ROOT
Some Tips and Tricks

Manuel Calderon de la Barca Sanchez
UC Davis
STAR Analysis Meeting, 23/Aug/2011
Resources for ROOT

○ ROOT Web page:
  ● http://root.cern.ch/

○ User guides

○ Tutorials
  ● $ROOTSYS/tutorials/

○ This talk:
  ● Use some examples from tutorials
  ● Add some other "real world" examples
```c
TCanvas *c1 = new TCanvas("c1","The FillRandom example",200,10,700,900); //last 4 arguments: top x-coord of window, top y-coord of window, x width, y width

     c1->SetFillColor(18);

     pad1 = new TPad("pad1","The pad with the function",0.05,0.50,0.95,0.95,21);
     pad2 = new TPad("pad2","The pad with the histogram",0.05,0.05,0.95,0.45,21);

     pad1->Draw();
     pad2->Draw();
```

The Pad Constructor:

TPad(const char* name, const char* title, Double_t xlow, Double_t ylow, Double_t xup, Double_t yup, Color_t color = -1, Short_t bordersize = -1, Short_t bordermode = -2)
Result of Canvas and Pad creation

Canvas:
700 px wide, 900 px high

Pad 1:
Lower left corner:
5% of width from left edge
50% of height from low edge
Upper right corner:
95% of width from left edge
95% of height from low edge

Canvas Fill color: 18
Pad Fill color: 21
fillrandom.C : Drawing function

pad1->cd();
form1 = new TFormula("form1","abs(sin(x)/x)");
sqroot = new TF1("sqroot","x*gaus(0) + [3]*form1",0,10);
sqroot->SetParameters(10,4,1,20);
pad1->SetGridx();
pad1->SetGridy(); pad1->GetFrame()->SetFillColor(42);
pad1->GetFrame()->SetBorderMode(-1);
pad1->GetFrame()->SetBorderSize(5);
sqroot->SetLineColor(4);
sqroot->SetLineWidth(6);
sqroot->Draw();
lfunction = new TPaveLabel(5,39,9.8,46,"The sqroot function");
lfunction->SetFillColor(41);
lfunction->Draw();
c1->Update();
TFormula is drawn

Width of line is 2
Line Color 4 (blue)

Grids are drawn
both vertically and horizontally

TPaveLabel is drawn.
Fill Color is 41

Question: does the Frame have a different color than the Pad? Should it? What about the frame border?
After executing `fillrandom`, type the following lines at the command prompt:

```cpp
pad1->GetFrame()->SetFillColor(42);
pad1->GetFrame()->SetBorderMode(-1);
pad1->GetFrame()->SetBorderSize(5);
```
Fill a histogram randomly from TF1

```cpp
pad2->cd();
pad2->GetFrame()->SetFillColor(42);
pad2->GetFrame()->SetBorderMode(-1);
pad2->GetFrame()->SetBorderSize(5);

h1f = new TH1F("h1f","Test random numbers",200,0,10);

h1f->SetFillColor(45);

h1f->FillRandom("sqroot",10000);

h1f->Draw();

C1->Update();
```
Canvas after filling TH1

- Histogram is filled with 10K entries
- Stat box displays Entries, Mean, RMS
- Title is displayed
- TH1 Fill color : 45

- Note: Frame in pad2 did not change color, bordermode, bordersize
Changing Stat Box options

- Do not display the Stat Box
  - `gStyle->SetOptStat(0);`

- Things that can be displayed in Stat Box
  - Name, Entries, Mean, RMS, Underflow, Overflow, Integral, Skewness, Kurtosis.

- Traditional way of turning them on:
  - Each one is turned on by a bit, order as in previous bullet.
    - Name is LSB, Kurtosis is MSB.
  - Example: `gStyle->SetOptStat(111110110)`
    - Turns on all, except RMS and Name.

- But there is an updated way of turning them on ...
Changing StatBox options, updated

// The parameter mode can be any combination of
// kKsSiourRmMen
// k : kurtosis printed
// K : kurtosis and kurtosis error printed
// s : skewness printed
// S : skewness and skewness error printed
// i : integral of bins printed
// o : number of overflows printed
// u : number of underflows printed
// r : rms printed
// R : rms and rms error printed
// m : mean value printed
// M : mean value mean error values printed
// e : number of entries printed
// n : name of histogram is printed
Displaying all Stat Box Options

- gStyle->SetOptStat("kKsSiourRmMen");
- Rule of thumb: Don’t use it if you don’t have to.
  - Most useful stat box variables: entries, under-, overflows
Use gStyle and rootlogon.C

- gStyle can help you streamline your code
- Gives your plots a consistent look
- Use the rootlogon.C macro:
  - There are three levels of logon macros that will be executed: the system logon etc/system.rootlogon.C, the global user logon ~/./rootlogon.C and the local ././rootlogon.C.
  - For backward compatibility also the logon macro as specified by the Rint.Logon environment setting, by default ./rootlogon.C, will be executed.
  - No logon macros will be executed when the system is started with the -n option.
My own rootlogon.C

// rootlogon.C
// Manuel Calderon de la Barca
{
// Add my own options here:
TStyle* mcStyle = new TStyle("mcStyle","Manuel's Root Styles");
mcStyle->SetPalette(1,0); // avoid horrible default color scheme
mcStyle->SetOptStat(0);
mcStyle->SetOptTitle(0);
mcStyle->SetOptDate(0);
mcStyle->SetLabelSize(0.03,"xyz"); // size of axis value font
mcStyle->SetTitleSize(0.035,"xyz"); // size of axis title font
mcStyle->SetTitleFont(22,"xyz"); // font option
mcStyle->SetLabelFont(22,"xyz");
mcStyle->SetTitleOffset(1.2,"y");
// default canvas options
mcStyle->SetCanvasDefW(500);
mcStyle->SetCanvasDefH(500);
mcStyle->SetCanvasColor(0); // canvas...
mcStyle->SetCanvasBorderMode(0);
mcStyle->SetCanvasBorderSize(0);
mcStyle->SetPadBottomMargin(0.1); // margins...
mcStyle->SetPadTopMargin(0.1);
mcStyle->SetPadLeftMargin(0.1);
mcStyle->SetPadRightMargin(0.1);
mcStyle->SetPadGridX(0); // grids, tickmarks
mcStyle->SetPadGridY(0);
mcStyle->SetPadTickX(1);
mcStyle->SetPadTickY(1);
mcStyle->SetFrameBorderMode(0);
mcStyle->SetPaperSize(20,24); // US letter size
gROOT->SetStyle("mcStyle");
cout << "Styles are Set!" << endl;
return;
Example Plot, fillrandom, With Style!

- Canvas color, bordersize, bordermode: all set to 0.
- Fonts set to 22
- Change font size.
  - titles, labels
- Change y-title offset
- Histograms:
  - change line color
  - change fill color
  - change fill style
  - add titles
A real world example: $\Upsilon$ d$\sigma$/dy plot
Theory calculations

- **CEM model**
  - TGraphErrors
  - MarkerStyle 25
  - MarkerColor 4
  - MarkerSize 1.3
  - Draw("P")

- **CSM model**
  - TGraph
  - LineColor 4
  - LineWidth 3
  - LineStyle 2

\[ \gamma(1S+2S+3S), B_\gamma \text{ d} \sigma / \text{d}y (\text{pb}) \]

\[ p+p, \sqrt{s}=200 \text{ GeV} \]

\[ \gamma \rightarrow e^+e^- \]

\[ N_+ - 2\sqrt{N_+N_-} \]

NLO pQCD d\sigma/dy

- STAR, |y|<0.5
- CEM
- CSM
Drawing data

- Set axis titles
- SetMaximum(155)
- SetMinimum(-10)
- y TitleOffset 1.5
  - via GetYaxis

- STAR data
  - TGraphErrors
  - MarkerStyle 29
    - STAR!
  - Marker, Line Color 4
  - MarkerSize 3.5
Systematic uncertainty

TPave* StarUpsSys = new TPave(-0.1, CrossSectionAverage-SystUncLo*CrossSectionAverage, 0.1, CrossSectionAverage+SystUncHi*CrossSectionAverage, 1,"tbrl"); // last two options: border size, “top bottom right left”
Histogram of raw yield

- Opened from a different file
- Line, Fill Color 2
- FillStyle 3005
- Draw(“ehistsame”) • error bars and histogram
- Add lines to indicate y integration region
- TLine: Color 1, Width 3, Style 4.
Additional Axis on Right side

TGaxis* RawYieldAxis = new TGaxis(3.4,0,3.4,90,0,90,209,"+L");
//+ : draw on positive side
//L : left adjusted
RawYieldAxis->SetName("RawYieldAxis");
RawYieldAxis->(2);
RawYieldAxis->SetTextColor(2);
RawYieldAxis->SetTitle("Yield (8 < m_{ee} < 11 \text{ GeV/c}^2)"UBLISHES);
RawYieldAxis->SetLabelColor(2);
RawYieldAxis->Draw();

Use LaTeX syntax in titles and Legends

TLatex* ltx1 = new TLatex();
ltx1->DrawLatex(-3,130,"p+p, \#sqrt{s}=200 \text{ GeV}");
ltx1->DrawLatex(-3,120,"\#varUpsilon \rightarrow e^+e^-");

From dummy title:

";y_{ee};\#varUpsilon(1S+2S+3S); \#font[32]\{B\} \#upoint \ d\#sigma/dy (pb)"
double mysine(double* x, double* par) {
    double Amplitude = par[0];
    double wavelength = par[1];
    double phase = par[2];
    return Amplitude*sin(2*TMath::Pi()/wavelength*x[0]+phase);
}

void plotsine() {

    TCanvas* sineCanvas = new TCanvas("sineCanvas","A*sin(2pi/lambda*x + phi)",500,500);

    TF1* sineFunc = new TF1("sineFunc",&mysine,0,2*TMath::Pi(),3);
    sineFunc->SetParameters(2,TMath::Pi(),TMath::Pi()/2);
    sineFunc->Draw();
    return;
}
A more realistic example: Crystal Ball

\[ f(x; \alpha, n, \bar{x}, \sigma) = N \cdot \begin{cases} 
\exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right), & \text{for } \frac{x-\bar{x}}{\sigma} > -\alpha \\
A \cdot (B - \frac{x-\bar{x}}{\sigma})^{-n}, & \text{for } \frac{x-\bar{x}}{\sigma} \leq -\alpha 
\end{cases} \]

\[ A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{|\alpha|^2}{2}\right) \quad \quad B = \frac{n}{|\alpha|} - |\alpha| \]

Graph showing the Crystal Ball distribution for different values of \( \bar{x}, \sigma, n, \) and \( \alpha. \)
double CrystalBall(double* x, double* par) {
//http://en.wikipedia.org/wiki/Crystal_Ball_function
double xcur = x[0];
double alpha = par[0];
double n = par[1];
double mu = par[2];
double sigma = par[3];
double N = par[4];
TF1* exp = new TF1("exp","exp(x)",1e-20,1e20);
double A; double B;
if (alpha < 0) {
    A = pow((n/(-1*alpha)),n)*exp->Eval((-1)*alpha*alpha/2);
    B = n/(-1*alpha) + alpha;
} else {
    A = pow((n/alpha),n)*exp->Eval((-1)*alpha*alpha/2);
    B = n/alpha - alpha;
}
double f;
if ((xcur-mu)/sigma > (-1)*alpha)
    f = N*exp->Eval((-1)*(xcur-mu)*(xcur-mu)/(2*sigma*sigma));
else
    f = N*A*pow((B- (xcur-mu)/sigma),(-1*n));
delete exp;

return f;
Three-Crystal Balls Fitting STAR data

- Fit includes
  - 3 Crystal-Ball functions
  - Drell-Yan power law.
  - bottom quark power law.
MINUIT can obtain the \( \chi^2 \) contours from a multi parameter fit.

**Example**
- dielectron Invariant mass
- Components
  - Upsilonons
  - Drell-Yan
  - bottom-antibottom

For a tutorial see: $ROOTSYS/tutorials/fit/fitcont.C
Fit $\chi^2$ contours: relevant code snippet

Somewhere in the macro, set:

```cpp
TVirtualFitter::SetDefaultFitter("Minuit");
```

Fitting part:

```cpp
InvMass->Fit(FitFunc,"i","",5,16);
```

```cpp
gMinuit->SetErrorDef(4); // 2-sigma, argument is 2^2;
cout << "Getting 2-sigma contour" << endl;
TGraph* cont2sigma =
    (TGraph*) gMinuit->Contour(20,17,16);
cont2sigma->SetName("cont2sigma");
```

```cpp
gMinuit->SetErrorDef(1); // 1-sigma, argument is 1^2;
cout << "Getting 1-sigma contour" << endl;
TGraph* cont1sigma =
    (TGraph*) gMinuit->Contour(20,17,16);
cont1sigma->SetName("cont1sigma");
```
More control over colors

// Use of TColor::CreateGradientColorTable
void colorPalette() {
    // example of new colors (greys) and definition of a new palette
    const Int_t NRGBs = 5;
    const Int_t NCont = 256;

    Double_t stops[NRGBs] = { 0.00, 0.30, 0.61, 0.84, 1.00 };
    Double_t red[NRGBs]   = { 0.00, 0.00, 0.57, 0.90, 0.51 };
    Double_t green[NRGBs] = { 0.00, 0.65, 0.95, 0.20, 0.00 };
    Double_t blue[NRGBs]  = { 0.51, 0.55, 0.15, 0.00, 0.10 };
    TColor::CreateGradientColorTable(NRGBs, stops, red, green, blue, NCont);
    gStyle->SetNumberContours(NCont);

    TF2 *f2 = new TF2("f2",
        "exp(-(x^2) - (y^2))",-1.5,1.5,-1.5,1.5);
    //f2->SetContour(colNum);
f2->SetNpx(300);
f2->SetNpy(300);
f2->Draw("colz");
    return;
}
if filename is "", the file produced is padname.ps
if filename starts with a dot, the padname is added in front
if filename contains .eps, an Encapsulated Postscript file is produced
if filename contains .pdf, a PDF file is produced
if filename contains .svg, a SVG file is produced
if filename contains .gif, a GIF file is produced
if filename contains .gif+NN, an animated GIF file is produced
if filename contains .xpm, a XPM file is produced
if filename contains .png, a PNG file is produced
if filename contains .jpg, a JPEG file is produced
NOTE: JPEG's lossy compression will make all sharp edges fuzzy.
if filename contains .tiff, a TIFF file is produced
if filename contains .C or .cxx, a C++ macro file is produced
if filename contains .root, a Root file is produced
if filename contains .xml, a XML file is produced
Animated gifs

Rendering thousands canvases in a for loop
Use SaveAs("MSet.gif+10")
Obtain an animated gif after each iteration
Additional Material
ROOT commands

- Starting root, just type “root”
- At the root prompt:
  - `.q = Exit from root`
  - `.ls = list the files loaded into root session`
  - `!. some-unix-command = execute some-unix-command in the shell`
- Most c++ commands can also be interpreted.
- Executing a macro “myMacro.C”:
  - `.x myMacro.C`
ROOT Classes

- Since it is C++, everything is represented by classes:
  - Windows (or canvases) : TCanvas
    - A window where we can draw data, functions, etc.
  - Functions : TF1, TF2, TF3
    - Classes to manipulate mathematical functions, such as sin(x), in order to draw, evaluate, and integrate them.
  - Graphs : TGraph
    - Class used to plot data on a canvas
  - Histograms: TH1, TH2, TH3
    - Classes to manipulate histograms. Can draw them on a canvas, integrate them, obtain means and RMS values, evaluate bin contents.
  - Tutorials (lots of code to try out ROOT):
    - $ROOTSYS/tutorials/
    - For example: ./hsimple.C, and ./hist/h1draw.C
Graph Draw Options

The various draw options for a graph are explained in TGraph::PaintGraph. They are:

- "L" A simple poly-line between every points is drawn
- "F" A fill area is drawn
- "F1" Idem as "F" but fill area is no more repartee around X=0 or Y=0
- "F2" Draw a fill area poly line connecting the center of bins
- "A" Axis are drawn around the graph
- "C" A smooth curve is drawn
- "*" A star is plotted at each point
- "P" The current marker of the graph is plotted at each point
- "B" A bar chart is drawn at each point
- "[]" Only the end vertical/horizontal lines of the error bars are drawn. This option only applies to the TGraphAsymmErrors.
- "1" ylow = rwymin

The options are not case sensitive and they can be concatenated in most cases. Let us look at some examples
Text Fonts, Part 1

- [http://root.cern.ch/root/html530/TAttText.html#T5](http://root.cern.ch/root/html530/TAttText.html#T5)

- **Text font code = 10*fontnumber + precision**
  - Font numbers must be between 1 and 14.
  - The precision can be:
    - precision = 0 fast hardware fonts (steps in the size)
    - precision = 1 scalable and rotatable hardware fonts (see below)
    - precision = 2 scalable and rotatable hardware fonts
    - precision = 3 scalable and rotatable hardware fonts. Text size is given in pixels.
## List of the currently supported fonts

<table>
<thead>
<tr>
<th>Font number</th>
<th>X11 Names</th>
<th>Win32/TTF Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>times-medium-i-normal</td>
<td>&quot;Times New Roman&quot;</td>
</tr>
<tr>
<td>2</td>
<td>times-bold-r-normal</td>
<td>&quot;Times New Roman&quot;</td>
</tr>
<tr>
<td>3</td>
<td>times-bold-i-normal</td>
<td>&quot;Times New Roman&quot;</td>
</tr>
<tr>
<td>4</td>
<td>helvetica-medium-r-normal</td>
<td>&quot;Arial&quot;</td>
</tr>
<tr>
<td>5</td>
<td>helvetica-medium-o-normal</td>
<td>&quot;Arial&quot;</td>
</tr>
<tr>
<td>6</td>
<td>helvetica-bold-r-normal</td>
<td>&quot;Arial&quot;</td>
</tr>
<tr>
<td>7</td>
<td>helvetica-bold-o-normal</td>
<td>&quot;Arial&quot;</td>
</tr>
<tr>
<td>8</td>
<td>courier-medium-r-normal</td>
<td>&quot;Courier New&quot;</td>
</tr>
<tr>
<td>9</td>
<td>courier-medium-o-normal</td>
<td>&quot;Courier New&quot;</td>
</tr>
<tr>
<td>10</td>
<td>courier-bold-r-normal</td>
<td>&quot;Courier New&quot;</td>
</tr>
<tr>
<td>11</td>
<td>courier-bold-o-normal</td>
<td>&quot;Courier New&quot;</td>
</tr>
<tr>
<td>12</td>
<td>symbol-medium-r-normal</td>
<td>&quot;Symbol&quot;</td>
</tr>
<tr>
<td>13</td>
<td>times-medium-r-normal</td>
<td>&quot;Times New Roman&quot;</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>&quot;Wingdings&quot;</td>
</tr>
<tr>
<td>15</td>
<td>Symbol italic (derived from Symbol)</td>
<td></td>
</tr>
</tbody>
</table>
12 : ABCDEFGH abcdefgh 0123456789 @#$
22 : ABCDEFGH abcdefgh 0123456789 @#$
32 : ABCDEFGH abcdefgh 0123456789 @#$
42 : ABCDEFGH abcdefgh 0123456789 @#$
52 : ABCDEFGH abcdefgh 0123456789 @#$
62 : ABCDEFGH abcdefgh 0123456789 @#$
72 : ABCDEFGH abcdefgh 0123456789 @#$
82 : ABCDEFGH abcdefgh 0123456789 @#$
92 : ABCDEFGH abcdefgh 0123456789 @#$
102 : ABCDEFGH abcdefgh 0123456789 @#$
112 : ABCDEFGH abcdefgh 0123456789 @#$
122 : ABXΔΕΦΓΗ αβχδεφγη 0123456789 ΟΞ
132 : ABCDEFGH abcdefgh 0123456789 @#$
142 : ~ςτμξηνοπφρςτμξηνοπφρςτμξηνοπφρςτμξη
152 : ABXΔΕΦΓΗ αβχδεφγη 0123456789 ΟΞ
Colors

- FSee
  http://root.cern.ch/root/html530/TAttFill.html
- Default color palette
Using the Color Wheel

myObject.SetFillColor(kRed);
myObject.SetFillColor(kYellow-10);
myLine.SetLineColor(kMagenta+2);
2-D plot options : draw2dopt.C

- $ROOTSYS/tutorials/hist/draw2dopt.C
- See THistPainter:Paint for drawing options
- Example uses: gStyle->SetPalette(1,0);
2-d options, contours
2-d options, lego, surfpol

Note: option lego2 not displayed
2-d options, surface
TGaxis