Kirchhoff-Insitut für Physik
Physikalisches Insitut

Winter semester 2013-14
KIP CIP Pool (1.401)

Exercises for Statistical Methods in Particle Physics

http://www.physi.uni-heidelberg.de/~nberger/teaching/ws13/statistics/statistics.php

Dr. Niklaus Berger (nberger@physi.uni-heidelberg.de)
Dr. Oleg Brandt (obrandt@kip.uni-heidelberg.de)

Exercise 1: using ROOT

14. October 2013
Hand-in solutions by 14:00, 20. October 2013

Please send your solutions by email to obrandt@kip.uni-heidelberg.de until Sunday, 20 October 2013, 14:00, so that we can discuss them on the following Monday, 21 October 2013. Make sure that you use *SMIPP:Exercise01* as subject line.
While you are encouraged to discuss and work together, each student is required to work out his/her own solution, which he/she should be able to present and explain to the other students when they are discussed next week.
It is important that you test all programme code, macros, etc. before you hand them in.
All solutions, unless stated explicitly, can be handed in either in C++/ROOT or in PyROOT frameworks.

1 ROOT tutorial

Go through the basic `root` tutorial (Exercise 0) and make sure you understand all the steps, can run root at the CIP pool, etc. – otherwise ask.

2 ROOT documentation

Using the root documentation (`http://root.cern.ch/root/html528/ClassIndex.html`), find the difference between a TH1F and a TH1D histogram. When would you use which?

3 ROOT macro/PyROOT macro

Write a macro in ROOT/C++ that implements a function taking two numbers as arguments and printing their arithmetic and geometric mean, then returns the sum. In C++, output happens via

```
float mynumber = 17;
cout << "A string followed by " << mynumber << " a numeric value" << endl;
```

Even though the native ROOT compiler, CINT, does not explicitly require it, make a habit of including the appropriate header files and using statements as you would write fully C++ compatible code. Thus, for print statements as above, start with

```
#include <iostream>

using std::cout;
using std::endl;
```

or alternatively

```
using namespace std;
```

Note that we will abandon CINT and switch to fully C++ compatible code after some weeks of familiarisation with ROOT.
Write the same function in Python, but this time take a list of numbers as argument (the list is a standard type in Python, which you can fill with `l = [1,2,3,4]`, its length is `len(l)`; you iterate over a list with `for x in l:` followed by a newline). You will need the math module of Python, thus add

```
import math
```

For the n^{th} root needed in the geometric mean use `math.pow()` with a fractional power. Output in Python is via `print`.
(Attach the .C and .py file.)

4 ROOT histogram

Write a macro (either C++ or Python) creating a histogram with 10 bins in the range [0, 10]. Fill the histogram with integer numbers in the same range of $[0, 10]$, where each number n is filled n times, i.e. $n = 0$ is not filled at all, $n = 1$ one time, etc. What happens in the case of $n = 10$?
Create a linear function (here shown for the C++ case):

```
TF1* fun = new TF1("function","[0]+x*[1]",0,10);
```

the first argument is as usual the name of the object, the second the actual formula, with x being the independent variable and `[0], [1]` denoting free parameters, the last two arguments define the range. You can now define the free parameters by setting

```
fun->SetParameter(0,1.7);
fun->SetParameter(1,2.2);
```

and draw the function via

```
fun->Draw();
```

Note that you can display several histograms and functions on the same canvas if for all additional objects but the first you use `->Draw("SAME")`.
More interestingly, you can fit the function to a given histogram using

```
hist->fit(fun);
```

The fit parameters are printed in the terminal – are they what you expect? If not, how would you fix that problem? Do so... *Hint:* `root` will always use the centres of bins in fits.
(Attach a .C or .py file.)

5 Graphical presentation of your results

Take the fitted histogram from the previous exercise and make it nice-looking: add labels to the axes, display the fit results, show the fit as a dotted blue line and the histogram as round markers with error bars. Make sure to get rid of all grey backgrounds. You can find out more about the syntax for formatting of histograms from the `TAttLine, TAttFill,` and `TAttMarker` classes linked from `http://root.cern.ch/root/html528/TH1.html`. Save the result as a .png file.
(Attach the .png file and the root macro with your "beautification" commands.)