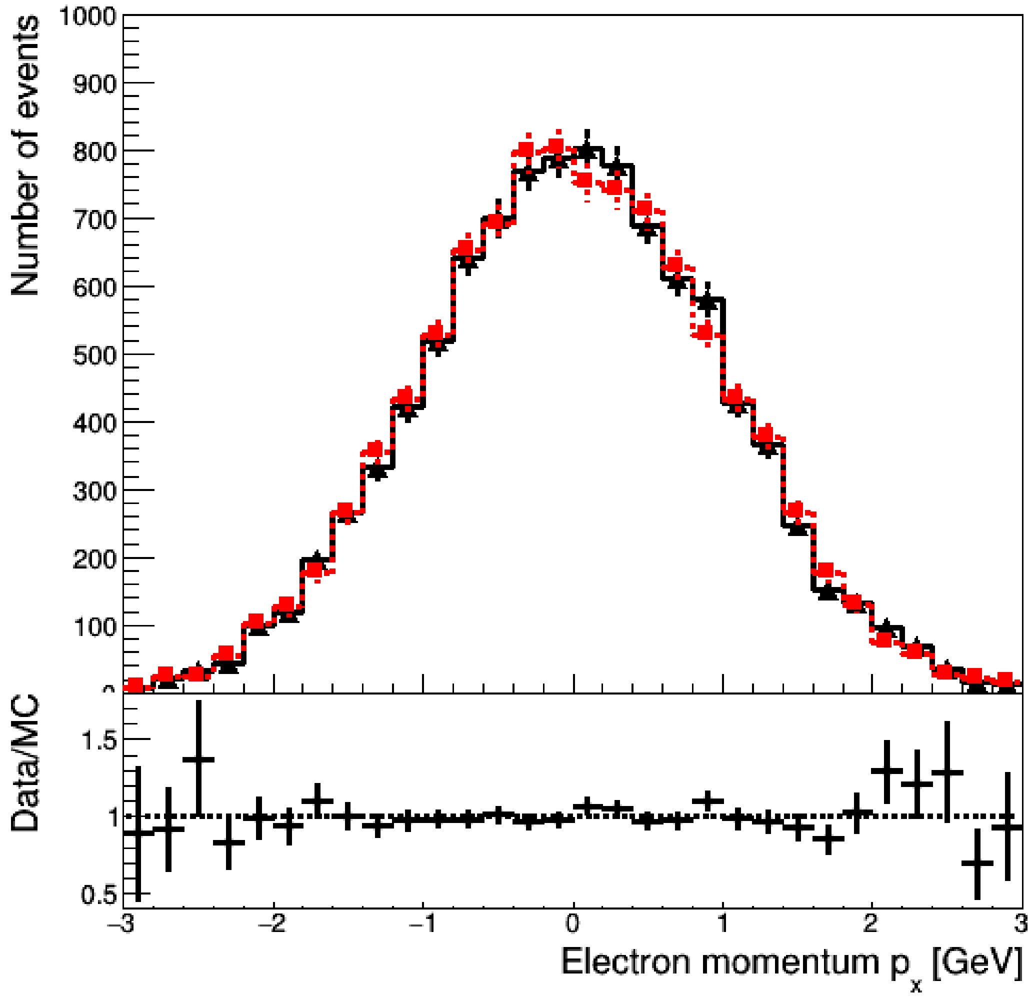
```
In [12]: hpx.GetYaxis()->SetTitle("Number of events");
         hpx.GetYaxis()->SetTitleSize(20);
         hpx.GetYaxis()->SetTitleFont(43);
         hpx.GetYaxis()->SetTitleOffset(1.55);
         hpx.GetYaxis()->SetLabelSize(15);
         hpx.GetYaxis()->SetLabelFont(43);
         hpx.GetYaxis()->SetRangeUser(0, 1000);
```

```
hratio->GetYaxis()->SetTitle("Data/MC");
hratio->GetYaxis()->SetNdivisions(505);
hratio->GetYaxis()->SetTitleSize(20);
hratio->GetYaxis()->SetTitleFont(43);
hratio->GetYaxis()->SetTitleOffset(1.55);
hratio->GetYaxis()->SetLabelSize(15);
hratio->GetYaxis()->SetLabelFont(43);
hratio->GetYaxis()->SetRangeUser(0.4, 1.8);

hratio->GetXaxis()->SetTitle("Electron momentum p_{x} [GeV]");
hratio->GetXaxis()->SetTitleSize(20);
hratio->GetXaxis()->SetTitleFont(43);
hratio->GetXaxis()->SetTitleOffset(4.);
hratio->GetXaxis()->SetLabelSize(15);
hratio->GetXaxis()->SetLabelFont(43);

c1.Draw();
```



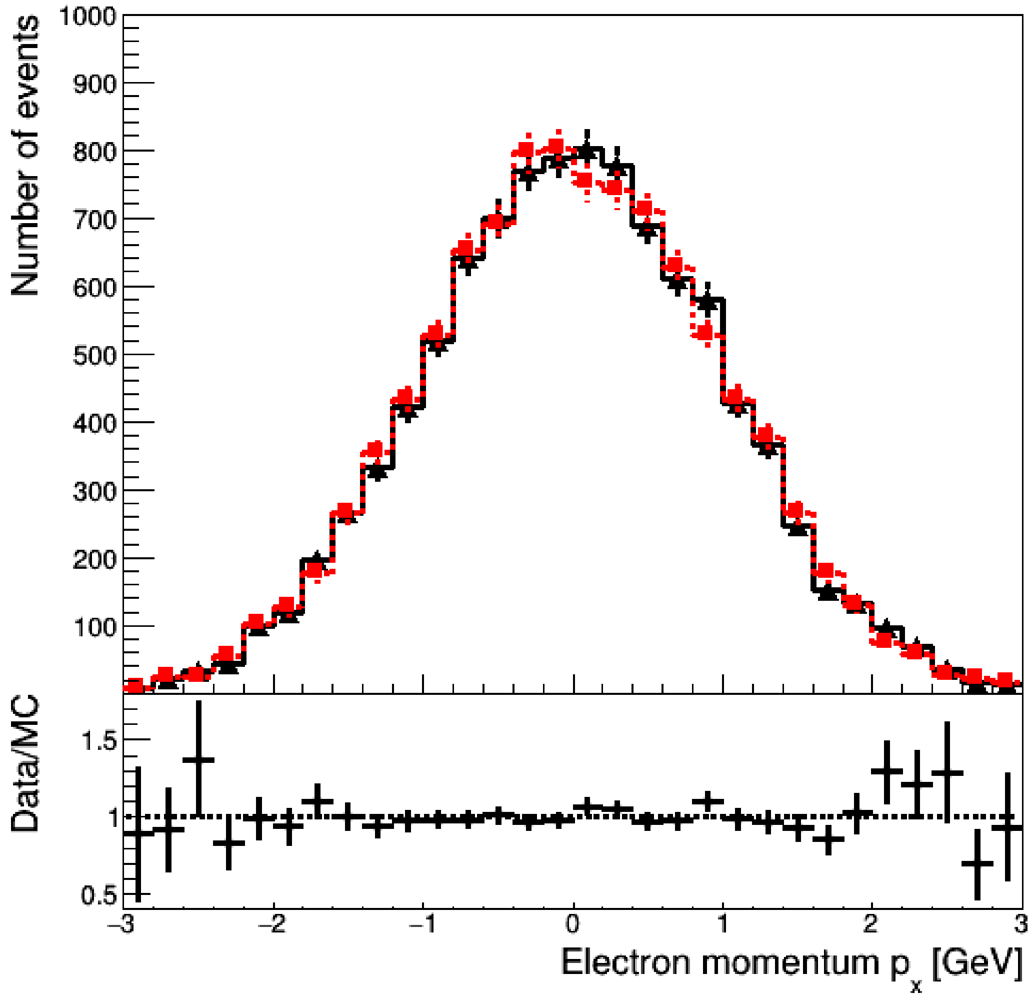
Connect lower pad with the upper pad and draw a vertical line in the ratio plot.

```
In [13]: pad1.SetBottomMargin(0);
         pad2.SetTopMargin(0);
         pad2.SetBottomMargin(0.2);

         TF1 one("one", "1", -3, 3);
         one.SetLineColor(kBlack);
         one.SetLineStyle(2);
         one.Draw("SAME");

         c1.Draw();
```



Now the first y-axis label 0 is clipped! Remove it and draw a new axis:

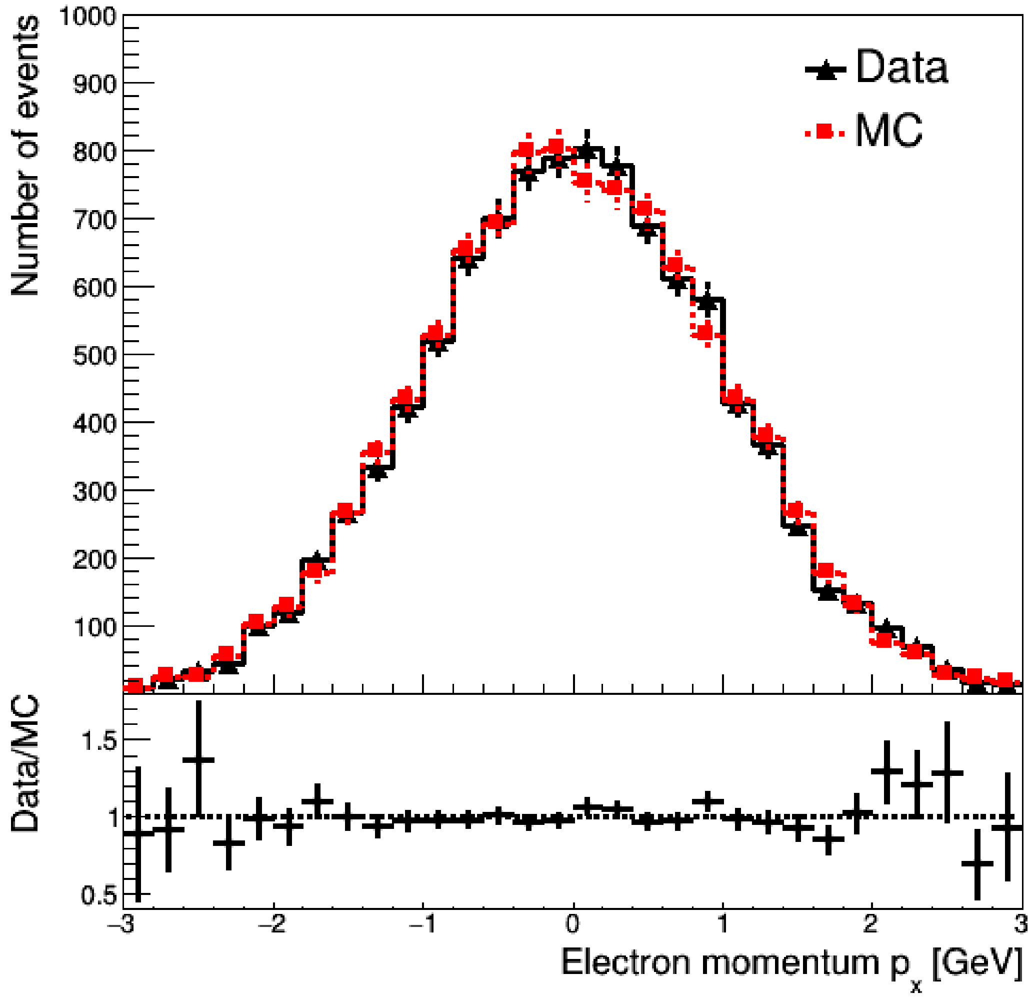
```
In [14]: hpx.GetYaxis()->SetLabelSize(0.);
         TGaxis *axis = new TGaxis( -3, 20, -3, 1000, 20,1000, 510,"");
         axis->SetLabelFont(43); // Absolute font size in pixel (precision 3)
         axis->SetLabelSize(15);
         pad1.cd();
         axis->Draw();
         c1.Draw();
```



Finally, we need to add a legend:

```
In [15]: TLegend *fLegend = new TLegend(0.7, 0.7, 0.9, 0.87);
         fLegend->AddEntry(&hpx,   "Data", "lp");
         fLegend->AddEntry(&hpx2,  "MC",   "lp");
         fLegend->SetFillColor(0);
         fLegend->SetLineColor(0);
```

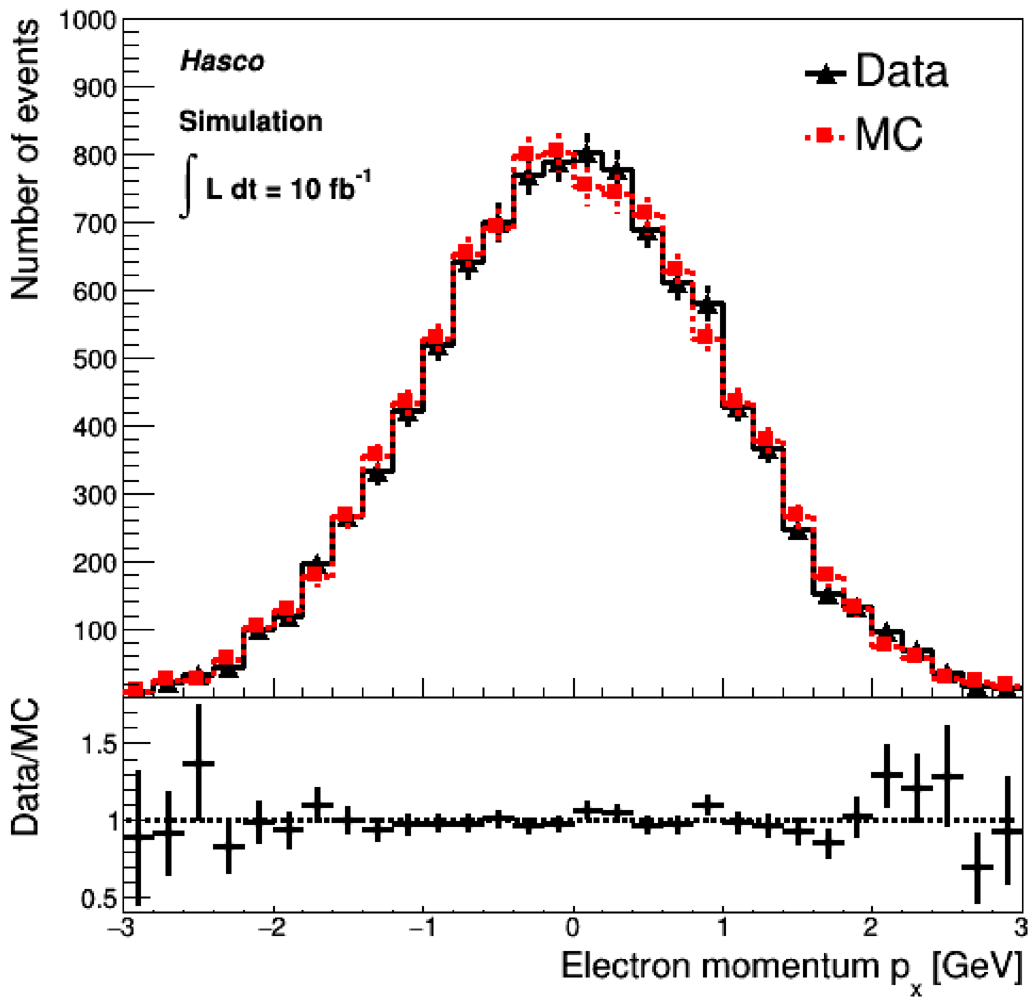
```
        fLegend->SetBorderSize(0);
        fLegend->SetTextFont(43);
        fLegend->SetTextSize(26);
        fLegend->Draw("SAME");
        c1.Draw();
```



And we can add latex labels for additional information:

```
In [16]: TLatex l1;
         l1.SetTextAlign(9);
         l1.SetTextSize(0.04);
         l1.SetNDC();
         l1.DrawLatex(0.15, 0.66, "#int L dt = 10 fb^{-1}");

         TLatex l2;
```

```
l2.SetTextAlign(9);
l2.SetTextFont(72);
l2.SetTextSize(0.04);
l2.SetNDC();
l2.DrawLatex(0.15, 0.83, "Hasco");

TLatex l3;
l3.SetTextAlign(9);
l3.SetTextSize(0.04);
l3.SetNDC();
l3.DrawLatex(0.15, 0.75, "Simulation");
c1.Draw();
```

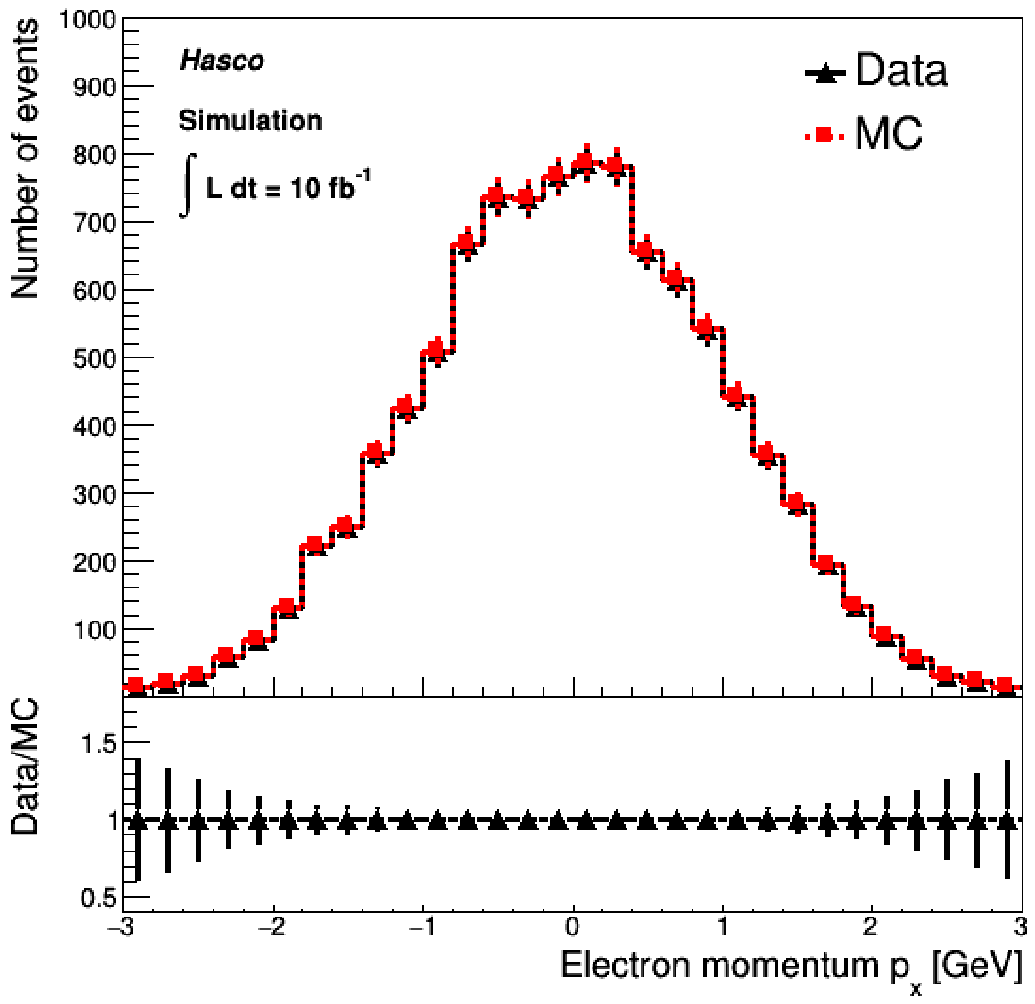


Ok, we are satisfied with the plot now, although you can argue that it is not yet perfect. Making really nice plots in ROOT can be very time consuming and often tricky, but it will go faster with growing experience!

1.4 Addendum: Repeatable random distributions

Let's compare our random distributions with the same seed, just to make sure, that they are really the same:

```
In [17]: //Reset histograms
         hpx.Reset();
         hpx2.Reset();

         //Set a seed=1234
         r.SetSeed(1234);

         //Loop
         for (Int_t i = 0; i < stats; i++) {
             double value = r.Gaus(0.0,1.0);
             hpx.Fill(value);
         }

         //Set a seed=1234
         r.SetSeed(1234);

         //Loop
         for (Int_t i = 0; i < stats; i++) {
             double value = r.Gaus(0.0,1.0);
             hpx2.Fill(value);
         }

         //Draw histogram
         pad1.cd();
         hpx.Draw("SAME");
         hpx2.Draw("SAME");

         //Draw ratio
         hratio->Reset();
         hratio = (TH1F*)hpx.Clone();
         hratio->Divide(&hpx2);
         pad2.cd();
         hratio->Draw("SAMEEP");
         c1.Draw();
```



Great, now we are sure that they are the same random distributions.