Exercise 1: Using `root`

N. Berger (nberger@physi.uni-heidelberg.de)

15.10.2012

Please send your solutions to nberger@physi.uni-heidelberg.de until 22. 10. 2012, 12:00. Put your answers in an email (subject line *SMIPP:Exercise01*) with macro files, root files and plots as mentioned in the attachments. Test macros and programs before sending them off...

1. **Root tutorial** Go through the basic `root` tutorial (Exercise 0) and make sure you understand all the steps - 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 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

   ```
   cout << "A string followed by " << 17 << " a number" << endl;
   ```

   Make a habit of including the appropriate header files and using statements as you would in compiled code, even if root is not very strict in these issues, thus start with

   ```
   #include <iostream>

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

   or alternatively

   ```
   using namespace std;
   ```

   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:`). 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 from 0 to 10 . Fill each histogram bin with its bin centre value. 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, the second the formula, with x being the independent variable and `[0], [1]` denoting free parameters, the last two parameters are the range. You can now set the parameters using

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

   and draw the function using

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

   More interestingly, you can fit the function to 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. **Presentation** Take the fitted histogram from the last 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. Save the result as a .png file.
   (Attach the .png file)