Rechnernutzung in der Physik: Zusatz Python-Einführung

Institut für Experimentelle Teilchenphysik
Institut für Theoretische Teilchenphysik

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

WS2018/19 – Zusatzblatt[1] 02
Bearbeitungszeitraum: bis Di, 13.11.2018

Aufgabe 106: Berechnung statistischer Größen (*) **Testat**

In dieser Aufgabe implementieren Sie die Berechnung einiger der statistischen Größen.
Sie haben ein Python-Skript (`a106-basic-statistics.py`) als Vorlage, und dazu eine Datei mit Daten (`a106-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 `a106-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 107: Funktionen von Zufallszahlen (*) **Testat**

Zu dieser Aufgabe gibt es ein nützliches Beispielprogram, `a107-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.

Aufgaben:

(a) Modifizieren Sie den Code in `a107-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

[1] Die Python-Einführung (5 Aufgabenblätter) ist von denjenigen Teilnehmern zu bearbeiten, die die Rechnernutzung im Umfang von 6 LP absolvieren möchten. In diesem Teil sind 80% der Pflichtaufgaben erfolgreich zu bearbeiten.

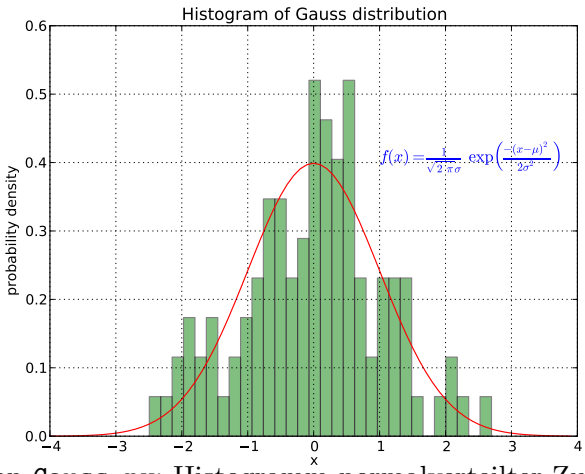

Abbildung 1: Ausgabe von `Gauss.py`: Histogramm normalverteilter Zufallszahlen mit Gauß-Kurve.

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.)
