Übungsserie 3: **04.09.2024**

Thema «Statistik mit Python» - Daten generieren, einlesen, vergleichen, analysieren und visualisieren

---

Aufgabe 1 – Daten visualisieren - Balkendiagramme

Es soll der Kaffeekonsum grafisch dargestellt werden. Dabei soll ein Balkendiagramm angewendet werden. Die Daten lauten wie folgt: Christian:7, Anita:11, Fabian:3, Peter:5, Bruno:2, Laura:2, Verena:9. Die Zahl nach dem Doppelpunkt bedeutet die konsumierte Portionenmenge Kaffee.

a) Erstelle ein Python-Programm, welches dir diese Daten visuell darstellt.
b) Berechne die Lage- (Arithmetischer Mittelwert, Median) und Streumasse (Spannweite, Standardabweichung.

Aufgabe 2 – Daten auswerten und vergleichen (AM / Median), Ausreisser

Es soll der Einfluss eines sogenannten «Ausreissers» festgestellt werden. Die Daten lauten wie folgt: [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68].

a) Berechne mit Hilfe des Moduls «statistics» das arithmetische Mittel und den Median.
b) Jetzt fügst du den Daten einen weiteren Wert «100.0» dazu und berechnest wiederum das arithmetische Mittel und den Median.
c) Was stellst du beim Vergleich fest?

Aufgabe 3 – Daten auswerten und vergleichen (Stand.Abw. / IQR)

Es soll der Einfluss eines sogenannten «Ausreissers» festgestellt werden. Die Daten lauten wie folgt: [8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68].

a) Berechne mit Hilfe der Module «statistics» und «scipy (stats)» die Standardabweichung und den Interquartilsabstand (IQA oder IQR).
b) Jetzt fügst du den Daten einen weiteren Wert «100.0» dazu und berechnest wiederum die Standardabweichung und den Interquartilsabstand (IQA oder IQR).
c) Was stellst du beim Vergleich fest?
d) Erstell nun noch einen Boxplot für beide Datensätze. Kannst du nun die berechneten Informationen und Unterschiede aus der Grafik herauslesen?

Aufgabe 4 – Daten generieren, speichern, einlesen, auswerten und visualisieren

a) Es sollen 1'000 zufällige, normalverteilte Zahlen generiert werden. Dazu kannst beispielsweise die in «numpy» eingebaute Methode «np.random.normal (*loc, scale*, *size*)» benutzen. Dabei bedeuten: *loc*: Mittelwert (Sollwert), *scale*: Standardabweichung und *size*: Anzahl Zufallszahlen (*n*). Als Beispiel kannst du folgende Vorgabewerte verwenden: *loc* = 25.5 und *scale* = 2.5. Die Anzahl ist bereits definiert.
b) Speichere diese Zufallszahlen in der Textdatei: «*daten_03_4_normal.txt*» ab. Verwende dazu die Methode «np.savetxt» aus «numpy». Vergewissere dich, mit einem Editor, dass diese 1'000 Zufallszahlen in die Datei geschrieben worden sind.
c) Jetzt liest du diese aus der soeben erstellten Datei ein und rundest die Werte auf eine Nachkommastelle mit «np.around» aus «numpy». Speichere die gerundeten Daten in der Datei «*daten_03_4_normal_gerundet.txt*» ab.
d) Berechne nun die folgenden Grössen der originalen und gerundeten Daten: Minimum, Maximum, Spannweite, Mittelwert und die Standardabweichung.

e) Stelle die gerundeten Daten als Histogramm dar. Nutze dabei die Methode «plt.hist()» aus «matplotlib.pyplot»

Aufgabe 5 – Daten einlesen, speichern, auswerten und visualisieren

a) Die in Aufgabe 4 generierten Daten sollen mit Hilfe der Methode «pd.read_csv()» aus dem Modul/Bibliothek «pandas», eingelesen werden.
b) Füge nun den Daten eine Spaltenüberschrift 'ZufallsZahl' hinzu und gib die ersten 5 Zeilen der Daten aus.
c) Da die Daten mit Hilfe der «pandas»-Methode «read_csv()» eingelesen worden sind, gibt es nun eine Möglichkeit, sich schnell einen Überblick über die Daten zu beschaffen. Der Befehl heisst «describe». Lass dir dadurch die wichtigsten Parameter anzeigen. Welche Bedeutung haben die einzelnen Parameter?
d) Berechne nun die folgenden Grössen der eingelesenen, gerundeten Daten: Minimum, Maximum, Spannweite, Mittelwert und die Standardabweichung und vergleiche sie mit den Werten aus Aufgabe 4. Was stellst du fest?
e) Erstelle mit Hilfe der Pandas-Methode ein Histogramm. Die Anzahl der Klassen k wird in der Regel durch die Formel $k = \lfloor\sqrt{n}\rfloor$ festgelegt.
f) Speichere die Daten mit der ergänzten Spaltenüberschrift und der Option «header=True» in der Datei «*daten_03_5_normal_ueberschrift.txt*» ab. Zudem soll das Dezimalzeichen ein Punkt «.» und das Trennzeichen für weiter Spalten ein «;» sein.

Aufgabe 6 – Daten auswerten, vergleichen, Verteilungen (Münze)

Eine Münze soll n-mal geworfen werden. Dabei soll jedes Ereignis «Kopf» oder «Zahl gleichwahrscheinlich auftreten.

a) Erstelle ein Pythonprogramm, welches dir diese Daten generiert und in die Datei «*daten_03_6_muenz_wurf.csv*» speichert.
b) Wenn du den ersten Teil auslassen willst, kannst du die Daten jetzt einfach einlesen. Analysiere diese Daten, zähle, wie oft die beiden Ereignisse «Kopf» und «Zahl» aufgetreten sind.
c) Erstelle ein Histogramm, welches diese Häufigkeiten visualisiert.
d) Vergleiche diese Häufigkeiten mit der erwarteten (theoretischen Wahrscheinlichkeit) absolut und relativ.

Aufgabe 7 – Daten auswerten, vergleichen, Verteilungen (Würfel)

In den beiliegenden Dateien haben fünf Personen mit einem Würfel (Augenzahlen 1 bis 6) gewürfelt. Die jeweiligen Ergebnisse sind in diesen Dateien gespeichert: «*daten_03_7_1.txt*», «*daten_03_7_2.txt*», «*daten_03_7_3.txt*», «*daten_03_7_4.txt*» und «*daten_03_7_5.txt*».

a) Untersuche diese Daten und kategorisiere sie.
b) Um welche Verteilungen handelt es sich bei den einzelnen Kategorien?
c) Visualisiere deine Resultate aus Teilaufgabe b), d.h. stelle die Häufigkeiten als Säulendiagramme dar.
d) Welche Würfel waren «gezinkt»?

Aufgabe 8 – Daten generieren, visualisieren und vergleichen, Verteilungen (QQ-Plot)

a) Erstelle durch die Methode «np.random.normal(0,1,1000)» 1000 Zufallszahlen mit dem Mittelwert 0 und der Standardabweichung 1. Damit wir jedes Mal die gleichen Werte erhalten, kannst du vorher, einen Initialwert für den Zufallszahlen-Generator festlegen «np.random.seed(0)»

b) Die Zufallszahlen, welche wir generiert haben, sollten normalverteilt sein. Was heisst das? Es gibt mehrere Möglichkeiten, von Daten festzustellen, ob sie normalverteilt sind oder nicht. Eine visuelle Möglichkeit ist der Q-Q-Plot. Erstelle mit der Methode «sm.qqplot()» ein solches Diagramm. Falls die Messpunkte angenähert eine Gerade darstellen, dann sind die Messwerte normalverteilt.
Hinweis: Es kann sein, dass du das Modul «statsmodels» zuerst installieren musst. Das heisst, du kannst in VS-Code das Terminal-Fenster anzeigen und dort den Befehl «pip install statsmodels» eingeben und ausführen. Danach sollte ein Import im JupyterNotebook möglich sein.

c) Ergänze in deinem Q-Q-Plot noch eine Gerade mit der Steigung 1 (Steigungswinkel ist 45°). Dies kannst du erreichen, indem du den Befehl «sm.qqplot()» durch einen Parameter erweiterst.

d) Gib jetzt noch das Histogramm der Daten aus.

Aufgabe 9 – Stichprobe - zufällige Auswahl von Elementen

Gegeben ist die Liste der Artikel: Badehosen, Langhosen, Kurzhosen, Wanderschuhe, Unterhosen, Skischuhe, Turnschuhe. Erstelle ein Pythonprogramm, welches dir jeweils 1, 2 oder 3 Artikel zufällig aus diesen auswählt. Dieser Vorgang soll dreimal wiederholt werden.

a) Bei jeder Auswahl eines Artikels wird dieser wieder zurückgelegt, so dass gleiche Artikel mehrfach vorkommen dürfen.

b) Nach der Auswahl eines Artikels in einem Vorgang, darf ein Artikel nicht mehrfach ausgewählt werden.

c) Wie können jeweils die Anzahl Möglichkeiten berechnet werden?

Variante 1: Mit Hilfe des Moduls «numpy».

Variante 2: Gibt es noch eine andere Möglichkeit? Versuche es mit den Modulen «itertools» und «math».

Aufgabe 10 – Zufallszahlen, Histogramm, Verteilungen

Es sollen je 1'000 Zufallszahlen generiert werden. Benutze dazu die entsprechenden Methoden aus dem Modul «scipy.stats» und für das Histogramm respektive für die Dichtefunktion das Modul «seaborn» und verwende die Standardeinstellungen.

a) Die Zufallszahlen sollen normalverteilt sein.

b) Die Zufallszahlen sollen gleichverteilt sein.

c) Die Zufallszahlen sollen einer Laplace-Verteilung genügen.

d) Stelle die Histogramme für die in a) bis c) berechneten Zufallszahlen nebeneinander dar.

e) Stelle die Density-Plots (Dichten) für die in a) bis c) berechneten Zufallszahlen nebeneinander dar. Für die Darstellung der sogenannten Dichtefunktion kannst du auf das Modul «seaborn» und die darin verfügbare Methode «kdeplot» verwenden.