# More on histograms: TSpectrum, TStyle

Anna Simon

November 4, 2015

# Comments about the homework

- The canvas CAN be defined in a ”portrait orientation”!

```
TCanvas *c1 = new TCanvas("c1","vertical canvas",600,1200)
```

- Label your axis ...
- The default font size for axis titles and labels doesn't always work
- If your error bars are smaller than the data symbol... Congrats! But maybe make the symbols smaller (or different shape) to indicate the error bar is actually there
- Legend ...

GOALS for today:

- use TSpectrum to find the peaks in the histogram
- make the plots look better...

# TSpectrum

A class for finding peaks in spectra

```
void get_peaks(TH1F* hist, double xMin, double xMax){

//set the range of the histogram to analyze
   hist->GetXaxis()->SetRangeUser(xMin,xMax);

//create an object TSpectrum, specify the max number of peaks to find (here: 30)
   TSpectrum *s = new TSpectrum(30);

//search for peaks
//scan histogram hist, for peaks with sigma=2 and amplitude of 0.02 of the highest peak in the spectrum
//search returns the number of the peaks found
   Int_t nfound = s->Search(hist,2,"",0.02);

//get an array of the positions of the peaks
   Float_t *xpeaks = s->GetPositionX();
}
```

## Step 1

Use the above to find peaks in your HW histogram. Print out the positions of the peaks to the terminal.

# Sorting an array n ROOT

Class TMath has a `Sort(int n, const Element* a, Index* index, Bool_t down)` function!
Sort the n elements of the array a of generic templated type Element. In output the array index of type Index contains the indices of the sorted array. If down is false sort in increasing order (default is decreasing order).

```cpp
{
double array_size=5;
const Double_t array[5] = { 23, -1, 9999, 0, 4 };
const int index[5]={1,2,3,4,5};

TMath::Sort(array_size,array,index,0);

for(int i=0;i<5;i++){
cout << array[i] << " " << array[index[i]] << endl;
}
}
```

**Step 2**

Sort your npeak array to match the order of the peaks in the spectrum.

# TStyle

Now that the plots are ready, make them look decent:

- Add titles to the axis: `h1->GetXaxis()->SetTitle("E_#gamma [keV]")`
- Set the number of ticks on the axis to be more reasonable:
  `h1->GetXaxis()->SetNdivisions(n)`
  n = N1 + 100*N2 + 10000*N3
  N1=number of primary divisions.
  N2=number of secondary divisions.
  N3=number of 3rd divisions.
  e.g.:
  nndi=0 = no tick marks.
  nndi=2 = 2 divisions, one tick mark in the middle of the axis.
- Set the font sizes to be readable:
  `h1->GetXaxis()->SetTitleSize(x)`
  `h1->GetXaxis()->SetLabelSize(x)`
  x is the size of the text in fractions of the canvas size

# TStyle

- Set the line thickness to be visible: `h1->SetLineWidth(x)`
- Adjust the color `h1->SetLineColor(4)`
- Set the points size and style: `gr1->SetMarkerStyle(n); gr1->SetMarkerSize(x)`
  n = number of the style (see:
  https://root.cern.ch/doc/master/classTAttMarker.html)
  marker size x=1 is 8 pixels

Note:
the same functions have "Get" equivalents, to read current values of color, size, etc.
the same functions can be used for any object on the canvas: lines, cuts, textboxes, etc.