Rechnernutzung in der Physik

Institut für Experimentelle Teilchenphysik
Institut für Theoretische Teilchenphysik

Prof. Dr. G. Quast, Prof. Dr. M. Steinhauser
Dr. A. Mildenberger, Dr. Th. Chwalek
http://comp.physik.kit.edu

WS2018/19 – Blatt 06
Prog. bis Di., 18.12.2018

Wiederholung CgDA

In diesem Aufgabenblatt werden einige aus der Veranstaltung „Rechnernutzung in der Physik“ bekannte Themen aufgegriffen. Es dient auch dazu, Ihre Arbeitsumgebung in *python* zu testen (s. dazu die Vorlesung „Empfohlene Arbeitsumgebung“ in der ersten Semesterwoche) und Ihre Fertigkeiten im Umgang mit dieser Sprache zu festigen.
Überprüfen Sie zunächst, ob das zu diesem Blatt mitgelieferte Beispiel *basicDistributions.py* in Ihrer Umgebung lauffähig ist.

Aufgabe 11: Binomialverteilung (*)

Das klassische Beispiel für die Binomialverteilung ist der mehrfache Wurf einer Münze. Bei jedem einzelnen Wurf wird – idealerweise – mit gleicher Wahrscheinlichkeit das Ergebnis „Kopf“ bzw. „Zahl“ erwartet. Man zählt nun, wie oft das Ergebnis „Zahl“ bei einer Gesamtanzahl von N Würfen auftritt. Die Wahrscheinlichkeit, bei N Würfen mit fairer Münze ($p = \frac{1}{2}$) insgesamt k Mal das Ergebnis „Zahl“ zu erhalten, wird durch die Binomialverteilung $B(k; p, N)$ beschrieben. Wenn man ein solches Experiment im Computer oft wiederholt, kann man die Häufigkeit des Auftretens des Ergebnisses k, $H(k)$, bestimmen und mit der Binomial-Verteilung vergleichen.

a) Computerexperiment

Führen Sie ein solches „Computer-Experiment“ nun 1000 mal für jeweils 10 Münzwürfe durch. Bestimmen Sie die Häufigkeit für das Auftreten von $0, 1, 2, \ldots, 10$ mal „Zahl“. Stellen Sie die Häufigkeitsverteilung grafisch dar.
Hinweis: Nutzen Sie die Funktion `numpy.random.rand(N)` zum Erzeugen eines `arrays` R von N zwischen 0 und 1 gleichverteilten Zufallszahlen. Erzeugen Sie durch eine geeignete Operation aus dem so erhaltenen `array` von Zahlen ein `array`, das nur noch die Werte 0 für das Ergebnis „Kopf“ und 1 für „Zahl“ enthält. Die Anzahl k der „Zahl“-Ereignisse erhält man dann als Summe aller `array`-Einträge. Speichern Sie das Ergebnis k_i für das i-te Ihrer insgesamt 1000 Experimente in einem weiteren `array` mit je einem Eintrag pro Computerexperiment und verwenden Sie zur Darstellung der Häufigkeitsverteilung die Funktion `hist()` aus `matplotlib.pyplot`.

b) Vergleich mit der Binomial-Verteilung

Tragen Sie die Binomialverteilung als Funktion in die unter **a)** erzeugte Grafik ein und vergleichen Sie.

c) Erwartungswert und Varianz

Bestimmen Sie nun den Erwartungswert und die Varianz der Häufigkeitsverteilung von k. Vergleichen Sie mit den für die Binomialverteilung bekannten Werten für Erwartungswert und Varianz.

Hilfe: Binomial-Verteilung:

$$P(k; p, n) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k} \;, k = 1, \ldots, n; \quad E[k] = np; \quad V[k] = np(1-p)$$

Aufgabe 12: Darstellung von Messdaten und Vergleich mit einer Modellfunktion (*)

Die Datei `A12_data.dat` enthält Datenpunkte, organisiert in Spalten für den x- bzw. y-Wert. Die y-Werte haben eine voneinander unabhängige, gaußverteilte Unsicherheit von 0.2.

a) Lesen Sie die Daten in ein *python*-Script ein und stellen Sie sie als Datenpunkte mit Fehlerbalken dar. Sie können als Vorlage das Beispiel-Skript `read_xydata.py` verwenden.

b) Passen Sie nun eine Gerade an die Daten an. In diesem einfachen Fall ist die Funktionalität der Funktion `curve_fit` aus dem Paket `scipy.optimize` ausreichend, das eine χ^2-Anpassung ausführt. Ein Anwendungsbeispiel zur Anpassung eines Polynoms zweiten Grades finden Sie in der Datei `curvefit_example.py`; diesen Code können Sie für Ihr Beispiel anpassen.

c) Als letzten Schritt überprüfen Sie, ob die Daten durch die Gerade gut beschrieben werden. Wie Sie sicher noch wissen, verwendet man dazu den Wert der Größe χ^2 am Minimum:

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - y_i^{model})^2}{\sigma_i^2}$$

Diese Größe folgt einer χ^2-Verteilung mit einer Anzahl von Freiheitsgraden, die die Anzahl der Datenpunkte verringert um die Anzahl der angepassten Parameter ist. Eine Implementierung der χ^2-Verteilung finden Sie im Paket scipy.stats. Die Wahrscheinlichkeit, einen größeren Wert von χ^2 am Minimum als den tatsächlich beobachteten zu erhalten, berechnen Sie über die kumulative Verteilung `chi2prb = 1. - scipy.stat.chi2.cdf( chi, ndf)`.

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