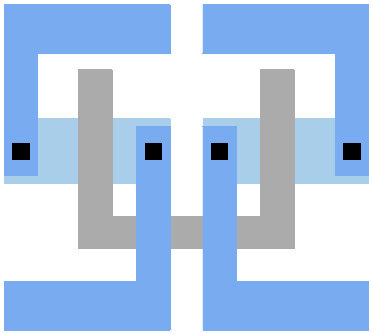
UNIVERSITÄT
HEIDELBERG

ROOT



Michael Ritzert

michael.ritzert@ziti.uni-heidelberg.de

Vorlesung „Tools“

05.02.2021

ROOT

- DAS Allround-Tool der Teilchenphysiker
- Plotting, Datenspeicherung, Datenanalyse (Histogramme).
- Interface über C++-Interpreter(!)
  - Auf der Kommandozeile: Relativ laxe Syntax.
  - ROOT-Skripte lassen sich auch compilieren.
- Hier wird nur ein WINZIGER Teil des Funktionsumfangs vorgestellt.
- https://root.cern/

erste Plots

- ROOT starten mit

```
> root
```

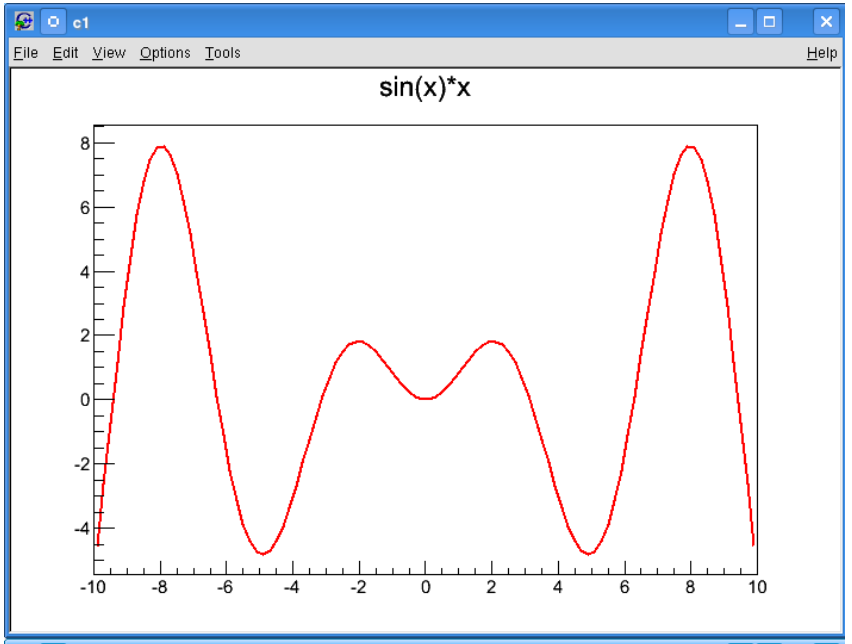
- Funktion definieren:

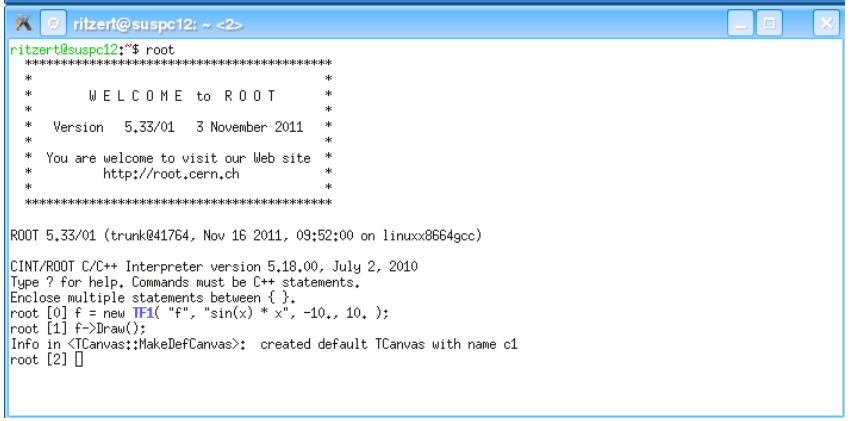
```
> f = new TF1( "f",
"sin(x) * x", -10., 10. );
```

Argumente:
  Name der Funktion
  Funktion
  min x, max x
Beachte: kein Typ für f angegeben.

- Funktion sich zeichnen lassen:

```
> f->Draw();
```





erste Plots II

- Mehrere Plots: Mehrere Funktionen definieren, jede Funktion mit `Draw()` zeichnen. Zweite und weitere Funktionen mit Option `"SAME"`. Farben manuell setzen über `g->SetLineColor( kGreen );`

- ```
  TF1* f = new TF1( "f", "sin(x)", -10., 10. );
  TF1* g = new TF1( "g", "cos(x)", -10., 10. );
  g->SetLineColor( kGreen );
  f->Draw();
  g->Draw( "SAME" );
  ```

- Weitere Optionen:
  - `"SAME"` superimpose on top of existing picture
  - `"L"` connect all computed points with a straight line
  - `"C"` connect all computed points with a smooth curve
  - `"FC"` draw a fill area below a smooth curve

  s.a. https://root.cern/doc/v620/classTF1.html#ab2c6b27e1558988dc7399408666588fa

- Mehrere Optionen einfach hintereinander schreiben.
  ```

Plots

- Plots sind nicht unbedingt einfach, aber der Funktionsumfang ist riesig.
  - Ableitungen, Integrale (numerisch)
  - Fits!
- Plots (und andere Objekte im Canvas) können angeklickt werden.
  - Zeichenstil ändern
  - Fits mit GUI ausführen
- Wichtige Funktion: Beenden von ROOT:
  `.q` oder `exit(0);`

Plots aus Datei

- Dateiformat: Zeilenweise Datensätze, mehrere Spalten
  - Option1: Mit *TGraph* (per Default nur 2 Spalten):
    ```
    g = new TGraph( "file.dat" );
    g->Draw( "ALP" );
    ```
    Optionen: https://root.cern/doc/v620/classTGraphPainter.html
- Option2: mit *TTree*:
  - Zuerst in einen TTree einlesen (Spaltennamen sp1, sp2, sp3)
    ```
    t = new TTree();
    t->ReadFile( "file.dat", "sp1:sp2:sp3" );
    ```
    (beliebige, sinnvolle(!) Namen für Spalten)
  - Dann die interessanten Spalten zeichnen. Format: y:x.
    ```
    t->Draw( "sp2:sp1", "", "line" );
    ```
    Rechnen in der Spaltenliste ist möglich.
    (2. Argument wird später erklärt)
    Optionen: https://root.cern/doc/v620/classTHistPainter.html
    (viele…).

Histogramme

- Übliche Anwendung von ROOT:
  Messdaten werden in .root-Dateien gespeichert und (offline) ausgewertet.
- .root-Datei: Kann alle ROOT-Objekte beinhalten. Funktionen, Histogramme, …, und **Tree**s.
- TTree: Liste von Einträgen gleichen (beliebig komplexen) Typs, üblicherweise Messdaten
- Sehr einfaches Erstellen von Histogrammen aus TTrees.
  - Statistiken über die Häufigkeit von Einträgen. 1D und 2D.
  - Verschiedenste Zeichenstile.
  - Filtern nach (komplexen) Kriterien.

Zu zeichnende Daten filtern

- ROOT kann Einträge aus einem Tree nach beliebigen Eigenschaften auswählen.
- Sehr wichtige Funktion, da die wenigstens Daten in der Reinform sofort zu verwenden sind.
- Das Kriterium (der „Cut“) wird der `Draw()`-Funktion von TTree als 2. Argument übergeben.

```
t->Draw( "sp2:sp1", "sp3>42", "line" );
```

- C++-Syntax. Also z.B. und-Verknüpfung:

```
"sp3>42 && sin(sp1/sp2)<0"
```

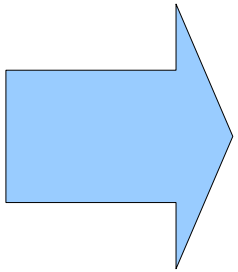
Histogramm aus Tree berechnen

- Häufigkeitsverteilung eines Eintrags als Histogramm darstellen.

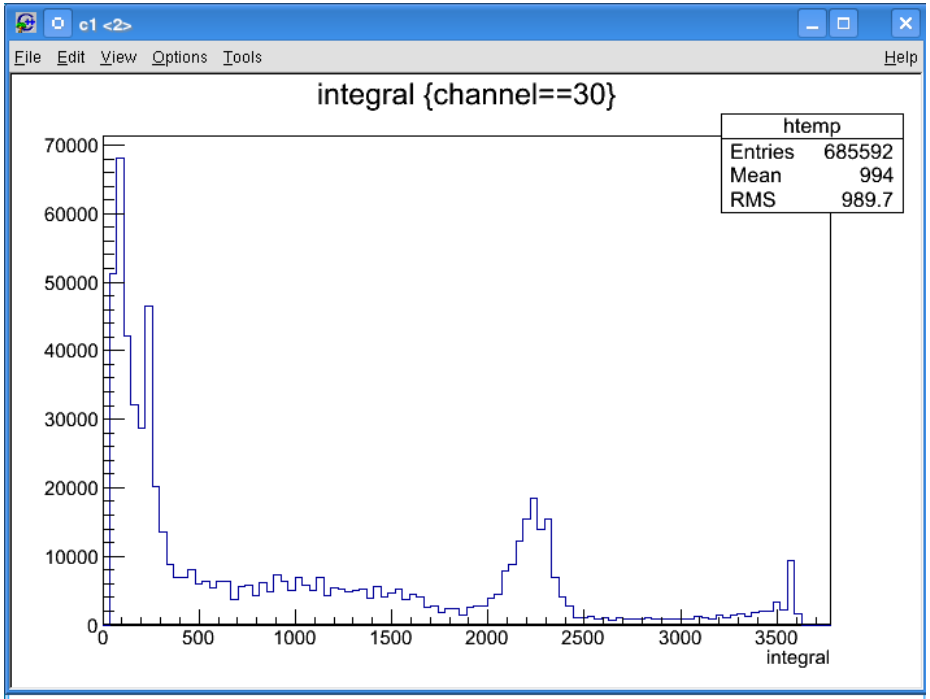
  `T->Draw( "Spalte", "Bedingung" );`

- Ergebnis ist ein neues Histogramm mit Default-Namen `htemp`.

a
1
3
2
1
1
3
1
2
2



x	y
1	4
2	3
3	2



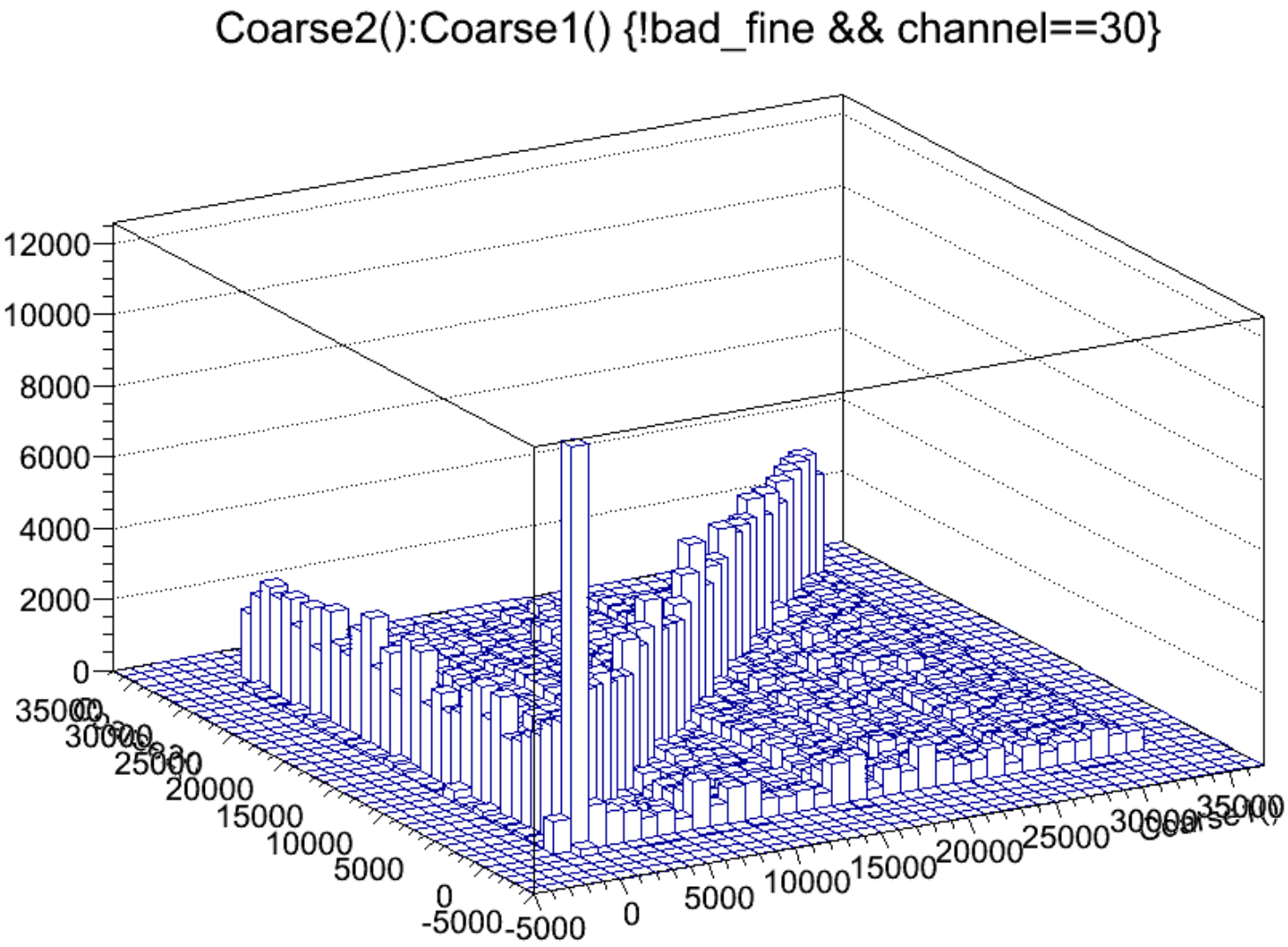

```
 ****************************************
 *                                      *
 *       W E L C O M E  to  R O O T     *
 *                                      *
 *   Version   5.33/01   3 November 2011*
 *                                      *
 *  You are welcome to visit our Web site*
 *         http://root.cern.ch          *
 *                                      *
 ****************************************

ROOT 5.33/01 (trunk@41764, Nov 24 2011, 12:27:00 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
Attaching file 2011-10-20-17:57:01.337.root as _file0...
root [1] Measurement->Draw( "integral", "channel==30" );
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c
root [2]
```

3D-Histogramme

- Einfach mehrere Spalten angeben und passende Plotoption auswählen, z.B. LEGO oder COL2.



Histogramme beschriften, etc.

- Binning vorgeben, z.B. 10 Bins 0…9:
  zuerst ein passendes Histogramm-Objekt anlegen:
  ```
  myh = new TH1I("myh", "title", 10, -0.5, 9.5);
  ```
  (oder TH2I). Dann das Histogramm mit den Daten füllen:
  ```
  t->Draw("spalte >>myh", …);
  ```
  oder ohne gleich zu zeichnen:
  ```
  t->Project("myh", "spalte", …);
  …;
  myh->Draw();
  ```
- Achsenbeschriftungen ändern:
  ```
  myh->SetXTitle( "Text" );
  ```
- Überschrift ändern:
  ```
  myh->SetTitle( "Text" );
  ```
- Farben ändern:
  ```
  myh->SetLineColor( kRed );
  myh->SetFillColor( kBlue );
  ```

Fits

- Einfachste Möglichkeit: Über Rechtsklick auf ein Histogramm und „Fit-Panel“.
- Zu fittende Funktion auswählen aus vordefinierter Liste,
  z.B. „`gaus`“ (ohne Parameter!),
  oder selbst eingeben: Parameter `[0]`, `[1]`, etc.
  Gerade a*x+b: `[1]*x + [ß]` (oder `pol1`)
- Fit-Ergebnis im Plot anzeigen:
  Options→Fit Parameters im Menü.
- Auf der Kommandozeile:
  zunächst eine Funktion (TF1) definieren.
  Dann den Fit durchführen:
  `o->Fit( "Funktion" );`
  (o kann ein Histogramm oder ein TGraph sein).

Skripte

- ROOT kann Skripte verarbeiten.
- Skripte sind kurze C++-Codestücke in .cpp-Dateien.
  Der Code ist komplett mit `{}` zu umschließen.

```
{
ehochx = new TF1( "ehochx", "exp(x)", 0, 10 );
ehochx->Draw();
}
```

- Wird ROOT mit dem Namen der Datei auf der Kommandozeile gestartet, so wird der Inhalt sofort zu Beginn der Sitzung ausgeführt.
  Um ROOT nach Ausführen der Datei automatisch zu beenden Kommandozeilenoption `-q` verwenden.
- Innerhalb einer Sitzung kann die Datei mit `.x "Dateiname"` ausgeführt werden.

.root-Dateien lesen

- Stehen Daten schon im .root-Format zur Verfügung, kann root einfach mit `root Datei.root` gestartet werden.
- Alle Objekte in der Datei stehen dann sofort mit Ihrem Namen zur Verfügung.
- Weiß man nicht, was in der Datei ist, hilft `.ls` oder der TBrowser weiter. Starten mit `new TBrowser();`
- Unter „ROOT Files“ findet man dann die Datei und kann browsen.
- Für Trees gibt es im Kontextmenü (rechte Maustaste) die Option „Start Viewer“, die ein mächtiges Tool zum analysieren des Trees (zusammenklicken von Histogrammen inkl. Cuts etc.) zur Verfügung stellt. (Ausführen: Symbol unten links)
- Der Viewer lässt sich auch direkt für jeden Tree starten, egal wie er erstellt wurde.

```
tree->StartViewer();
```

Plots speichern

- Auch ROOT kann Plots in vielen Vektor- und Bitmap-Formaten speichern.
- Allerdings mit weniger Einstellmöglichkeiten als gnuplot.
- Zum Speichern im Menü File→Save As… auswählen
- oder zunächst einen Canvas selbst anlegen:

  ```
  c = new TCanvas();
  ```

  Dann ganz normal zeichnen, und mit

  ```
  c->Print( "Dateiname" );
  ```

  in Ausgabedatei speichern. Der Dateityp wird durch die Endung (.pdf, .png, …) festgelegt.

ROOT 7

- https://root.cern/root-7
- expected ~2018 (during LS2)
- „For the first time since 20 year (i.e. ever), the ROOT team plans to **break backward compatibility** for crucial interfaces – once.“
- „This new major version of ROOT will make ROOT much simpler and safer to use: we want to increase clarity and usability. If you are a physicist, please read on - this is about your ROOT.“
- „We will use standard C++ types, standard interface behavior, good documentation and tests: we are trying to be nice!“

Thank you!

Komplexe Daten in Trees

- Trees können (fast) beliebige C++-Klassen speichern.
- Spezielle Syntax nötig.
  - relativ komplex
  - ⇨ s. Dokumentation zur jeweiligen ROOT-Version
- Funktionen der Klassen können aufgerufen werden, z.B. als Spaltenname in einem Tree.
  - ⇨ aufwändige Operationen in compiliertem (=performanterem) Code.