ylymny5l4

May 23, 2025

1 Statistik-Labor Testat Aufgabe 3

Hinweise: - Bitte überprüfen Sie Ihre Resultate vor der Abgabe Ihre Ergebnisse mit den Teilergebnissen aus der Datei ***Teilergebnisse_xy.txt***. - Die Unterlagen sind im **pdf-Format** in Moodle hochgeladen abzugeben. Richtige Lösungen werden nicht mehr an Sie zurückgegeben. Eine Abgabe per Email ist nicht möglich. - Bitte füllen Sie das jeweilige **Deckblatt** mit aus und geben es mit Ihrer Lösung zusammen ab.

Aufgabenstellung: In dieser Aufgabe wird die Lotterie „KENO“ untersucht, die die staatlichen Lottogesellschaften täglich (von Montag bis Samstag) anbieten. Bei dieser Lotterie besteht ein Tippfeld aus 70 Zahlen. Ein:e Teilnehmer:in an der Lotterie kann selbst entscheiden, wie viele dieser Zahlen er/sie ankreuzt (mindestens 2, höchstens 10) und welchen Betrag er/sie einsetzt (1, 2, 5 oder 10 Euro). Von den 70 Zahlen werden 20 Gewinnzahlen gezogen. Je nachdem, wie viele der Gewinnzahlen er/sie angekreuzt hatte, bekommt der/die Lotterieteilnehmer:in einen festen Geldbetrag ausgezahlt. Der Gewinnplan (Stand 01.01.2005) ist im Folgenden aufgelistet. (Es gibt Sonderregelungen für die jeweils höchsten Gewinnklassen bei 10 oder 9 getippten Zahlen; diese sollen nicht berücksichtigt werden und sind daher nicht hier aufgeführt.)

Ablesebeispiel: Sie haben auf dem Spielschein in einem Tippfeld 7 Zahlen angekreuzt; Ihr Einsatz beträgt 2 €. Wenn Sie unter Ihren 7 Zahlen genau 5 Richtige sind, erhalten Sie $2 \cdot 12$ € = 24€ ausbezahlt (d. h. Ihr Gewinn beträgt 22 €). Wenn Sie stattdessen nur 3 Richtige haben, erhalten Sie nichts (d. h. Sie haben einen Verlust in Höhe Ihres Einsatzes von 2 €).

1.0.1 a) Tippfeld mit 10 Kästchen

Die Zufallsvariablen X_{10} beschreibt den Gewinn eines:r Lotterieteilnehmer:in, der/die in einem Tippfeld 10 Kästchen ankreuzt und 2 € einsetzt. Berechnen Sie Erwartungswert und Standardabweichung von X_{10}.

[9]:
```
from scipy.stats import hypergeom
import numpy as np

# KENO: 70 Zahlen insgesamt, davon werden 20 gezogen, und man tippt auf 10
N = 70   # Gesamtanzahl der Zahlen (Grundgesamtheit)
K = 20   # Anzahl der gezogenen Gewinnzahlen (Erfolge in der Population)
n = 10   # Anzahl der getippten Zahlen pro Spiel (Stichprobenumfang)
dist = hypergeom(N, K, n)  # Hypergeometrische Verteilung für Ziehung ohne␣
 ↪Zurücklegen
```

```python
# Brutto-Auszahlungen laut Gewinnplan bei 2 € Einsatz abzüglich Einsatz (-2 €)
# Key = Anzahl richtiger Tipps, Value = Netto-Auszahlung (inkl. Verlust bei 0-4␣
↪Treffern)
gewinne = {
    10: 100000 * 2 - 2,
    9: 1000 * 2 - 2,
    8: 100 * 2 - 2,
    7: 15 * 2 - 2,
    6: 5 * 2 - 2,
    5: 2 * 2 - 2,
    4: -2,   # kein Gewinn, nur Verlust des Einsatzes
    3: -2,
    2: -2,
    1: -2,
    0: 2 * 2 - 2  # Sonderfall: bei 0 Treffern gibt es minimalen Gewinn
}

# Array der möglichen Trefferzahlen von 0 bis 10
x = np.arange(0, 11)

# Wahrscheinlichkeiten für jede mögliche Anzahl an Treffern
wahrscheinlichkeiten = dist.pmf(x)

# Passende Netto-Auszahlung für jede Anzahl an Treffern
auszahlungen = np.array([gewinne[i] for i in x])

# Erwartungswert des Gewinns: Summe aus (Wahrscheinlichkeit * Auszahlung)
mu = np.sum(wahrscheinlichkeiten * auszahlungen)

# Varianz: mittlere quadratische Abweichung vom Erwartungswert
varianz = np.sum(wahrscheinlichkeiten * (auszahlungen - mu)**2)

# Standardabweichung: Wurzel aus der Varianz
stdabw = np.sqrt(varianz)

# Ausgabe der Ergebnisse
print(f"Erwartungswert     von X10: {mu:.4f}")
print(f"Varianz            von X10: {varianz:.4f}")
print(f"Standardabweichung von X10: {stdabw:.4f}")
```

```
Erwartungswert     von X10: -1.0120
Varianz            von X10: 18735.7766
Standardabweichung von X10: 136.8787
```

1.0.2 b) Weitere Tippfelder

Berechnen Sie ebenso die Erwartungswerte und Standardabweichungen von $X_9, ..., X_2$, d. h. dem Gewinn bei Ankreuzen von 9 (bzw. 8, ..., 2) Kästchen in einem Tippfeld jeweils bei Einsatz von

2 €.

Beispiellösung: - Abgabe für a) und b) mit vollständiger Berechnung (nicht nur Endergebnisse) - am liebsten in tabellarischer Form für $X_{10}, ..., X_2$

[10]:
```python
import numpy as np
from scipy.stats import hypergeom
import pandas as pd

# Grunddaten für KENO
N = 70   # Gesamtanzahl möglicher Zahlen (Population)
K = 20   # Anzahl gezogener Gewinnzahlen (Treffer in Population)

# Gewinnplan für unterschiedliche Anzahlen getippter Zahlen
# Jeder Schlüssel (z.B. 10, 9, ..., 2) steht für n getippte Zahlen
# Die inneren Dictionaries geben für k Richtige die Auszahlungen an
# Auszahlung = (brutto Gewinn * 2 Euro Einsatz) - 2 Euro Einsatz
gewinne_dict = {
    10: {10: 100000 * 2 - 2, 9: 1000 * 2 - 2, 8: 100 * 2 - 2, 7: 15 * 2 - 2, 6: 5 * 2 - 2, 5: 2 * 2 - 2, 4: -2, 3: -2, 2: -2, 1: -2, 0: 2 * 2 - 2},
    9: {9: 50000 * 2 - 2, 8: 1000 * 2 - 2, 7: 20 * 2 - 2, 6: 5 * 2 - 2, 5: 2 * 2 - 2,
        4: -2, 3: -2, 2: -2, 1: -2, 0: 2 * 2 - 2},
    8: {8: 10000 * 2 - 2, 7: 100 * 2 - 2, 6: 15 * 2 - 2, 5: 2 * 2 - 2, 4: 1 * 2 - 2,
        3: -2, 2: -2, 1: -2, 0: 1 * 2 - 2},
    7: {7: 1000 * 2 - 2, 6: 100 * 2 - 2, 5: 12 * 2 - 2, 4: 1 * 2 - 2, 3: -2,
        2: -2, 1: -2, 0: -2},
    6: {6: 500 * 2 - 2, 5: 15 * 2 - 2, 4: 2 * 2 - 2, 3: 1 * 2 - 2, 2: -2, 1: -2, 0: -2},
    5: {5: 100 * 2 - 2, 4: 7 * 2 - 2, 3: 2 * 2 - 2, 2: -2, 1: -2, 0: -2},
    4: {4: 22 * 2 - 2, 3: 2 * 2 - 2, 2: 1 * 2 - 2, 1: -2, 0: -2},
    3: {3: 16 * 2 - 2, 2: 1 * 2 - 2, 1: -2, 0: -2},
    2: {2: 6 * 2 - 2, 1: -2, 0: -2},
}

# Liste zur Aufnahme aller Auswertungen für n = 10 bis 2 getippte Zahlen
daten = []

# Schleife über alle Tippvarianten von 10 bis 2 Zahlen
for n in range(10, 1, -1):
    dist = hypergeom(N, K, n)  # Hypergeometrische Verteilung für n getippte Zahlen
    gewinne = gewinne_dict.get(n, {})  # Zuordnung der Gewinnauszahlungen

    x = np.arange(0, n + 1)  # Alle möglichen Treffer von 0 bis n
    wahrscheinlichkeiten = dist.pmf(x)  # Wahrscheinlichkeiten für genau x Treffer
```

```
    auszahlungen = np.array([gewinne.get(i, -2) for i in x])  # Zu jeder␣
↪Trefferanzahl die Auszahlung

    # Erwartungswert = Summe (Wahrscheinlichkeit * Auszahlung)
    mu = np.sum(wahrscheinlichkeiten * auszahlungen)

    # Varianz = Summe der quadrierten Abweichungen vom Erwartungswert
    varianz = np.sum(wahrscheinlichkeiten * (auszahlungen - mu) ** 2)
    stdabw = np.sqrt(varianz)  # Standardabweichung = Wurzel aus Varianz

    # Ergebnis für diese Tippzahl speichern
    daten.append({
        "n getippte Zahlen": n,
        "Erwartungswert": round(mu, 4),
        "Varianz": round(varianz, 4),
        "Standardabweichung": round(stdabw, 4)
    })

# In DataFrame zur tabellarischen Darstellung überführen
df = pd.DataFrame(daten)

# Ausgabe der Tabelle in Jupyter/Notebook
df  # zeigt die Ergebnisse für 9 bis 2 getippte Zahlen
```

[10]:

	n getippte Zahlen	Erwartungswert	Varianz	Standardabweichung
0	10	-1.0120	18735.7766	136.8787
1	9	-0.9991	26218.0757	161.9200
2	8	-1.0212	5358.7205	73.2033
3	7	-1.0087	331.7899	18.2151
4	6	-1.0051	301.3534	17.3595
5	5	-1.0020	56.0134	7.4842
6	4	-1.0111	11.2623	3.3559
7	3	-0.9865	20.9925	4.5818
8	2	-1.0559	10.4379	3.2308

1.0.3 c) Gewinnchance-Maximierung

Wie viele Kästchen pro Tippfeld sollte ein:e KENO-Spieler:in ankreuzen, der/die den Erwartungswert seines/ihres Gewinns maximieren möchte?

[11]:
```
# 3 Tippfelder, da man den Höchsten Erwartungswert hat
# 3 Tippfelder -0.9865 > 9 Tippfelder -0.9991 :)))))))
```

1.0.4 d) Individual Aufgabe

Beantworten Sie die in Datei **sr__aufg__3d__xy.txt** im Unterverzeichnis **Endziffer__xy** gestellte Frage (*xy = Endziffern Ihrer Matrikelnummer*). Welche Funktion verwenden Sie hier? Welche

Werte muss man für die Parameter einsetzen?

Tipp: - Die Datei sr_aufg_3d_xy.txt ist nur aus technischen Gründen im .txt-Format abgespeichert. Sie muss nicht in das Notebook eingelesen werden, sondern kann auch mit Microsoft Word, WordPad oder dem Editor gelesen werden. - Es treten (je nach Matrikelnummer) Formulierungen wie "**höchstens** 9-mal", "**mindestens** 9-mal" oder "**genau** 9-mal" auf. Verwechseln Sie diese nicht!

[12]:
```
# Endziffer 54
# Testat-Aufgabe sr_aufg_3 Aufgabenteil d)

# Sie spielen an 16 Tagen KENO. Dabei kreuzen
# Sie jeden Tag in einem Tippfeld 5 K stchen an.
# Ihr Einsatz betr gt jeden Tag 2 Euro.

# Wenn Sie an einem Tag 0, 1 oder 2 Richtige haben,
# erhalten Sie von der Lottogesellschaft nichts ausbezahlt,
# d. h. Sie verlieren Ihren Einsatz von 2 Euro.

# Wie wahrscheinlich ist es, dass dies an den 16 Tagen
# genau 12-mal vorkommt?

# Mit anderen Worten: Wie wahrscheinlich ist es, dass Sie
# bei 16 Spielen genau 12-mal einen Verlust
# von 2 Euro haben?

from math import comb
from scipy.stats import binom

# Schritt 1: Verlustwahrscheinlichkeit berechnen mit Hypergeometrie
def hypergeo(k, N=70, K=20, n=5):
    return comb(K, k) * comb(N - K, n - k) / comb(N, n)

# Einzelwahrscheinlichkeiten für 0, 1, 2 Richtige
p_0 = hypergeo(0)
p_1 = hypergeo(1)
p_2 = hypergeo(2)

# Gesamtwahrscheinlichkeit für Verlust (0, 1 oder 2 Richtige)
p_loss = p_0 + p_1 + p_2

# Schritt 2: Binomialwahrscheinlichkeit für genau 12 Verluste bei 16 Spielen
n_spiele = 16
k_verluste = 12

p_genau_12_verluste = binom.pmf(k_verluste, n_spiele, p_loss)
```

```python
print(f"Wahrscheinlichkeit für genau 12 Verluste in 16 Spielen: {p_genau_12_verluste:.4f}")
#oder halt 10,89%
# 10,89% Wahrscheinlichkeit
```

```
Wahrscheinlichkeit für genau 12 Verluste in 16 Spielen: 0.1089
```

[ ]:

```python

```

[ ]:

```python

```