



Programmierkurs 2022 | ROOT

Fabian Koch aufbauend auf der Arbeit von Alexander Ratke

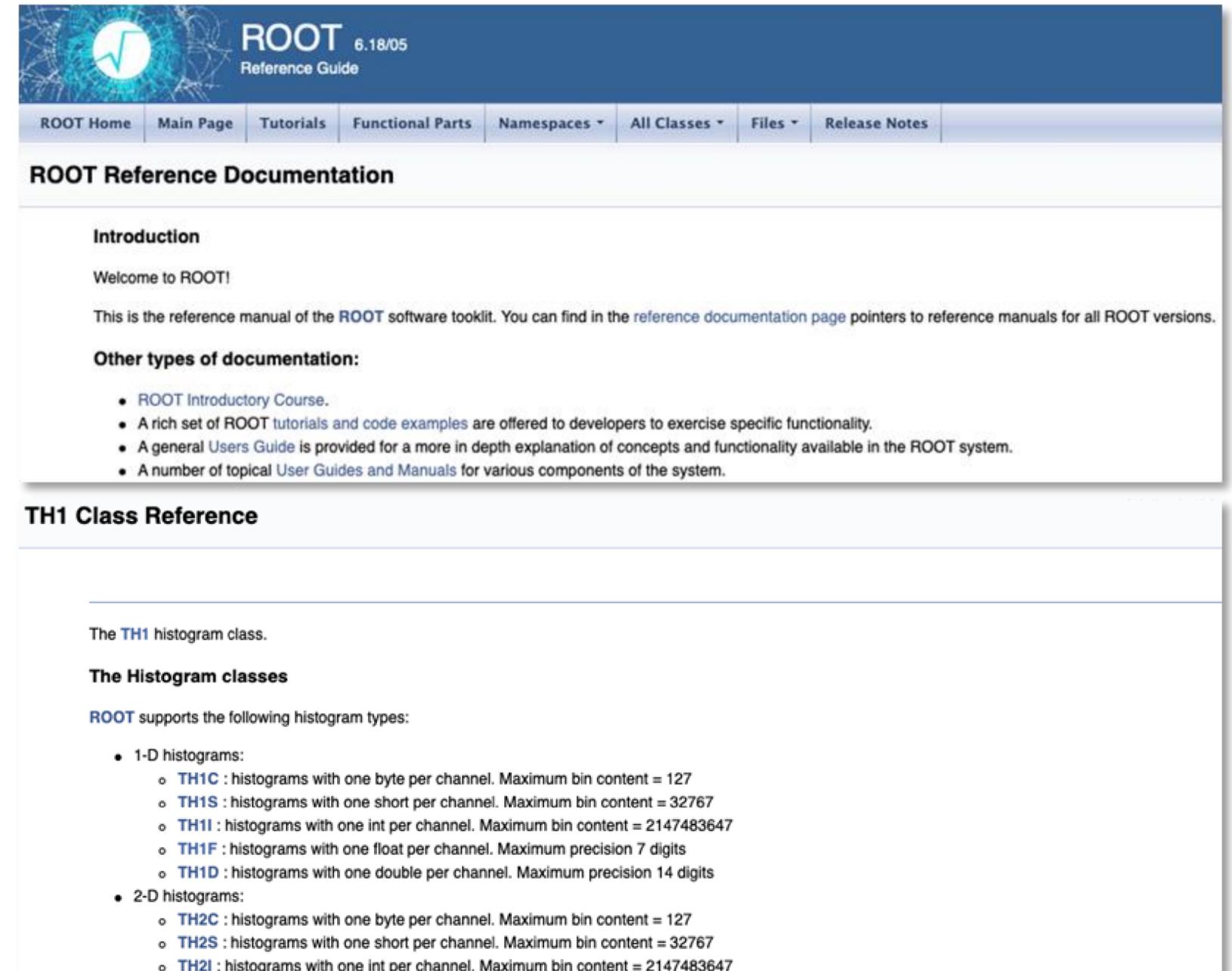
fabian3.koch@tu-dortmund.de

Was ist ROOT?

- ▶ Am Cern entwickeltes objektorientiertes Softwarepaket
- ▶ Basierend auf C++ Klassen, jedoch keine detaillierte 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
- ▶ Kommilitonen fragen
- ▶ E5 Question and answer website <https://qa.e5.physik.tu-dortmund.de/>
- ▶ Eure Betreuer und andere am Lehrstuhl



The screenshot shows the ROOT Reference Guide website. The header includes the ROOT logo and version 6.18/05. The navigation menu contains links for ROOT Home, Main Page, Tutorials, Functional Parts, Namespaces, All Classes, Files, and Release Notes. The main content area is titled "ROOT Reference Documentation" and includes an "Introduction" section with a welcome message and a link to the reference documentation page. Below this is a section for "Other types of documentation" with a bulleted list of resources. The second section is titled "TH1 Class Reference" and describes the TH1 histogram class, listing supported histogram types and their bin content limits.

ROOT 6.18/05
Reference Guide

ROOT Home Main Page Tutorials Functional Parts Namespaces All Classes Files Release Notes

ROOT Reference Documentation

Introduction

Welcome to ROOT!

This is the reference manual of the ROOT software toolkit. You can find in the [reference documentation page](#) pointers to reference manuals for all ROOT versions.

Other types of documentation:

- [ROOT Introductory Course](#).
- A rich set of [ROOT tutorials and code examples](#) are offered to developers to exercise specific functionality.
- A general [Users Guide](#) is provided for a more in depth explanation of concepts and functionality available in the ROOT system.
- A number of topical [User Guides and Manuals](#) for various components of the system.

TH1 Class Reference

The [TH1](#) histogram class.

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

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`

- ▶ 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

1. Histogramm-Objekt muss angelegt werden:

- `myHist = R.TH1D("myHist", "myHist", 10, 0.0, 1.0)`

Objektname Name Titel Anzahl der Bins, x-Achsen-Intervalle

2. Werte zum Histogramm hinzufügen:

- `myHist.Fill(x)`

3. Histogramm zeichnen:

- `myHist.Draw()`

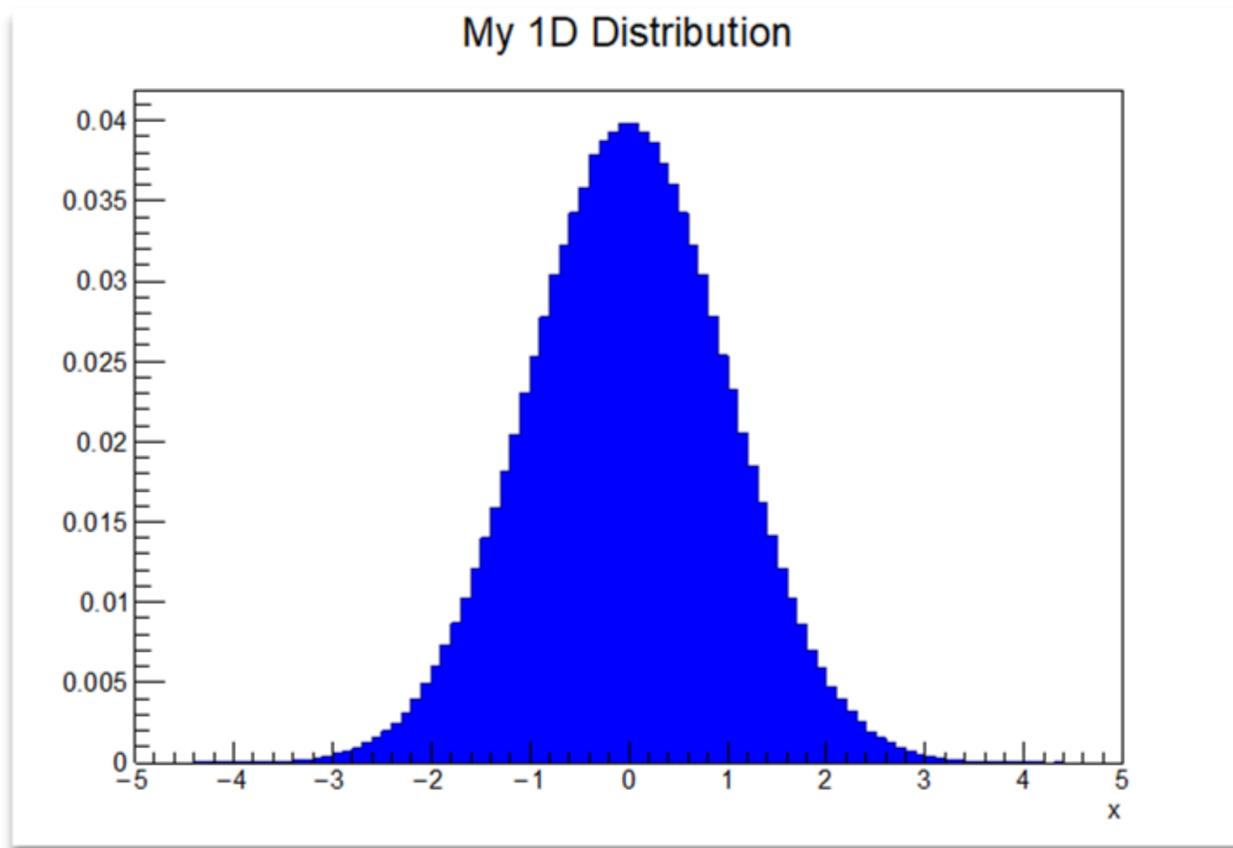
The Histogram classes

ROOT supports the following histogram types:

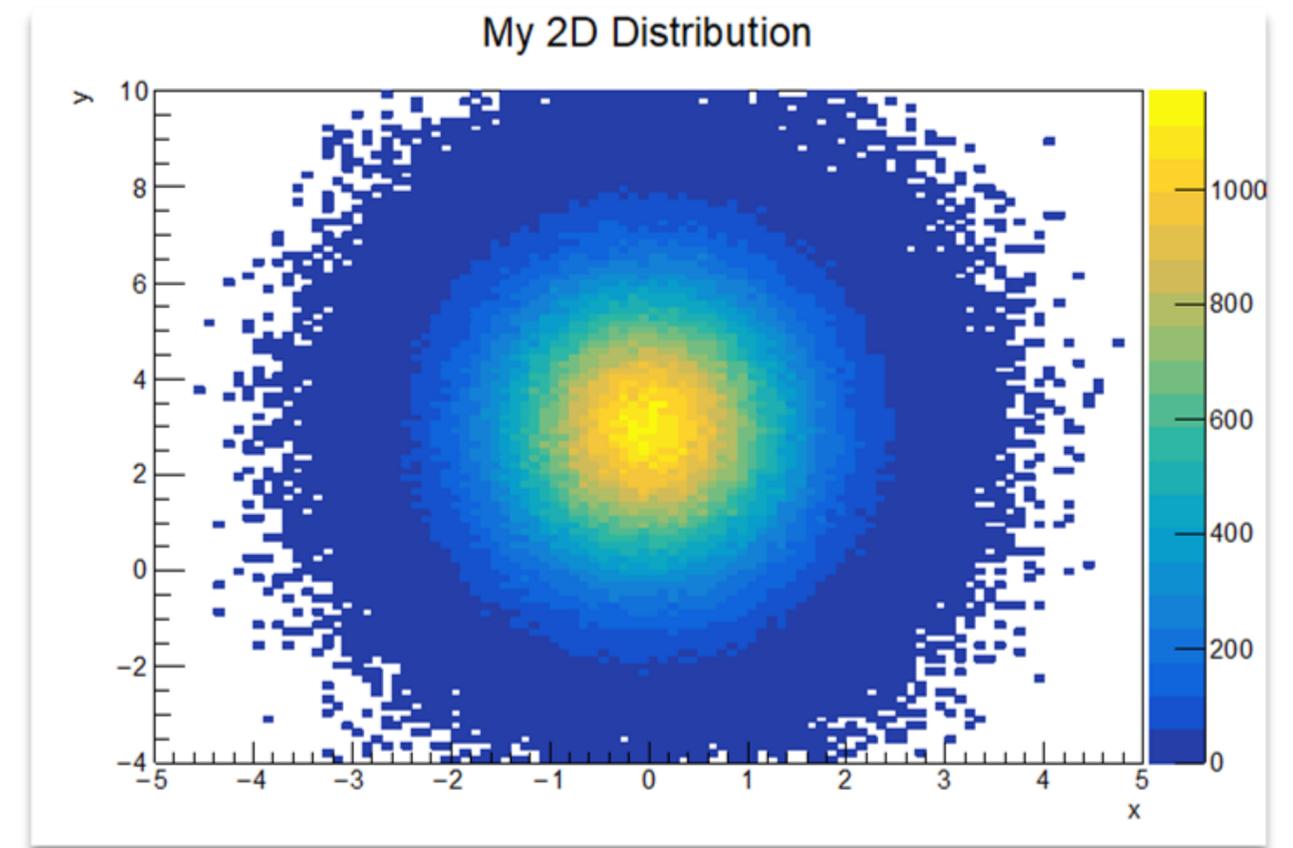
- 1-D histograms:
 - **TH1C** : histograms with one byte per channel. Maximum bin content = 127
 - **TH1S** : histograms with one short per channel. Maximum bin content = 32767
 - **TH1I** : histograms with one int per channel. Maximum bin content = 2147483647
 - **TH1F** : histograms with one float 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

- 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

Aufgabenteil a) und b)



Aufgabenteil c)



- ▶ 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-Dateien zum Speichern des Trees:
 - `myFile = R.TFile("myFiel.root", "RECREATE")`
- 2. Tree-Objekt muss angelegt werden:
 - `myTree = R.TTree("myTree", "myTree")`

Vorsicht bei RECREATE: das überschreibt die Datei, falls sie vorher existiert.
Es gibt auch NEW, UPDATE, READ

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

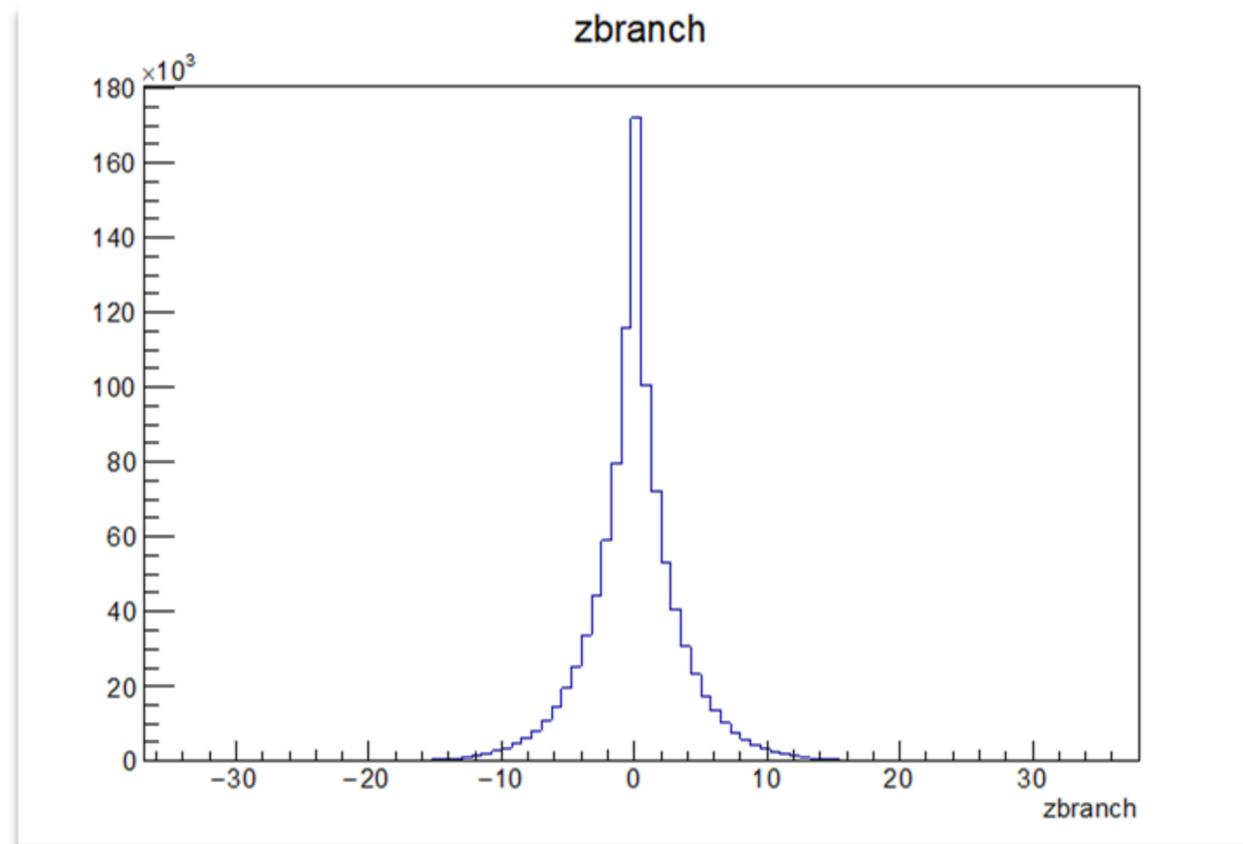
- a) Erzeugt 1.000.000 zufällige Werte aus einer Standardnormalverteilung, füllt diese in einen Tree und speichert den Tree als ROOT-Datei ab.
- b) Erzeugt weitere 1.000.000 Werte aus einer Normalverteilung mit $\mu = 3$ und $\sigma = 2$ und fügt diese dem Tree als weitere Variable hinzu.

- ▶ 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)`
- ▶ Anschauen aller Branche und deren Werte:
 - `myTree.Show(i)` oder `myTree.Print()`

- 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)



► Einfache Visualisierung von Variablen in einer ROOT-Datei

1. Terminal öffnen und Eingabe von

- rootls zum Anzeigen der Inputfiles und root path/to/file.root zum Öffnen

2. Interaktive ROOT Shell wird gestartet

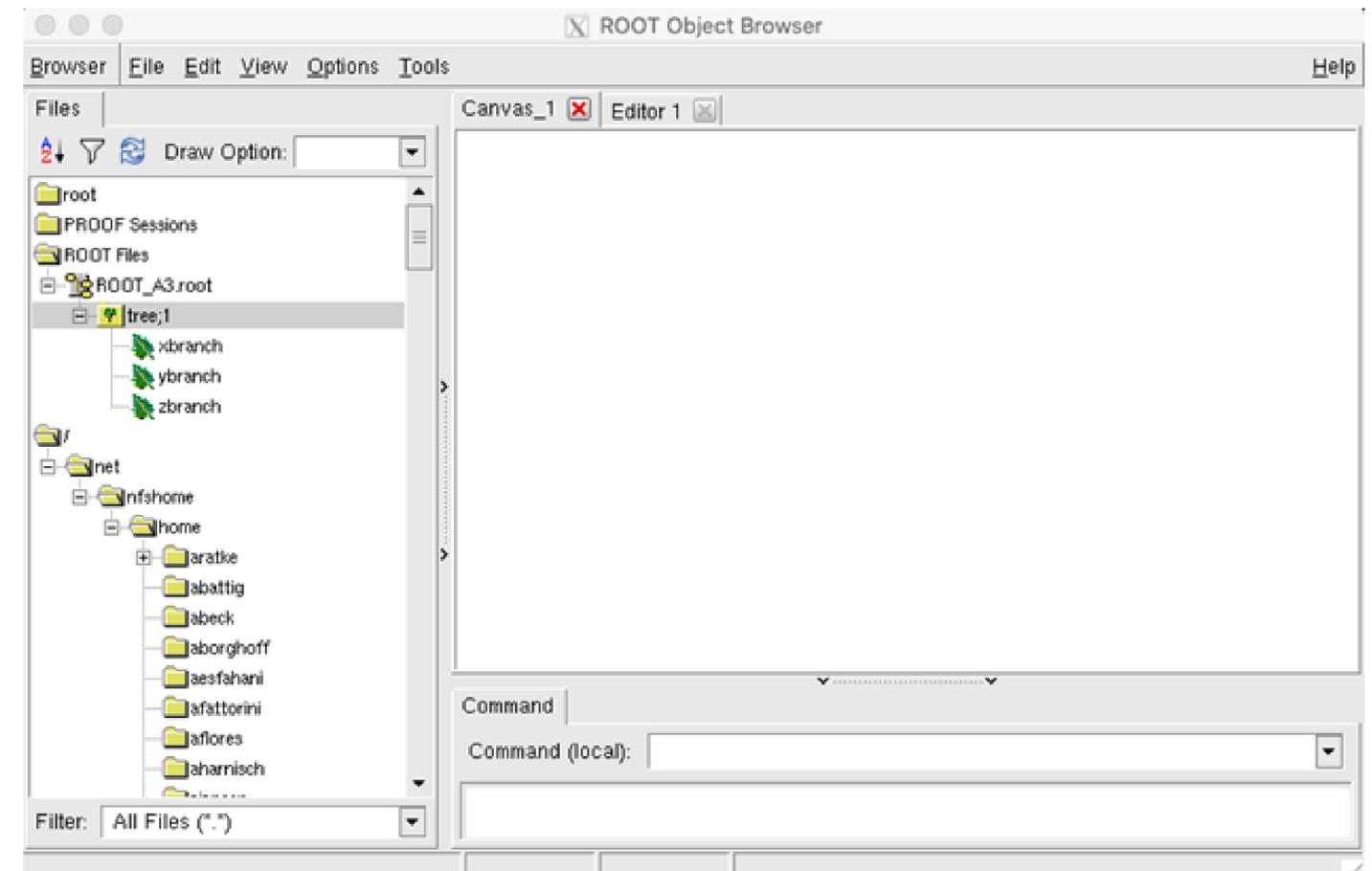
- Ähnlich wie Ipython, aber nur in C++
- Selbe Befehle wie im Skript möglich

3. Browser starten mit

- new TBrowser()

Bei nicht-lokalen
Dateien längere
Ladezeiten

► Jetzt folgt eine kleine Vorführung



- ▶ Hilfreiche Befehle anhand der ROOT-Datei aus vorheriger Aufgabe:
- ▶ Variablen aus einem Tree zeichnen:
 - `tree->Draw("zBranch", "", "")`
- ▶ Bestimmten Bereich einer Variablen 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 $B^+ \rightarrow K^+ J/\psi(\rightarrow e^+ e^-)$
- 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 $2400 \text{ MeV}/c^2 < m(J/\psi) < 3300 \text{ MeV}/c^2$ und berechnet die Unsicherheit als Binomialfehler.

Was kommt als Nächstes?

- Anpassen von Verteilung mit RooFit

