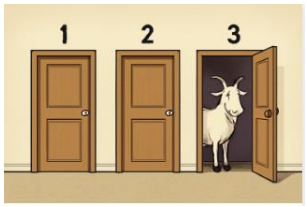
Hubert Langlotz
Wilfried Zappe

Das Ziegenproblem - Eine denkwürdige Aufgabe zur Stochastik

In einer Fernsehshow bietet sich für den Kandidaten hinter einer von drei Türen ein Auto als Hauptpreis. Der Kandidat wählt im ersten Schritt eine der drei Türen, worauf der Moderator, der weiß, hinter welcher Tür sich das Auto befindet, eine der beiden anderen Türen öffnet und die dahinter befindliche Ziege dem Kandidaten zeigt. Soll der Kandidat nun im 2. Schritt seine Wahl der Tür beibehalten oder zur anderen Tür wechseln, um seine Gewinnaussichten zu steigern?



(Bild: ChatGPT)

„Die Aufgabe wurde zuerst von dem Biostatistiker Steve Selvin formuliert. Er stellte sie 1975 im American Statistician in einem Leserbrief vor“[1]**.** Das ist also, folgt man dieser Angabe in WIKIPEDIA (es gibt auch Quellen, die andere historische Daten angeben), ca. 50 Jahre her und gibt Anlass, dieses Problem aufzugreifen.
Hierzu werden im Folgenden verschiedene Angebote für eine unterrichtliche Umsetzung beschrieben

- Hinweise zu einer Vorgehensweise im Unterricht
- Simulation mit dem TI-30X Prio MathPrint™ und eine theoretische Lösung ergänzt um das $\frac{1}{\sqrt{n}}$ –Gesetz und Konfidenzintervalle
- Simulation mit der TI-Nspire™ CX Technologie
- Simulation mit einem Basic- bzw. Python-Programm mit der TI-Nspire™ CX Technologie

1. Hinweise zu einer Vorgehensweise im Unterricht

Bevor eine Lösung mit technischen Hilfsmitteln gesucht wird, sollte man das Problem erst einmal wirken lassen. Dazu bietet sich im ersten Schritt eine Vermutungsphase an. Denn das Ziegenproblem ist ein sehr schönes Beispiel für eine Contra-Intuition und nicht wenige Lernenden vermuten die Wahrscheinlichkeit bei 50%. Dieses lädt zu einer oft kontroversen Diskussion ein. Zur Klärung sollte in einem zweiten Schritt die Spielsituation mit einer gegenständlichen Simulation greifbar gemacht werden. Dazu kann man Gruppen bilden, in denen ein Spieler, ein Spielleiter, ein Protokollant und ein Beobachter ernannt werden. Die Gruppen bekommen jeweils drei nummerierte Becher, drei Spielfiguren (die beiden Ziegen und das Auto können abstrahiert werden). Dabei verfolgen die Gruppen unterschiedliche Strategien: ein Teil der Gruppen wechselt strikt, ein anderer Teil der Gruppen behält die erste Entscheidung strikt bei. Ca. 20 Simulationen sollten durchgeführt und protokolliert werden. Hierbei sollten sich schon aussagekräftige Ergebnisse zeigen.
Um die Anzahl der Simulationen noch weiter zu erhöhen, können nun technologische Hilfsmittel eingesetzt werden.

[1] https://de.wikipedia.org/wiki/Ziegenproblem

© T3 Deutschland 2025

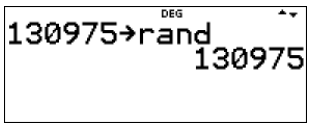
2. Eine Simulation mit dem WTR TI-30X Prio MathPrint™ *(Wilfried Zappe)*

Vorüberlegungen:
Fall 1: Angenommen, der Kandidat wählt zufällig die Tür aus, hinter der das Auto steht. Dann wird der Moderator eine der beiden anderen Türen öffnen. Der Kandidat gewinnt nur dann, wenn er nicht wechselt.
Fall 2: Angenommen, der Kandidat wählt zufällig eine Tür aus, hinter der das Auto nicht, sondern eine der beiden Ziegen steht. Der Moderator muss dann die andere „Ziegentür" öffnen. Der Kandidat gewinnt nur dann, wenn er wechselt.

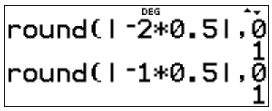
Simulation:
Die Türen seien mit den Zahlen 1, 2 und 3 nummeriert.
Die Tür, hinter der das Auto steht, wird durch eine ganzzahlige Zufallszahl 1, 2 oder 3 bestimmt: randint(1,3)
Die Tür, die der Kandidat zunächst wählt, wird ebenfalls durch eine ganzzahlige Zufallszahl 1, 2 oder 3 bestimmt: randint(1,3)
Nun wird durch Differenzbildung für die beiden oben genannten Fälle untersucht, ob diese beiden Zufallszahlen gleich sind oder nicht.
Die Differenz der beiden Zufallszahlen enthält genau einen der folgende Werte: – 2 ; – 1 ; 0 ; 1 oder 2. Dabei bedeutet nur der Eintrag 0, dass der Kandidat zu Beginn das richtige Tor ausgewählt hat.

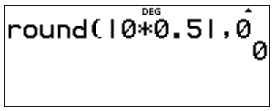
- Wenn der Kandidat also bei seiner Wahl bleibt (Fall 1), gewinnt er nur, falls eine Null vorkommt.
- Wenn der Kandidat also das Tor wechselt (Fall 2), gewinnt er, falls eine Zahl ungleich 0 steht.

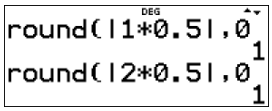
Umsetzung im Listeneditor:



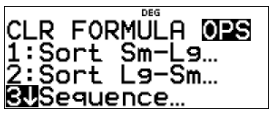
Zunächst wird der Zufallsgenerator durch Eingabe einer individuell gewählten positiven ganzen Zahl und Drücken von [sto] mit *rand* ([2nd][!nCr][1]) und [enter] initialisiert.
In der Liste L1 wird eine Folge der Differenzen der beiden Zufallszahlen $\text{randint}(1,3) - \text{randint}(1,3)$ erzeugt. In der Liste L2 wird zu jedem Wert aus der Liste L1 über den Befehl $round(abs(L1 \cdot 0{,}5), 0)$ eine Folge generiert, die nur die Elemente 0 und 1 enthält. Das Element 0 entsteht in der Liste L2 genau dann, wenn in L1 die zugehörige Differenz den Wert 0 hat. In allen anderen Fällen steht dann in der Liste L2 der Wert 1.

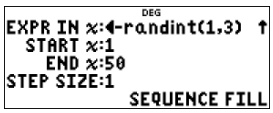


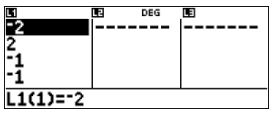




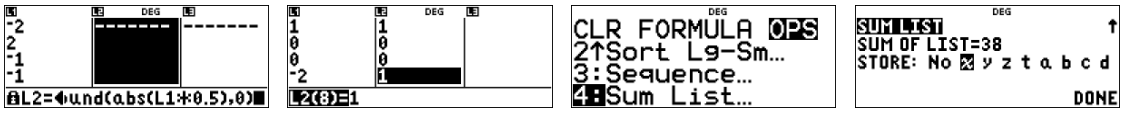
Mit [data][data] OPS 3 Sequence wird die Liste L1 mit $\text{randint}(1,3) - \text{randint}(1,3)$ für die maximal möglichen 50 Listenelemente generiert.





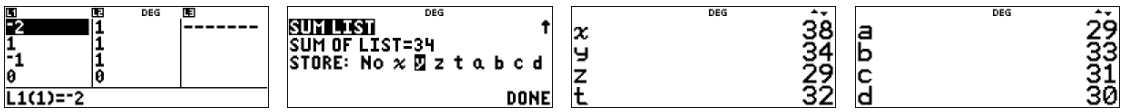


In der Liste L2 wird zunächst mit der Taste [sto] die Eingabe des Befehls $round(abs(L1 \cdot 0{,}5), 0)$ vorbereitet und dann mit Hilfe der Taste [math] NUM realisiert. Abschließend wird die Summe der Liste L2 mit [data] OPS 4 Sum List gebildet und das Ergebnis unter der Variablen x gespeichert.

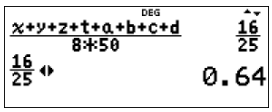


Diese Simulation zeigt also, dass sich in 38 von 50 Fällen der Wechsel lohnen würde. Das entspricht einer relativen Häufigkeit von $\frac{38}{50} = 0{,}76$.

Mit [data][data] OPS 3 Sequence wird die Liste L1 mit $\text{randint}(1,3) - \text{randint}(1,3)$ erneut generiert. Die Anpassung der Liste L2 erfolgt automatisch. Die Summenbildung muss neu veranlasst werden. Das Ergebnis wird unter der Variablen y gespeichert.

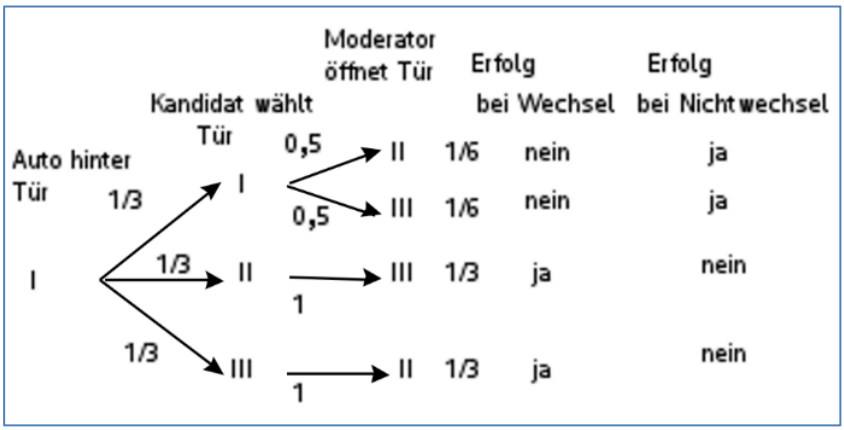


Die neue Simulation zeigt also, dass sich in 34 von 50 Fällen der Wechsel lohnen würde. Das entspricht einer relativen Häufigkeit von $\frac{34}{50} = 0{,}68$. Die Simulation könnte nun noch sechsmal wiederholt werden. Die Ergebnisse werden unter den Variablen z, t, a, b, c und d gespeichert. Abschließend wird die Summe der Variablen gebildet und die relative Häufigkeit der Gesamtanzahl der Einsen berechnet. Die Anzahl der Gesamtsimulationen wäre dann $n = 8 \cdot 50 = 400$.



Man erkennt an den relativen Häufigkeiten, dass im Falle des Wechselns die Gewinnchance mit 64% ungefähr doppelt so hoch ist wie im Falle des Nichtwechselns.

Theoretischer Weg



Aus dem Vorgehen wird auch ein Ansatz für eine theoretische Lösung deutlich, die man dem Baumdiagramm[1] entnehmen kann. Die Türen sind hier nicht mit 1,2 und 3, sondern mit I, II und III bezeichnet, um sie besser von den anderen Zahlenangaben unterscheiden zu können. Das Ergebnis der Simulation (0,64) steht näherungsweise in Übereinstimmung mit der theoretischen Lösung $\left(\frac{2}{3} \approx 0{,}67\right)$.

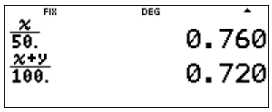
[1] Das Baumdiagramm wurde mit der TI-Nspire™ CX Technologie und dem TI-Widget-Programm „Physics-Drawing“ erstellt.

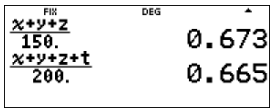
© T3 Deutschland 2025

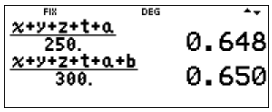
Anmerkungen zum $\frac{1}{\sqrt{n}}-$ Gesetz

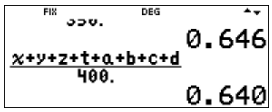
Etwas Theorie:
Man kann aufgrund der 2σ –Regel vor der Durchführung eines Bernoulliversuchs voraussagen, dass bei einer bekannten oder als wahr angenommenen Wahrscheinlichkeit p die relative Häufigkeit h des untersuchten Merkmals in einer Stichprobe mit dem Umfang n mit großer Wahrscheinlichkeit (ca. 95,4 %) im Zwei-Sigma-Prognoseintervall $\left[p-2\cdot\frac{\sqrt{p\cdot(1-p)}}{\sqrt{n}};p+2\cdot\frac{\sqrt{p\cdot(1-p)}}{\sqrt{n}}\right]$ liegen wird. Die Länge dieses Intervalls ist $d(n)=4\cdot\frac{\sqrt{p\cdot(1-p)}}{\sqrt{n}}$. Weil $d(n)$ mit wachsendem n mit dm Faktor $\frac{1}{\sqrt{n}}$ schrumpft, spricht man auch vom $\frac{1}{\sqrt{n}}-$ Gesetz.[1]

Zurück zum Ziegenproblem:
Die bisher erhobenen Daten sind geeignet, dieses Gesetz zu veranschaulichen.
Mit den gespeicherten relativen Häufigkeiten lassen sich durch Summenbildung die relativen Häufigkeiten der Simulationsergebnisse nach 50, 100, …, 400 Simulationen ermitteln.

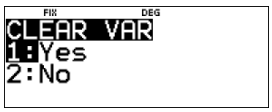


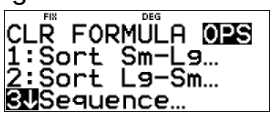


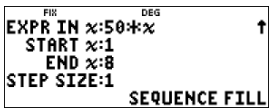


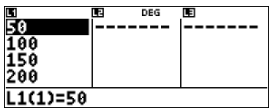


Für $p=\frac{2}{3}$ und n = 50, n = 100, …, n= 400 werden nun die untere und die obere Grenze der Prognoseintervalle mit dem Listeneditor des TI-30X Prio MathPrint™ berechnet. Vorher müssen die Variablen gelöscht werden. In der Liste L1 werden die Werte für n berechnet.

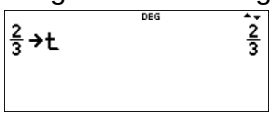


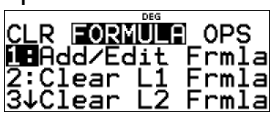


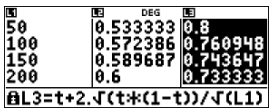


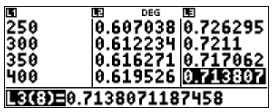


Der Wert für p wird unter der Variablen t gespeichert. In der Liste L2 wird die untere Grenze g_u des zugehörigen Prognoseintervalls berechnet, in L3 werden die oberen Grenzen g_o der Prognoseintervalle gespeichert.







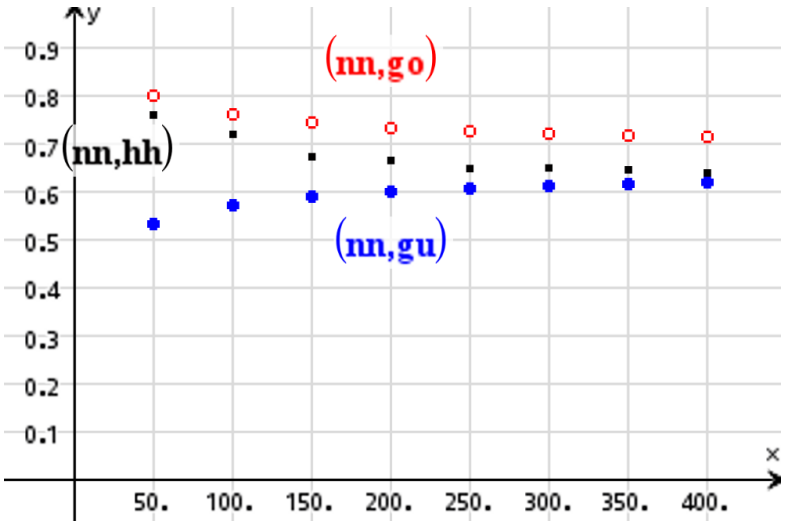


In der Tabelle sind die Ergebnisse zusammengefasst:

nn	50	100	150	200	250	300	350	400
hh	0,760	0,720	0,673	0,665	0,648	0,650	0,646	0,640
g_u	0,533	0,572	0,590	0,600	0,607	0,612	0,616	0,620
g_o	0,800	0,761	0,744	0,733	0,726	0,721	0,717	0,714

[1] Vgl. Wilfried Zappe :Handreichung zu Prognoseintervallen. Thüringer Institut für Lehrerfortbildung, Lehrplanentwicklung und Medien, Bad Berka, März 2019 https://www.schulportal-thueringen.de/web/guest/media/detail?tspi=2942&tspt=%3A%3B%3AbackUrl%3A%3D%3A%2Fmedia%2Fdetail%3Ftspi%3D1392

Die Daten werden grafisch dargestellt.[1]



Man erkennt gut den „$\frac{1}{\sqrt{n}}$ –Trichter“, der u. a. die Merkhilfe „eine Vervierfachung des Stichprobenumfangs halbiert die Länge des Prognoseintervalls“ veranschaulicht.
Alle durch Simulation ermittelten relativen Häufigkeiten liegen innerhalb des jeweiligen 95,4%-Prognoseintervalls.

Konfidenzintervall mit dem TI-30X Prio MathPrint ™ ermitteln

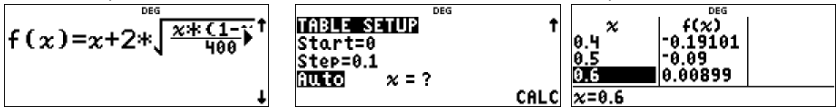
Es sind die Grenzen für p zu bestimmen, für die h im 95,4%-Prognoseintervall $\left[p - c \cdot \sqrt{\frac{p\cdot(1-p)}{n}} \leq h \leq p + c \cdot \sqrt{\frac{p\cdot(1-p)}{n}}\right]$ mit c = 2 liegt.

Die Grenzen des Konfidenzintervalls erhält man mit einer Sicherheit von etwa 95,4%, über die Zwei Sigma-Regel, indem man die folgenden Gleichungen für p löst:

$$p \pm 2 \cdot \sqrt{\frac{p\cdot(1-p)}{n}} = h \Leftrightarrow p \pm 2 \cdot \sqrt{\frac{p\cdot(1-p)}{n}} - h = 0$$

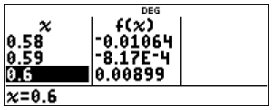
Diese Gleichungen lassen sich näherungsweise mit dem TI-30X Prio MathPrint™ mithilfe von [table] lösen. Man gibt die linke Seite jeder Gleichung als Funktion ein und sucht die Intervalle, in denen die Funktion das Vorzeichen wechselt, in dem also die Nullstelle von f liegt. Wegen 0 < p < 1 kann man mit einer Schrittweite von 0,1 beginnen. Durch Verkleinerung der Schrittweite in der Tabelle werden die vorzeichenwechselnden Intervalle immer kürzer, bis eine gewünschte Genauigkeit erreicht ist.
In unserem Beispiel ist $h = 0{,}64$ und n = 400. Statt der Variablen h wird die Variable x verwendet.

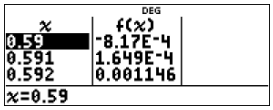
$p + c \cdot \sqrt{\frac{p\cdot(1-p)}{n}} - h = 0$ wird zu $f(x) = x + 2 \cdot \sqrt{\frac{x\cdot(1-x)}{400}} - 0{,}64$



[1] Die grafische Darstellung erfolgte hier aus formalen Gründen mit der TI-Nspire™ CX Technologie. Bei Verwendung des TI-30X Prio MathPrint™ sollte man ein geeignetes Koordinatensystem ausgedruckt zur Verfügung stellen.

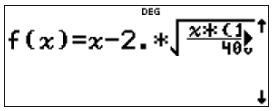
Mit [clear] lässt sich zurück ins TABLE SETUP wechseln.
Neue Schrittweite wird 0,01. Startwert wird 0,5.
Die Lösungen sollen auf Hundertstel gerundet bestimmt werden.

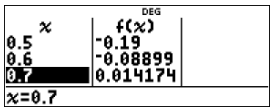


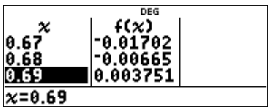


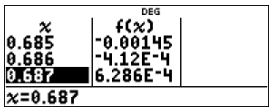
Das Verfahren wir mit immer kleinerer Schrittweite und neuen Startwerten weitergeführt.

Analog wird $p - 2 \cdot \sqrt{\frac{p \cdot (1-p)}{400}} - 0{,}64 = 0$ bearbeitet.



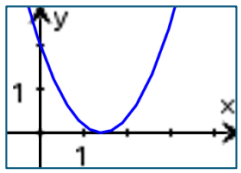






Die Grenzen des Konfidenzintervalls sind auf Hundertstel gerundet [0,59; 0,69], d. h. die Wahrscheinlichkeit mit 400 Würfen und 95,4%iger Sicherheit durch einen Wechsel die richtige Tür mit dem Auto zu finden, liegt etwa zwischen 59% und 69%.

Hinweis:
Das Verfahren der Intervallverfeinerung funktioniert nicht oder nur eingeschränkt, wenn z. B. die Nullstelle eine Berührstelle der x-Achse ist, oder wenn Nullstellen sehr dicht beieinander oder „sehr weit draußen“ liegen.



3. Simulation mit der TI-Nspire™ CX Technologie *(Hubert Langlotz)*

Da der Listeneditor des TI-30X Prio MathPrint™ nur Listen mit höchstens 50 Elementen verarbeiten kann, ist es uns ein Anliegen zu zeigen, wie mit einem komfortableren Rechner wie dem TI-Nspire™ CX II-T CAS die Simulationsergebnisse durch größere Stichprobenergebnisse stabilisiert werden können.
Eine direkte Übertragung des Vorgehens beim TI-30X Prio MathPrint™ könnte z. B. so aussehen:

	A liste1	B liste2	C
=	=seq(randint(1,3)−randint(1,3),k,1	=round(abs(liste1*0.5),0)	
1	0	0.	0.67
2	2	1.	
3	2	1.	
4	-1	1.	
5	0	0.	
6	0	0.	

Liste 1 erzeugt 2500 Differenzen von zwei Zufallszahlen:
seq(randint(1,3)-randint(1,3),k,1,2500)
Liste 2 wandelt diese Liste mit round(abs(liste1*0,5),0) um in eine Folge von Nullen und Einsen, wobei die Einsen für den Erfolg beim Wechseln simulieren.

© T3 Deutschland 2025

In der Zelle C1 wird mit dem Befehl $= \frac{sum(liste2)}{2500.}$ die relative Häufigkeit der Erfolge beim Wechseln berechnet.

Konfidenzintervall auf Hundertstel gerundet: [0,65; 0,69]:

$$\text{solve}\left(p-2\cdot\sqrt{\frac{p\cdot(1-p)}{2500}}=0.67,p\right)$$
$$p=0.688524$$
$$\text{solve}\left(p+2\cdot\sqrt{\frac{p\cdot(1-p)}{2500}}=0.67,p\right)$$
$$p=0.650933$$

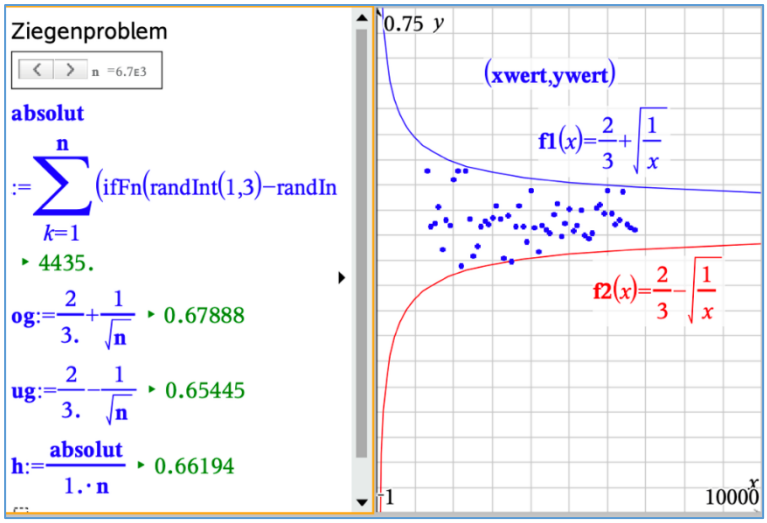
Aber es geht noch eleganter mit der TI-Nspire™ CX Technologie, insbesondere kann man die Versuchszahl noch weitaus steigern: In der Applikation Notes wird nur das Summenzeichen und der Befehl IfFn() genutzt (ifFn: „if Function“ wertet einen booleschen Ausdruck aus und gibt bei „wahr“ in diesem Fall den Wert „1“, sonst den Wert „0“ zurück.

Aufgrund der relativ hohen Versuchszahlen stabilisiert sich das Ergebnis wesentlich besser als bei Nutzung der Tabelle.

$$\sum_{k=1}^{1000}\left(\text{ifFn}\left(\text{randInt}(1,3)-\text{randInt}(1,3)=0,0,1\right)\right) \blacktriangleright 683$$
$$\sum_{k=1}^{10000}\left(\text{ifFn}\left(\text{randInt}(1,3)-\text{randInt}(1,3)=0,0,1\right)\right) \blacktriangleright 6694$$
$$\sum_{k=1}^{100000}\left(\text{ifFn}\left(\text{randInt}(1,3)-\text{randInt}(1,3)=0,0,1\right)\right) \blacktriangleright 66743$$

Mit diesem Ansatz kann man nun auch eine vereinfachte dynamische Anwendung z. B. zum Konfidenzintervall mit dem $\frac{1}{\sqrt{n}}$ –Gesetz erstellen.

© T3 Deutschland 2025



Im Notes-Fenster sind noch die Berechnungen der ermittelten relativen Häufigkeit und die obere und untere Grenze des Wurzeltrichters dargestellt. Im Graphikfenster kann man die schrittweise Vergrößerung durch Veränderung des Schiebereglers verfolgen.
Hinweis: Die Daten werden mittels capture-Befehl in den Listen xwert und ywert gesammelt und dann dargestellt.

4. Für Programmierfreunde *(Hubert Langlotz)*

Es liegt nahe, bei Verwendung eines MMS für das Ziegenproblem ein kleines Basic- oder Python-Programm zu schreiben. Die Programme stehen als Download unter „ziege_basic-Python.tns“ zur Verfügung.

In dieser Applikation ist ein kleines Programm in TI-Basic gegeben, dass die relative Häufigkeit für einen Gewinn für den Wechsler simuliert.

```
ziegen_basic
Define ziegen_basic()=
Prgm
Local i,n,a,t,g
Request "Anzahl Spiele:",n
g:=0
For i,1,n
  a:=randInt(1,3)
  t:=randInt(1,3)
  If a≠t Then
    g:=g+1
  EndIf
EndFor
Disp g,"mal Auto"
Disp "p=",approx(g/n)
EndPrgm
```

```
ziegen_basic()
Anzahl Spiele: 1000
660 mal Auto
p=0.66

ziegen_basic()
Anzahl Spiele: 10000
6626 mal Auto
p=0.6626
```

In Python geht dies ähnlich:

ziege1.py

```
from random import *
n = int(input("anzahl der Versuche: "))

g = 0
for i in range(n):
    a = randint(1, 3)
    t =  randint(1, 3)
    if a!=t:
        g +=1
print("Anzahl Gewinn: ",g)
```

Python-Shell

```
>>>#Running ziege1.py
>>>from ziege1 import *
anzahl der Versuche: 100000
Anzahl Gewinn:  66690
>>>
```

Auch die Veranschaulichung des Wurzeltrichters ist möglich:

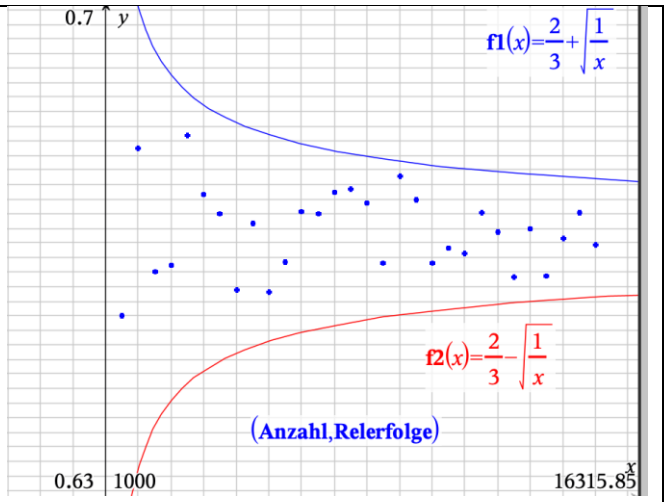
ziege2.py

```
from random import *
from ti_system import *
n = int(input("anzahl der Versuche: "))
z = int(input("Schrittweite, z.B. 500: "))

werte=list(range(z,n+1,z))
aliste =[0]*len(werte)
rliste =[0]*len(werte)

for i in range(0,len(werte)):
    for j in range(1,werte[i-1]):
        a = randint(1, 3)
        t =  randint(1, 3)
        if a!=t:
            aliste[i-1] +=1
for i in range(0,len(werte)):
    rliste[i]=aliste[i]/werte[i]

store_list("Anzahl", werte)
store_list("Erfolge", aliste)
store_list("RelErfolge", rliste)
print("Anzahl Versuche: " ,werte)
print("Anzahl Autos: ", (rliste))
```



Zum Download steht auch noch ein weiteres Python-Programm[1] unter „Ziegenproblem.tns“ zur Verfügung, welches neben der Simulation mit großen Versuchszahlen auch eine Einzelschrittsimulation ermöglicht und damit das Nachvollziehen der eigentlichen Simulation vereinfacht.

[1] Das Programm wurde von Veit Berger entwickelt.

© T3 Deutschland 2025

Autoren:
Dr. Hubert Langlotz
Dr. Wilfried Zappe

© T3 Deutschland 2025