Praktische Übung: Computergestützte Datenauswertung

Institut für Experimentelle Teilchenphysik

Prof. Dr. G. Quast
Dr. Andreas Poenicke
http://comp.physik.kit.edu

SS16 – Blatt 03
zu bearbeiten am 30.5. (Gr. a) bzw. 6.6. (Gr. b)

Die richtige Bearbeitung der Pflichtaufgaben wird von den Tutoren während der Übungszeit testiert. Für die Vergabe der Leistungspunkte ist die erfolgreiche Bearbeitung von 50% der Pflichtaufgaben erforderlich, von denen es auf insgesamt vier Blättern jeweils drei geben wird.

Aufgabe 3.1: Darstellen von Funktionen — Testat

In dieser Aufgabe soll eine Schar von Funktionen mit Hilfe von `numpy / scipy / matplotlib` dargestellt werden. Als Beispiel zur Verwendung sehen Sie sich bitte die Vorlesungsbeispiele und insbesondere das Programm `PlotBeispiel.py` von der Webseite an, das Sie als Vorlage für diese Aufgabe modifizieren können.

Aufgabe: Zeichnen Sie eine Schar von Resonanzkurven eines mit einer harmonischen Kraft $F_0 \sin(\omega t)$ angetriebenen harmonischen Oszillators mit Resonanzfrequenz ω_0, statischer Amplitude $A_s = F_0/m$ und Dämpfung $D = \gamma/\omega_0$, die Sie aus der klassischen Mechanik kennen,

$$A(\eta = \omega/\omega_0, A_s, D) = \frac{A_s}{\sqrt{(1-\eta^2)^2 - (2\eta D)^2}}$$

Stellen Sie für $A_s = 1.0$ im Bereich $\eta \in [0., 3.]$ eine Schar von Resonanzkurven mit verschiedenen Parametern $D \in [0.05, 3.]$ grafisch dar.

Hilfe: In der Vorlage wird eine `python`-Funktion `MyFunction()` zur Berechnung der Y-Werte aus einem `numpy-array` von X-Werten definiert. Diese Funktion sollten Sie modifizieren, so dass die Werte der Resonanzkurven für die entsprechenden Parameter berechnet werden.

Hinweis: Da Sie recht viele Kurven einzeichnen wollen, empfiehlt es sich, die Werte der gewünschten Parameter aus einer Liste zu entnehmen, z. B. für $D = 0.1$, $D = 1.0$ und $D = 3.0$:

```
for D in (0.1, 1., 3.0):
    Y = ...
    plt.plot(...)
```

Anmerkung: Das Script zur Lösung dieser Aufgabe ergibt eine recht allgemeine Vorlage, die Sie auch später zur Darstellung von Kurven immer wieder verwenden (und ggf. verfeinern) können.

Aufgabe 3.2: Berechnung statistischer Größen — Testat

In dieser Aufgabe implementieren Sie die Berechnung einiger der statistischen Größen, die Sie bereits aus der Vorlesung kennen.

Sie haben ein Python-Skript (`basic-statistics.py`) als Vorlage, und eine Datei mit Daten (`numbers.dat`). In der Vorlage werden aus der Datei die Daten (ganze Zahlen im Bereich $0 \ldots 9$) eingelesen und dann mit Hilfe von `numpy`-Funktionen die statistischen Größen Mittelwert, Varianz und Standardabweichung berechnet und auf dem Bildschirm ausgegeben.

a) Eigene Implementierung von mean(), variance() und sigma()

Implementieren Sie Ihre eigene Version zur Berechnung von Mittelwert, Varianz und Standardabweichung, indem Sie die vorbereiteten Definitionen der Funktionen in der Vorlage ergänzen. Vergleichen Sie Ihre Ergebnisse mit denen, die mit Hilfe der `numpy`-Funktionen in der Vorlage berechnet wurden.

b) Häufigkeitsverteilung

Bestimmen Sie nun die Häufigkeit der einzelnen Zahlen in der Datei `numbers.dat`, d. h. füllen Sie einen `numpy array` mit der jeweiligen Häufigkeit, mit der die Zahlen 0 bis 9 vorkommen. Dieser `array` enthält nun statt der ursprünglichen Datenmenge nur noch 10 Zahlen, die eine erheblich komprimiertere Version der ursprünglichen Daten darstellt und außerdem einen sehr viel besseren Überblick erlaubt.

c) Berechnung von Mittelwert und Varianz aus der Häufigkeitsverteilung
Berechnen Sie nun den Mittelwert und die Varianz der Daten direkt aus der Häufigkeitsverteilung. Vergleichen Sie mit dem Ergebnis, das Sie in a) erhalten haben.

Aufgabe 3.3: Funktionen von Zufallszahlen **Testat**

Zu dieser Aufgabe gibt es ein nützliches Beispielprogram, `Gauss.py`, die Sie zunächst anschauen und verstehen sollten. Das sehr kurze erste Beispiel zeigt, wie einfach man normalverteilte Zufallszahlen erzeugen und mit Hilfe von Funktionen des Moduls `matplotlib` darstellen kann, das dazu die Methode `matplotlib.pyplot.hist()` bereit stellt.
Die Rückgabewerte sind die Inhalte der einzelnen Intervalle, die Intervallgrenzen und – hier nicht weiter interessierende – Grafikobjekte (`patches`). Zusätzlich zum Histogramm wird die im Kopfteil des Programms definierte Gauß-Funktion zum Vergleich eingezeichnet, verschönert mit Titel, Achsenbeschriftungen und der in LaTeX gesetzten Formel der Gauß-Kurve.

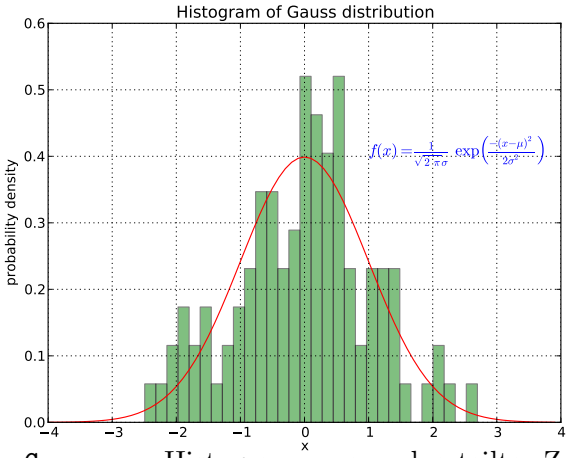

Abbildung 1: Ausgabe von `Gauss.py`: Histogramm normalverteilter Zufallszahlen mit Gauß-Kurve.

Aufgaben: a) Modifizieren Sie den Code in `Gauss.py` so, dass Sie gaußförmig verteilte Zufallszahlen mit beliebigem Mittelwert μ und Standardabweichung σ erzeugen können.
b) Zunächst studieren wir Summen von Gauß-verteilten Zufallszahlen. Bilden Sie 1000 Mittelwerte m_i ($m_i = \frac{1}{9}\sum_{j=1}^{9} x_j$) von je 9 normalverteilten (d.h. $\mu = 0$, $\sigma = 1$) Zufallszahlen x_j und tragen Sie diese in ein Histogramm ein. Vergleichen Sie die Verteilung der m_i mit der Verteilung mit der die x_j erzeugt wurden. Was beobachten Sie?
c) Erzeugen Sie nun zwei Datensätze mit von je 1000 Gauß-förmig verteilten Zufallszahlen x_i mit $(\mu_x, \sigma_x) = (1.5, 0.5)$ und y_i mit (μ_y, σ_y)=$(0.6, 0.15)$. Bilden Sie das Verhältnis von je zwei der beiden Zahlen, $v_i = x_i/y_i$, und stellen Sie das Histogramm der v_i grafisch dar. Ist die resultierende Verteilung noch Gauß-förmig?
Berechnen Sie mit Hilfe des Fehlerfortpflanzungsgesetzes die Standardabweichung σ_v der v_i und zeichnen Sie zum Vergleich eine Gauß-Kurve mit $(\mu_v = \mu_x/\mu_y, \sigma_v)$ ein. Passt das? Haben Sie eine Erklärung?
(Erinnerung:) Fehlerfortpflanzung: Der quadrierte relative Fehler eines Verhältnisses ist gegeben durch die quadratische Summe der relativen Fehler von Zähler und Nenner.
d) (freiwillig) Schreiben Sie eine kleine Funktion `histstat(binc,bine)`, die Mittelwert und Standardabweichung aus den von `matplotlib.pyplot.hist` zurückgegebenen Arrays berechnet und in die Grafik einträgt. (Eine solche Funktionalität ist allgemein sehr nützlich.)

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