Rechnernutzung in der Physik

Institut für Experimentelle Teilchenphysik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. M. Giffels, Dr. R. Wolf
Dr. A. Mildenberger

WS2015/16 – Blatt 09
`http://comp.physik.kit.edu` Prog.: Di., 22.12.2015 / Ausarb.: Di., 12.01.2016

Einführung in die Anwendung von ROOT

ROOT (`http://root.cern.ch`) ist ein Programmpaket zur grafischen Auswertung insbesondere großer Datenmengen. Die von ROOT zur Verfügung gestellten Klassen können entweder in eigene Programme eingebunden oder interaktiv in der „ROOT shell“ aufgerufen werden. Zusätzlich gibt es die `Python`-Anbindung `pyroot`, die die Nutzung einzelner ROOT-Klassen aus `Python` erlaubt.

ROOT-Tutorial — für Bachelorstudiengang

Um sich mit ROOT vertraut zu machen, laden Sie die Datei `divingROOT.zip` in Ihr Arbeitsverzeichnis und entpacken Sie sie. Arbeiten Sie Kapitel 1 – 3 und 8 des Tutorials `diving_into_ROOT.pdf` durch. Alle Beispiele finden Sie nach Entpacken der `zip`-Datei im Unterverzeichnis `macros`. Bitte machen Sie sich auch mit den im Tutorial angegebenen Informationsquellen vertraut, insbesondere mit den Klassendefinitionen im `Reference-Guide` und dem `ROOT User's Guide`.

Aufgabe 16: Gauß-Funktion in ROOT — freiwillig

Nachdem Sie das Tutorial durchgearbeitet haben, sollten Sie nun in der Lage sein, eine normierte Gauß-Funktion mit vorgegebenem Mittelwert von $\mu = 5.0$ und Standardabweichung $\sigma = 1.5$ in ROOT darzustellen. Als Vorlage zu dieser Aufgabe dienen die Dateien `PlotGauss.C` bzw. `PlotGauss-pyroot.py`, die Sie erweitern können.
Als Beispiel sollten Sie auch die Dateien `PlotFermi.C` bzw. `PlotFermi-pyroot.py` anschauen, die eine Lösung von Aufg. 5, Blatt 2 als ROOT C++-Makro bzw. als `pyroot`-Skript zeigen (s. auch Anhang).
Ob Sie Code in `C++/C` oder `Python` verwenden wollen, bleibt Ihnen überlassen. Wie schon in der Vorlesung angemerkt, ist für große, zeitkritische Projekte die Erstellung von C-Code unbedingt vorzuziehen, für die Übungsaufgaben dieses Kurses ist aber `Python` ausreichend.
Zeichnen Sie auch die Ableitung und das Integral der von Ihnen implementierten Funktion im vorgegebenen Intervall. Sehen Sie dazu unter `http://root.cern.ch/root/html/TF1.html` nach, welche `Draw()`-Funktionen es zur Klasse `TF1` gibt.

Aufgabe 17: Zentraler Grenzwertsatz — Ausarbeitung[1]

In dieser Aufgabe soll der wichtige zentrale Grenzwertsatz illustriert werden. Sie können diese Aufgabe wahlweise mit `ROOT` oder `Python` bearbeiten.
`ROOT`: Machen Sie sich bitte mit der Klasse `TH1` vertraut, und lesen Sie das entsprechende Kapitel im Tutorial „Diving into Root“. Die Beispieldateien `PlotUniform.C` (siehe dazu auch im Anhang dieses Übungsblatts) und `PlotUniform-pyroot.py` zeigen, wie man gleichverteilte Zufallszahlen in `ROOT` in ein Histogramm einträgt und es darstellt. Eine Vorlage zur Bearbeitung

[1] Zum Teil „Statistik und Datenanalyse“ der Computerübung werden mindestens vier Ausarbeitungen angeboten, von denen drei verpflichtend sind.

dieser Aufgabe enthalten die Dateien `CentralLimit.C` bzw. `CentralLimit-pyroot.py`.
`Python`: In Aufg. 6, Blatt 3 hatten Sie ein Beispiel für die Darstellung eines Histogramms und einer Gauß-Verteilung kennen gelernt; das Beispielskript `PlotUniform.py` demonstriert, wie ganz analog gleichverteilte Zufallszahlen erzeugt und histogrammiert werden können. Als Vorlage für diese Aufgabe können Sie das Skript `CentralLimit.py` verwenden.

Betrachten Sie die Summe y von n gleichverteilten Zufallszahlen x_i im Intervall [0,1], $y = \sum_{i=1}^{n} x_i$.
a) Zeigen Sie analytisch, dass eine Gleichverteilung im Intervall [0,1] eine Standardabweichung von $\sigma = \sqrt{1/12}$ hat. Zeigen Sie weiter, dass y einen Erwartungswert von $n/2$ und eine Varianz von $n/12$ hat.
b) Nach dem zentralen Grenzwertsatz nähert sich die Verteilung von y für große n einer Gauß-Verteilung an. Dies soll nun überprüft werden. Wir betrachten dazu die Variable

$$z = \frac{(\sum_{i=1}^{n} x_i) - \frac{n}{2}}{\sqrt{n/12}}$$

mit Mittelwert 0 und Standardabweichung 1.
Schreiben Sie ein Programm, um jeweils 100 000 Werte von z für $n = 2$, 3 und 20 zu erzeugen und füllen Sie diese in Histogramme. Zeichnen Sie jeweils die passend normierte Standard-Gauß-Kurve (d.h $\mu = 0$, $\sigma = 1$) in Ihre Histogramme ein und vergleichen Sie. Achten Sie auf eine ausreichende Anzahl an Bins, damit der Vergleich aussagekräftig ist.
Zu dieser Aufgabe wird eine **schriftliche Ausarbeitung** verlangt. Neben den analytischen Berechnungen von oben fügen Sie bitte Ausdrucke der Histogramme mit überlagerten Gauß-Kurven für $n = 2$, 3 und 20 bei.

Aufgabe 18: Binomialverteilung und Nachweiseffizienz — Programmtestat

Oft treten zufällige Ereignisse mit fest vorgegebener Wahrscheinlichkeit auf - z. B. „Kopf“ oder „Zahl“ beim Münzwurf oder das Ansprechen eines Detektors. Solch einen Prozess wollen wir in dieser Aufgabe in einer einfachen Simulation untersuchen. Sie können die Aufgabe in `Python` oder `ROOT` lösen; hilfreich als Vorlagen sind die in den Vorlesungen 8 und 9 gezeigten Beispiele zu Histogrammen und Verteilungen.
Schreiben Sie eine Funktion, die eine Folge von n Zahlen b_i erzeugt, die mit vorgegebener Wahrscheinlichkeit p den Wert 1 und sonst 0 annehmen[2]. Die Funktion soll als Rückgabewert die Zahl der Einsen angeben.
Rufen Sie diese Funktion mit $n = 100$ und $p = 0.75$ 1000 mal auf und histogrammieren Sie die jeweilige Häufigkeit k_i der Zahl 1.[3] Wie Sie aus der Vorlesung wissen, sollten die k_i einer Binomialverteilung folgen. Berechnen Sie Mittelwert und Standardabweichung der k_i und vergleichen Sie mit den Eigenschaften der Binomialverteilung. Überlagern Sie die Binomialverteilung und beobachten Sie die Übereinstimmung mit dem Histogramm, wenn Sie die Zahl der „Pseudo-Experimente“ von 1000 auf 10000 und evtl. 100’000 erhöhen. Studieren Sie die Fälle $p = 0.75$ und $p = 0.98$.
Anmerkung zur Nachweiseffizienz: In einem Experiment wird die Zahl k der günstigen Ausgänge eines n-mal durchgeführten Experiments (z. B. der Teilchennachweis mit einem Geiger-Müller-Zählrohr) benuzt, um einen „Schätzwert“ für die Nachweiseffizienz, $\hat{\epsilon} = \frac{k}{n}$, und ihre Unsicherheit, $\Delta\hat{\epsilon} = \sqrt{\frac{\hat{\epsilon}\cdot(1-\hat{\epsilon})}{n}}$ zu bestimmen. Für sehr kleine oder sehr große Effizienzen führt dies bei kleiner Ereignisstatistik zu vielen Experimenten, in denen die Unsicherheit fälschlicherweise zu null bzw. zu klein abgeschätzt wird. Dann sind aufwändigere statistische Methoden notwendig, um eine Untergrenze für die wahre Nachweiseffizienz als sogenanntes „Konfidenzintervall“ festzulegen.

[2] *Tipp:* man verwendet dazu gleichverteilte Zufallszahlen $r_i \in [0, 1[$; falls $r_i < p$ wird $b_i = 1$, sonst $b_i = 0$.
[3] Da nur ganze Zahlen auftreten, sollten Sie auf die Bingrenzen achten!

ANHANG: Makros und ROOT-Dateien

Als Beispiel zum Umgang mir ROOT betrachten wir das einfache Programm `PlotFermi.C` bzw. `PlotFermi-pyroot.py`. `C++/C`-Code kann sowohl als ROOT-Makro innerhalb der ROOT-Shell ausgeführt werden (mittels `.x PlotFermi.C`), als auch mit Hilfe der Datei `Makefile` für sich alleinstehend kompiliert, gelinkt (`make SOURCE=PlotFermi`) und dann ausgeführt werden. `Python`-Code wird direkt von der Linux-Kommandozeile ausgeführt (`python PlotFermi-pyroot.py`) Die erzeugte `.root`-Datei können Sie in ROOT öffnen und mit dem ROOT-Browser ansehen (s. dazu weiter unten).
Source-Code in `C` kann sowohl als ROOT-Makro ausgeführt als auch kompiliert und dann ausgeführt werden. Die notwendigen Schritte zum Kompilieren und Linken sind in der Datei `Makefile` definiert, die vom Befehl `make` verwendet wird.
Ein weiteres Beispiel zur Erzeugung und Histrogrammierung von 10000 zwischen 0 und 1 gleichverteilten Zufallszahlen ist `PlotUniform.C` bzw. `PlotUniform-pyroot.py`.
Zur Funktion des einfachen Beispiels:
Mit der Methode `gRandom→Rndm()` aus dem ROOT-Paket, d.h. `x=gRandom→Rndm()` erzeugt man eine Zufallszahl x. Mit der `Fill`-Methode der Histogramm-Klasse `TH1F` werden die Ergebnisse in ein Histogramm mit 100 Bins gefüllt. Das Buchen des eindimensionalen Histogramms geht mit dem Befehl

```
TH1F *h100 = new TH1F("h100","some title",NumberBinsX,XMin,XMax);
```

und das Füllen mit `h100->Fill(x);`.
Das Programm kann entweder mit Hilfe von `make` und dem vorbereiteten `Makefile` übersetzt (`make SOURCE=PlotUniform`) und ausgeführt (`./PlotUniform`) oder direkt als Makro aus der ROOT-Shell gestartet werden `.x PlotUniform.C`.
Das Ergebnishistogramm `h100` wird in der Datei `Uniform.root` gespeichert, die Sie in einer interaktiven ROOT-Sitzung durch Eingabe von `> root Uniform.root` in einem Konsolenfenster anzeigen können. Dies ist äquivalent zu folgender Vorgehensweise:

1. ROOT starten, indem Sie `root` in einem Konsolenfenster eingeben.
2. Datei (Uniform.root) einlesen, d.h. mittels `TFile file("Uniform.root")` öffnen.

Jetzt erzeugen Sie noch eine Instanz der Klasse `TBrowser` mittels `TBrowser b;`. Durch Klicken auf `ROOT Files` und `Uniform.root` sowie einen Doppel-Klick auf das Histogramm können Sie dieses anzeigen.
Alternativ können Sie in der Funktion `PlotUniform` auch die Methode `Draw()` der Histogramm-Klasse benutzen, um Ihr Histogramm direkt am Ende des Programms ohne den interaktiven Schritt anzeigen zu lassen (geht nur bei Ausführung als ROOT-Makro oder in pyroot).

Hinweis: Mit dem Rechnernamen `fphctssh.physik.uni-karlsruhe.de` können Sie von überall aus mittels ssh/scp Programm per Netzwerk auf einen Poolrechner zugreifen.
