technische universität dortmund

e5 experimentelle physik 5
teilchenphysik & medizinphysik

Programmierkurs 2020 | ROOT

Alexander Ratke

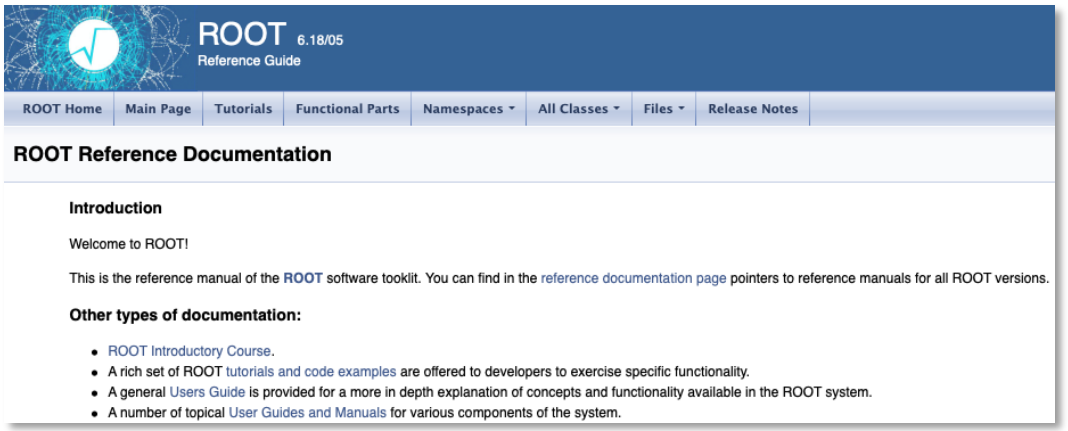
alexander.ratke@tu-dortmund.de

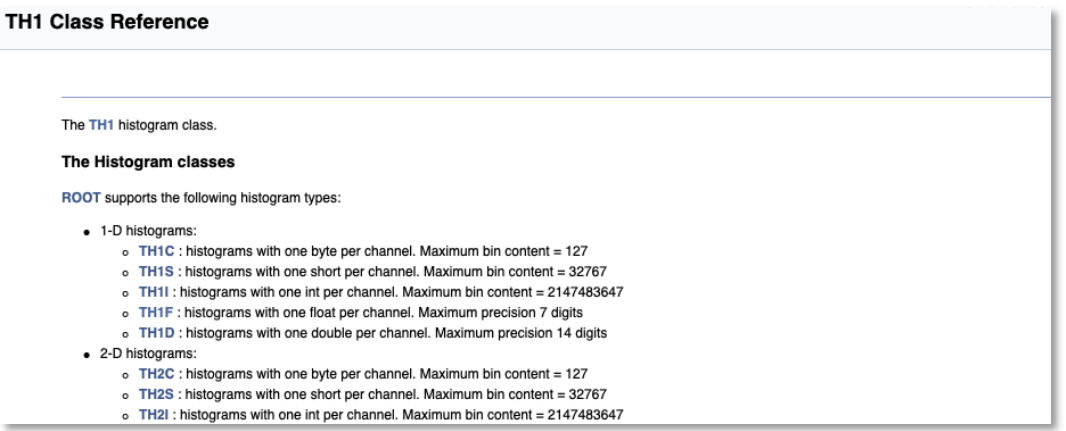
Was ist ROOT?

- Am CERN entwickeltes objektorientiertes Softwarepaket
- Basierend auf C++ Klassen, jedoch keine detaillierten Kenntnisse von C++ notwendig
- Verwendung:
  - Datenverarbeitung
  - Visualisierung von Daten
  - Vielzahl von Fitfunktionen mit der Erweiterung RooFit
  - Statistikfunktionen

Wo bekomme ich Hilfe?

- Website: https://root.cern.ch/
  - User‘s Guide
  - Tutorials
  - ROOT-Forum
- Google:
  - Bspw. ROOT TH1 oder TTree





tu e5

Wie benutze ich ROOT?

- Möglichkeiten, um ROOT zu benutzen:
  - Interaktive Shell (CINT)
  - Schreiben von Makros, die von ROOT ausgeführt werden können
  - Einbindung von ROOT-Klassen in C++ Code
  - ROOT-Modul für Python (PyROOT)

- Importieren von ROOT in Python:
  - `import ROOT as R`

Überblick

- ROOT bietet verschiedene Datenstrukturen, in denen Daten gespeichert werden können oder mit denen auf Daten zugegriffen werden kann:
  - Histogramme (TH Klassen in 1D, 2D oder 3D)
  - ROOT-Dateien (TFile, TTree)
  - TBrowser (schnelles Betrachten von ROOT-Dateien)
  - Grafische Darstellung (TGraph, TGraphErrors)
  - und vieles mehr

Histogramme

1. Histogramm-Objekt muss angelegt werden:
   - `myHist = R.TH1D("myHist", "myHist", 10, 0.0, 1.0)`
     Objektname | Name | Titel | Anzahl der Bins, x-Achsen-Intervall
2. Werte zum Histogramm hinzufügen:
   - `myHist.Fill(x)`
3. Histogramm zeichnen:
   - `myHist.Draw()`

The Histogram classes

ROOT supports the following histogram types:

- 1-D histograms:
  - TH1C : histograms with one byte per channel. Maximum bin content = 127
  - TH1S : histograms with one short per channel. Maximum bin content = 32767
  - TH1I : histograms with one int per channel. Maximum bin content = 2147483647
  - TH1F : histograms with one float per channel. Maximum precision 7 digits
  - TH1D : histograms with one double per channel. Maximum precision 14 digits
- 2-D histograms:
  - TH2C : histograms with one byte per channel. Maximum bin content = 127
  - TH2S : histograms with one short per channel. Maximum bin content = 32767
  - TH2I : histograms with one int per channel. Maximum bin content = 2147483647
  - TH2F : histograms with one float per channel. Maximum precision 7 digits
  - TH2D : histograms with one double per channel. Maximum precision 14 digits
- 3-D histograms:
  - TH3C : histograms with one byte per channel. Maximum bin content = 127
  - TH3S : histograms with one short per channel. Maximum bin content = 32767
  - TH3I : histograms with one int per channel. Maximum bin content = 2147483647
  - TH3F : histograms with one float per channel. Maximum precision 7 digits
  - TH3D : histograms with one double per channel. Maximum precision 14 digits

Aufgabe

a) Erzeugt 1.000.000 zufällige Werte aus einer Standardnormalverteilung und stellt diese in einem ROOT-Histogramm dar. Tipp: `R.gRandom.Gaus()`

b) Versucht mithilfe der ROOT-Dokumentation herauszufinden, wie das Histogramm so bearbeitet werden kann, dass

   I. die x-Achse den Titel „x“ hat.

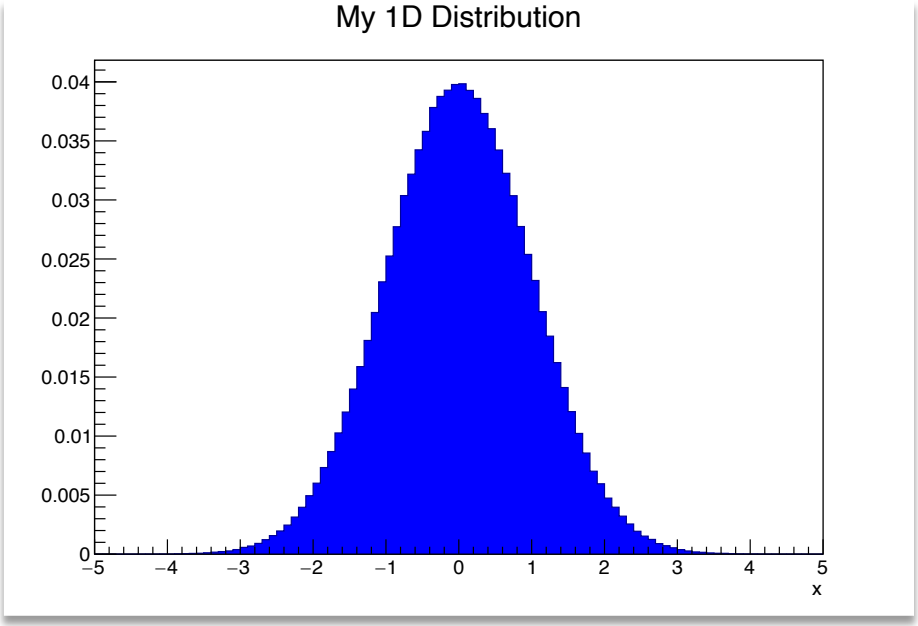
   II. die Fläche unterhalb des Histogramms blau ist.

   III. es normiert ist.

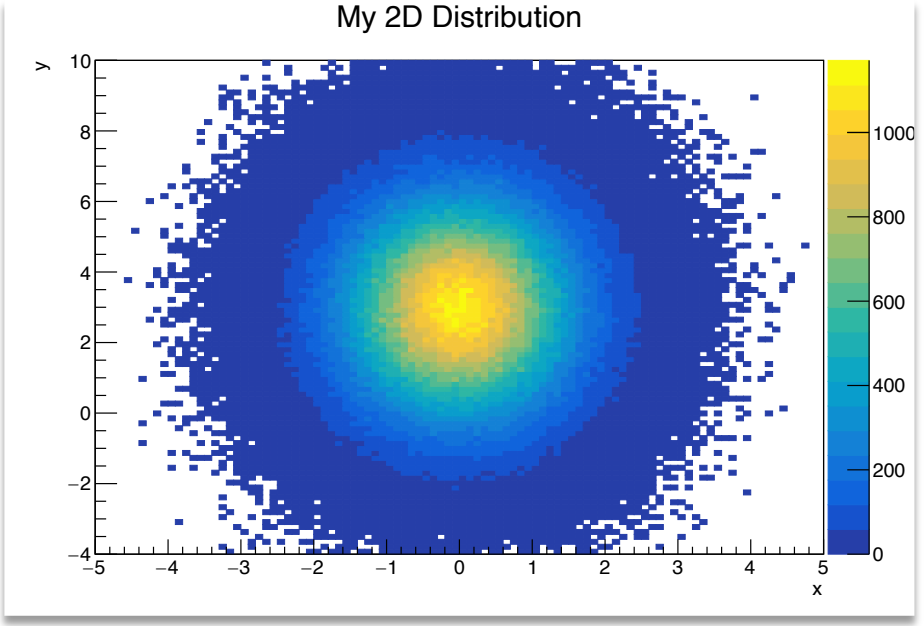
c) Erzeugt weitere 1.000.000 Werte aus einer Normalverteilung mit $\mu = 3$ und $\sigma = 2$ und stellt diese mit den vorhandenen Werten aus Teil a) in einem 2D Histogramm dar.

Resultate

Aufgabenteil a) und b)



Aufgabenteil c)



ROOT-Dateien

- Datensätze in einer ROOT-Datei anlegen mithilfe von TTree:
  - Tree besteht aus Branches, die Variablen mit unterschiedlichen Datentypen (bool, int, double) darstellen können
  - Speicherung im `.root`-Datenformat möglich

1. Öffnen einer ROOT-Datei zum Speichern des Trees:
   - `myFile = R.TFile("myFile.root", "RECREATE")`
2. Tree-Objekt muss angelegt werden:
   - `myTree = R.TTree("myTree", "myTree")`

ROOT-Dateien

3. Variablen in einen Tree als Branches hinzufügen:
   - `value = np.zeros(1, dtype=np.float64)`
   - `myTree.Branch("myBranch", value, "myBranch/D")`

     Double
4. Werte von `value` in den Tree schreiben:
   - `myTree.Fill()`
5. Tree muss in die Datei geschrieben werden:
   - `myTree.Write("", R.TObject.kOverwrite)`
6. Datei muss am Ende geschlossen werden:
   - `myFile.Close()`

Aufgabe

a) Erzeugt 1.000.000 zufällige Werte aus einer Standardnormalverteilung, füllt diese in einen Tree und speichert den Tree in einer ROOT-Datei ab.

b) Erzeugt weitere 1.000.000 Werte aus einer Normalverteilung mit $\mu = 3$ und $\sigma = 2$ und fügt diese dem Tree als weitere Variable hinzu.

ROOT-Dateien

- Einlesen von Daten aus einer ROOT-Datei mit dem Modus `READ`
- Zuweisung des Trees:
  - `myTree = myFile.Get("myTree")`
- Anzahl aller Einträge im Tree:
  - `myTree.GetEntries()`
- Auf die Werte einer Variablen zugreifen:
  - `myTree.GetEntry(i)`

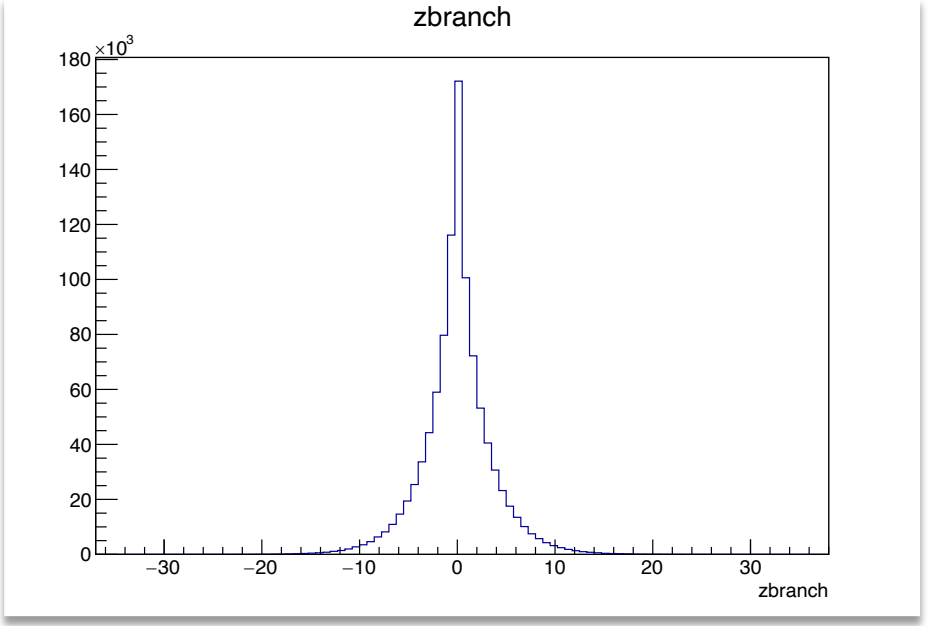
Aufgabe

a) Lest den in Aufgabe 2 erzeugten Datensatz ein und erzeugt mit den Daten eine neue ROOT-Datei, in der zusätzlich noch das Produkt beider Zufallsvariablen als dritte Variable enthalten ist.

b) Zeichnet die neue Variable als Histogramm.

Resultate

Aufgabenteil b)

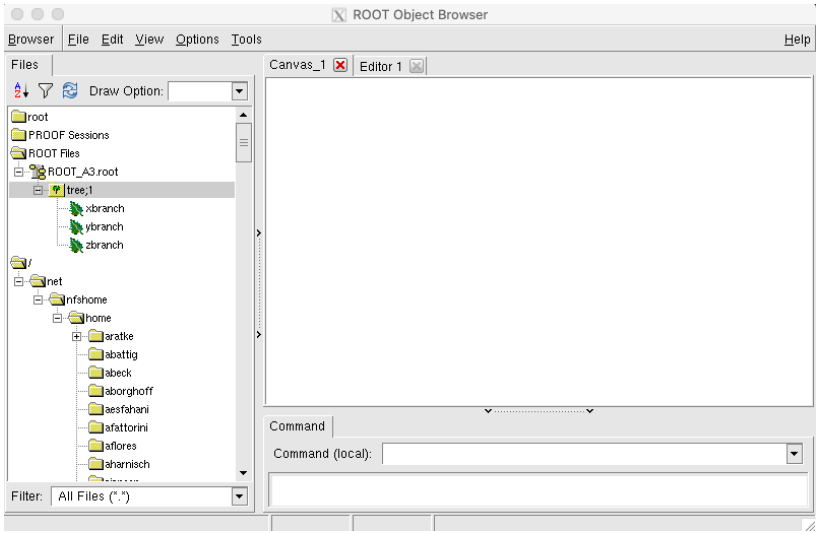


TBrowser

- Einfache Visualisierung von Variablen in einer ROOT-Datei

1. Terminal öffnen und Eingabe von
   - `root path/to/file.root`
2. Interaktive ROOT Shell wird gestartet
   - Ähnlich wie ipython, aber nur in C++
3. Browser starten mit
   - `new TBrowser()`

- Jetzt folgt eine kleine Vorführung



TBrowser

Hilfreiche Befehle anhand der ROOT-Datei aus vorheriger Aufgabe:

- Variable aus einem Tree zeichnen
  - `tree->Draw("zbranch", "")`
- Bestimmten Bereich einer Variable anschauen
  - `tree->Draw("zbranch>>h(100,-10,10)", "")`
- Visualisierung einer Variablen mit Selektion
  - `tree->Draw("zbranch", "xbranch>-1&&xbranch<1")`
- TBrowser kann auch 2D plotten
  - `tree->Draw("ybranch:xbranch", "")`

Komplexere Aufgabe

- Wir betrachten einen Datensatz von simulierten $B^+ \rightarrow K^+ J/\psi\,(\rightarrow e^+e^-)$-Zerfällen

a) Öffnet die Datei A4_file.root mit dem TBrowser und verschafft euch einen Überblick über die enthaltenen Variablen und deren Bedeutung.

b) Berechnet die invariante Masse des J/ψ-Mesons (relativistisch) und fügt diese als neue Variable dem Tree hinzu.

c) Schaut euch die berechnete Massenverteilung des J/ψ-Mesons an.

d) Berechnet die Effizienz der Selektion 2400 MeV/c^2 < m(J/ψ) < 3300 MeV/c^2 und berechnet den Fehler als Binomialfehler.

Was kommt als Nächstes?

- Anpassung von Verteilungen mit RooFit