Statistische Methoden der Datenanalyse WS 2017/18

Prof. Dr. Ulrich Landgraf

Aufgabenblatt 2 vom 07.11.2017

Aufgabe 1 (2 Punkte)

In der Statistik verwendet man zur Darstellung von Ergebnissen und zur weiteren Verarbeitung oft Histogramme. Ein Histogramm ist ein Balkendiagramm, bei dem die horizontale Achse in endlich viele Abschnitte eingeteilt ist, die dem Wert einer Zufallsvariablen entsprechen. Auf der vertikalen Achse wird angegeben, wie oft der betreffende Wert aufgetreten ist.

Um die Verwendung von Histogrammen in ROOT zu erlernen, gehen Sie bitte von dem kleinen C++ Programm `MyHisto.C` aus, das Sie unter
`http://hep.uni-freiburg.de/tl_files/home/wwwherten/statistik/MyHisto.C`
finden. Sie sehen darin eine Konstruktion, die als Datenfeld oder Array bezeichnet wird und die es uns erlaubt, mehrere Zahlen unter einem gemeinsamen Namen (hier: `data`) zusammenzufassen. Um Ihnen zu zeigen, wie man in einer Schleife auf einzelne Elemente des Datenfeldes zugreifen kann, werden danach die Elemente auf dem Bildschirm ausgegeben.

Vervollständigen Sie bitte jetzt dieses ROOT-Programm, indem Sie mit den Befehlen der Klasse `TH1` ein eindimensionales Histogramm erzeugen, in einer Schleife die Elemente des Arrays `data` einfüllen und schließlich das Ergebnis auf dem Bildschirm anzeigen lassen. Auf der Seite `https://root.cern.ch/doc/v610/classTH1.html` finden Sie die Histogrammklasse `TH1` mit Ihren Methoden erklärt. Erzeugen Sie zunächst ein Histogramm der Unterklasse `TH1F` mit 10 Kanälen von 0.5 bis 10.5. Füllen Sie danach in einer Schleife die Daten aus dem Array `data` in das Histogramm (wie es unter der Überschrift "Filling histograms" beschrieben wird). Verwenden Sie schließlich die Methode `Draw()` der Klasse `TH1` um das Histogramm anzuzeigen.

Aufgabe 2 (2 Punkte)

Als nächstes wollen wir ROOT benutzen, um uns Zufallszahlen erzeugen zu lassen. Dafür stehen verschiedene Methoden zur Verfügung, die als Klassen `TRandom, TRandom1, TRandom2` und `TRandom3` implementiert sind. Wie alle anderen ROOT-Klassen, finden Sie die Dokumentation unter `https://root.cern.ch/doc/v610/annotated.html`. `TRandom3` ist der beste der Generatoren; die Klasse `TRandom3` ist aber von `TRandom` abgeleitet (das nennt man bei Klassen Vererbung/Inheritance); daher sind in der Dokumentation unter `TRandom3` nur die zusätzlichen Methoden aufgeführt und man findet fast alle Methoden bereits in `TRandom`. Es gibt aber bei der Dokumentation unter `TRandom3` auch einen Verweis unter „Additional Inherited Members“.

Erstellen Sie ein Objekt der Klasse `TRandom3` und lassen Sie sich mit der Methode `Binomial` 20 Zufallszahlen erzeugen, die gemäß einer Binomialverteilung von $ntot = 20$ und einer Wahrscheinlichkeit $p = 0.2$ verteilt sind. Histogrammieren Sie diese 20 Zahlen und vergleichen Sie sie mit der in der Vorlesung gezeigten Verteilung $B(k; 20, 0.2)$. Vergleichen Sie auch Mittelwert und Standardabweichung (welche im Histogramm als RMS angezeigt wird).

Bitte wenden!

Wenn Sie dieses Programm mehrere Male laufen lassen, bekommen Sie immer dasselbe Histogramm. `Random3` erzeugt nämlich (wie jeder Zufallsgenerator auf dem Rechner) nur Pseudozufallszahlen. Wenn man eine andere Zufallssequenz erhalten will, muss man mit der Methode `SetSeed(i)` die sogenannte Seed ändern, wobei `i` eine Integer Zahl ist. `SetSeed(0)` leitet die gewählte Seed von der Sekundenanzeige der Computeruhr ab, liefert also stets eine andere Sequenz von Zufallszahlen.

Klicken Sie auch in der Dokumentation den Namen der Methode `Binomial` an, um zunächst zu deren Erklärung und mit einem weiteren Klick zu deren C++-Code zu gelangen. Verstehen Sie diesen 7-zeiligen Code? (Hinweis: `Rndm()` erzeugt eine Zufallszahl zwichen 0 und 1.)

Aufgabe 3 (2 Punkte)

Erzeugen Sie ein Histogramm mit der theoretischen Binomialverteilung für $n = 20$ und $p = 0.2$. Dazu müssen Sie für jedes $k = 0, \ldots, n$ mit der Formel für $B(k; n, p)$ ausrechnen, wie hoch die Wahrscheinlichkeit für diesen Wert von k ist. Anschließend müssen Sie das Ergebnis in den entsprechenden Kanal im Histogramm eintragen. Dafür steht Ihnen die Methode `SetBinContent(k,value)` zur Verfügung.

Bei der Berechnung von $B(k; n, p)$ müssen Sie Potenzen wie p^k ausrechnen. Dazu gibt es in C++ die Funktion `pow(p,k)`. Alternativ können Sie auch die Methode `Power(Double_t x, Double_t y)` der Klasse `TMath` von ROOT verwenden, deren Code allerdings einfach {`return pow(x,y)`} lautet.

Weiterhin gilt es, die Binomialkoeffizienten $\binom{n}{k}$ auszurechnen. Sie stehen in der Klasse `TMath` ebenfalls als Methode `Binomial(n,k)` zur Verfügung. Auch hier gibt es eine Alternative: Es gibt in `TMath` auch die Methode `Factorial(n)`, die Ihnen $n!$ berechnet. Schließlich können Sie auch leicht selbst eine kurze Funktion schreiben, mit der Sie $n!$ ausrechnen. Sie müssen diese Funktion in ROOT allerdings in eine eigene Datei schreiben, z.B. mit dem Namen `Factorial.C`. Dann würden Sie in ROOT zunächst mit `.L Factorial.C` diese Funktion laden, bevor Sie mit `.x PlotBinomial.C` Ihr Programm zur Erzeugung des Histogramms aufrufen, welches diese Funktion dann verwenden kann.

Aufgabe 4 (3 Punkte)

In einer Spielshow gibt es drei Türen. Hinter einer Tür wartet ein Gewinn, hinter den beiden anderen Türen befinden sich Nieten. Nachdem der Kandidat eine Tür ausgewählt hat, öffnet der Moderator, der weiß wo sich der Gewinn befindet, eine der Türen, die der Kandidat nicht gewählt hat. Es kommt eine Niete zum Vorschein.

Der Kandidat darf erneut eine der beiden übrigen Türen wählen. Sollte er bei der ursprünglichen Tür bleiben oder besser die andere Tür wählen, um seine Gewinnchance zu erhöhen?

Tipp: Sie können bei der Lösung der Aufgabe bedingte Wahrscheinlichkeiten verwenden!