Drawing with ROOT, and few other thingies

May 9, 2022

In this second ROOT exercise we cover the following frequently used classes in ROOT:

1. TProfile (`http://root.cern.ch/root/html/TProfile.html`)
2. TGraphErrors (`http://root.cern.ch/root/html/TGraphErrors.html`)
3. TLegend (`http://root.cern.ch/root/html/TLegend.html`)
4. TCanvas (`http://root.cern.ch/root/html/TCanvas.html`)
5. TRandom (`https://root.cern.ch/root/html/TRandom.html`)
6. THistPainter (`http://root.cern.ch/root/html/THistPainter.html`)

Profile is essentially a special type of histogram, which in each bin holds the average value (the mean) of all entries filled in that bin, instead of their sum like it is done in an ordinary histogram. Next, TGraphErrors is ROOT's main plotting class, and it is regularly used nowadays to produce figures in the publications of all major collaborations at Large Hadron Collider (see for instance Fig. 3 in ALICE publication `http://arxiv.org/pdf/1011.3914v2.pdf`). The meaning of TLegend and TCanvas classes shall be self-evident, you will for instance see that TCanvas class comes very handy to save programmatically any graphics it holds as pdf, eps, png, etc., instead of clicking in ROOT GUI with the mouse.

Moreover, in this challenge, you will learn in a toy example how the results of your analysis have to be formatted before they can be made public. Predefined standards when it comes to formatting exist usually in each field, and it's an always good idea to conform to the rules from the beginning, otherwise weird things can happen... In high energy physics, in order to make the experimental results publicly available (mostly to theorists and other experiments), we use the Durham High-Energy Physics Database (HEPData) website (`https://www.hepdata.net/`), which hosts all the data points from official publications in a predefined format. In particular, the data points from publications at LHC you can fetch easily from this repository (Example: the data points for ALICE publication `http://arxiv.org/pdf/1011.3914v2.pdf` you can fetch via the direct link `https://www.hepdata.net/record/ins877822`). You can of course also use the search engine available on HEPData website to dig out the data points you are after. The start-up of this exercise is the same as in Exercise #1, so you can save some time here.

Consider the univariate function $f(x)$ defined by the following formula:

$$f(x) = axe^{-bx}\,, \tag{1}$$

where a and b are arbitrary constants.

Question 1: *Please implement a piece of the code in a standalone ROOT macro which can be run in interpreted mode (just start with the blank file, its content surround within {}, and save it as yourFile.C) which will treat the function $f(x)$ defined in Eq. (1) as a probability density function (p.d.f.) on the interval $0 < x < 1$, where x is variable and "a" and "b" are parameters (see examples in `http://root.cern.ch/root/html/TF1.html`).*

Question 2: *Set parameters "a" and "b" to be 0.1 and 0.4, respectively, and sample 50000 times variable x from the implemented p.d.f. $f(x)$.*

Question 3: *Define TProfile with 10 bins from 0. to 1., and fill it in the following (schematic) way:*

```
for(Int_t i=0;i<50000;i++)
{
 pointer-to-profile->Fill(gRandom->Uniform(1.),pointer-to-pdf-from-Question1->GetRandom());
}
```

In the above code snippet, `gRandom` is one of the global ROOT variables, and therefore it is available without explicit declaration in each ROOT session. It is very handy to use it in case you need only some basic random distributions (e.g. uniform, Gauss, etc., see `https://root.cern.ch/root/html/TRandom.html` for further details). In the above example, you are basically with `gRandom->Uniform(1.)` indicating that you want to fill all 10 profile's bins in its defined range [0.,1.) with the same frequency, nothing more nor less than that;

Question 4: *Please set the title of profile's x axis to be 'x', and of y axis to be 'y';*

Question 5: *Define markers of TProfile to be red full squares. When it comes to statistical errors associated with each marker, please draw them with a perpendicular line at each end, for aesthetic reasons (please do not try to achieve this manually, play instead with some predefined drawing options in* `http://root.cern.ch/root/html/THistPainter.html` *) for* `Draw()` *method. If you draw your profile at this point, you will see that markers are red, while the lines of statistical errors remained by default blue. You can correct for this by using:*

```
pointer-to-profile->SetLineColor(kRed);
```

Question 6: *What is the mean value of x in each of 10 bins (you may found TProfile's member function* `GetBinContent(bin-number)` *practical here)? The questions of this type justify the usage of TProfile, instead of histograms! Since the bin was selected randomly with* `gRandom->Uniform(1.)`*, you should see within statistical uncertainty about the same mean value in each bin;*

Question 7: *Please dump the content of your TProfile in the file named 'dataPoints.log', by using TProfile's member functions* `GetBinCenter(bin-number)`, `GetBinContent(bin-number)` *and* `GetBinError(bin-number)`. *Please structure the file 'dataPoints.log' in the following standard way which is characteristic for high-energy physics:*

```
x1          y1          yErr1
x2          y2          yErr2
.............................
x10         y10         yErr10
```

where x1 is the center of the 1st bin of TProfile, y1 is the content (value) of TProfile in the first bin, yErr1 is the stat. error of TProfile in the first bin, etc. For the above formatting, you may found `printf` *command very useful (*`http://www.cplusplus.com/reference/cstdio/printf/`*), or you can play with the construct* `cout<<Form(...)<<endl` *instead;*

Question 8: *Precision (i.e. number of significant digits). Please check the documentation of* `printf` *command, and set the precision of first column (x) to be 2 significant digits, while the precision of the second (y) and the third (yErr) column please set to 8 significant digits, when making the final version of 'dataPoints.log' file;*

Question 9: *Now you will finally learn about the ROOT class TGraphErrors. Please write a separate standalone ROOT plotting macro 'results.C' (or any other fancier name you can come up with) which will access directly the file 'dataPoints.log' that you have made in previous step, which will feed TGraphErrors object directly at construction, and which should be smart enough to recognize what each column corresponds to (see documentation of TGraphErrors how to achieve this, in particular, please have a look at* `https://root.cern.ch/doc/master/classTGraphErrors.html#a5f6ab98471c9e48337cbbedbbfad07e1` *);*

Question 10: *Please implement a canvas in the macro 'results.C' (see documentation of TCanvas), and plot TGraphErrors holding input from 'dataPoints.log' on this canvas. Set markers of TGraphErrors object to be blue open circles;*

Question 11: *Plot on the same canvas legend (see documentation and examples for TLegend) indicating that markers blue open circles correspond to the entry named 'dataPoints';*

Question 12: *Please save the resulting canvas holding TGraphErrors and TLegend explained above as figures in three frequently used formats: png, pdf and eps, and send me via email all three of them, together with the file 'dataPoints.log'. You can achieve this saving either 'by clicking', but also programmatically (this certainly shall be a preferred way!) ⇒ just see the documentation for TCanvas and SaveAs method!). Now you see why it is useful to define TCanvas object explicitly when drawing, and have a pointer to it available in the macro.*