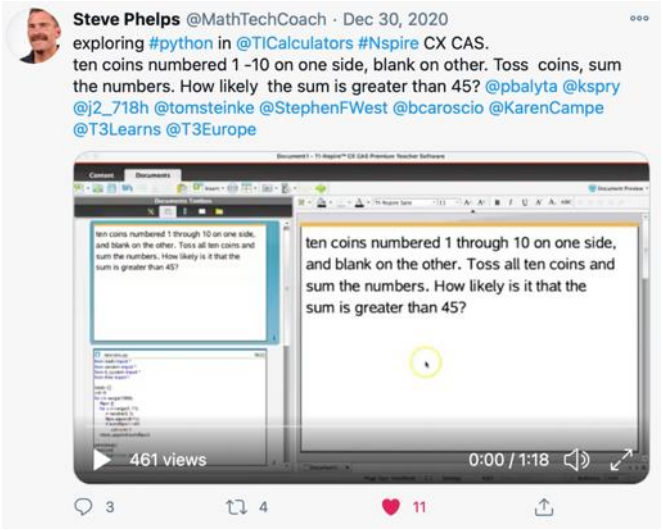
Hubert Langlotz
Sebastian Rauh

Das Problem der Zehn Münzen – Simulation mittels Programmierung mit Python/Basic

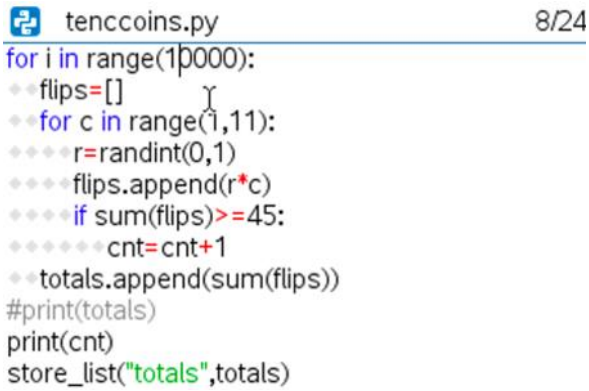
Kurz vor Weihnachten stand auf Twitter die folgende Aufgabe:

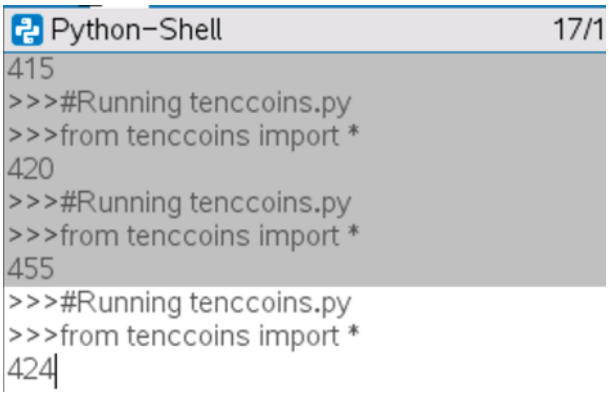
Ten Coins (idea S. Phelps via twitter):

Ten coins numbered 1 through 10 on one side and blank on the other. Toss all ten coins and sum the numbers. How likely is that the sum is grater than 45?[1]



Stephe Phelps simulierte das Problem mit Hilfe von Python auf dem TI Nspire.

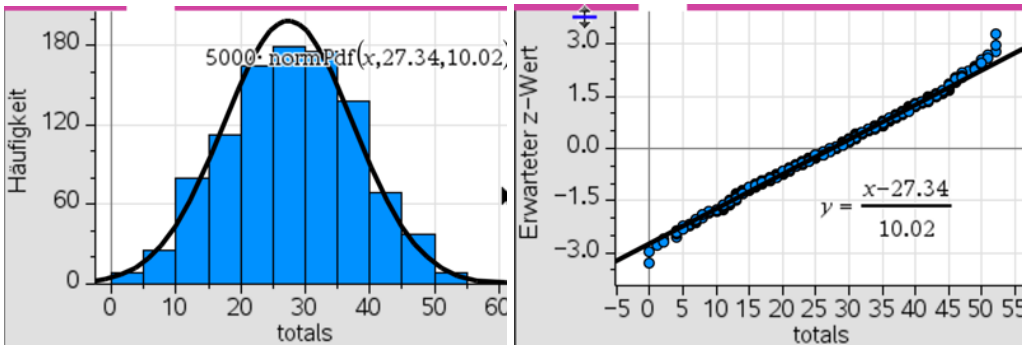




Die Ergebnisse deuten auf einen Wert um 4 Prozent hin.

Betrachtet man die Verteilung von 1000 Simulationen, so liegt es nahe, eine Normalverteilung zu vermuten, auch der Mittelwert von 27,5 ist einfach zu erklären.
Die Standardabweichung von rund 9,8 kann man sich mit Hilfe des Statistikmoduls ausgeben lassen (theoretische Erklärung siehe hinten).

[1] https://twitter.com/MathTechCoach/status/1344339088678264832?s=20



normCdf(44.5,∞,27.34,10.02) 0.0434

Hiermit erhält man für die gesuchte Wahrscheinlichkeit etwa 0,043.

Das zugehörige Konfidenzintervall (berechnet mit dem Eins-durch-Wurzel-n-Gesetz) wäre damit [0,.0118; 0,.075]).

$0.0434+\sqrt{\frac{1}{1000}}$ 0.075

$0.0434-\sqrt{\frac{1}{1000}}$ 0.0118

Möchte man eine Simulation mit z. B. 1 Million Versuchen durchführen, muss man auf das Speichern der einzelnen Werte verzichten (das Programm benötigt nur ca. 5 Sekunden):

```
>>>#Running tenccoins.py
>>>from tenccoins import *
41899
```

Unter Verwendung des 1/Wurzel(n) -Gesetzes liegt die gesuchte Wahrscheinlichkeit nun im Intervall [0,.0409; 0,.0429].

$0.041899+\sqrt{\frac{1}{1000000}}$ 0.0429

$0.041899-\sqrt{\frac{1}{1000000}}$ 0.0409

Theoretische Betrachtung
Zehn Münzen können auf $2^{10} = 1024$ Möglichkeiten angeordnet werden.
Nun muss man nur noch die für das Ereignis günstigen Möglichkeiten zählen:

Für die Summen größer gleich 45 ergeben sich insgesamt 43 Möglichkeiten. So kann man z.B. die Summe 52 nur erhalten, wenn die Münze mit der 3 oder die beiden Münzen mit der 1 und der 2 „verkehrt herum liegen“.

Summe	Auflistung (der fehlenden)	Anzahl
45	10;9+1;8+2;7+3;6+4;7+2+1;6+3+1;5+4+1;5+3+2;4+3+2+1	10
46	9; 8+1;7+2; 6+3; 5+4; 6+2+1; 5+3+1; 4+3+2	8
47	8; 7+1; 6+2; 5+3; 5+2+1; 4+3+1	6
48	7; 6+1; 5+2;4+3; 4+2+1	5
49	6; 5+1;4+2; 1+2+3	4
50	5; 4+1; 3+2	3
51	4; 3+1	2
52	1+2; 3	2
53	2	1
54	1	1
55		1
	Summe	43

Sofort erhält man P=43/1024 =4,2%.

Damit sind die Ergebnisse der Simulation bestätigt.

Erklärungen zur Berechnung des Erwartungswertes und der Standardabweichung für die Zufallsgröße X: Summe der Zahlen nach einmaligem Werfen.

X =x_i	0	k=$\{1; 2; \dots; 9; 10\}$
P(X =x_i)	$\frac{1}{2}$	$\frac{1}{2}$

Hiermit folgt E(X) = $\frac{k}{2}$ und mit $\sum_{k=1}^{10} k = 55$ folgt als Erwartungswert $E(X) = 27{,}5$.
Für die Varianz gilt damit V(X) = $\sum \mathrm{P}(\mathrm{X} = x_i) \cdot (x - E(X))^2 = \frac{k^2}{4}$ und mit $\sum_{k=1}^{10} k^2 = \frac{385}{4}$.
Somit ergibt sich eine Standardabweichung von $\sigma_x = \sqrt{V(X)} \approx 9{,}81$.

Eine Programmierung in Basic und Ermittlung der Häufigkeiten für die Summen von 0 ...55

Das Abzählen der möglichen Kombinationen der Münzen für eine Summe wird sehr schnell recht kompliziert - die Ermittlung der sog. Partialsummen einer Zahl - ist ein Problem, das viele reale Anwendungen hat. Die Programmierung erfolgt i.d.R. rekursiv.
Hier bietet sich allerdings der umgekehrte Weg an. Dazu werden alle möglichen Konfigurationen der Münzen systematisch durchgegangen und die Summe jeweils bestimmt.
Also anstelle alle Wege zu finden, die genau zur Summe 5 führen, werden alle 1024 möglichen Ausgänge erzeugt und diejenigen gezählt, die die 5 als Summe haben.

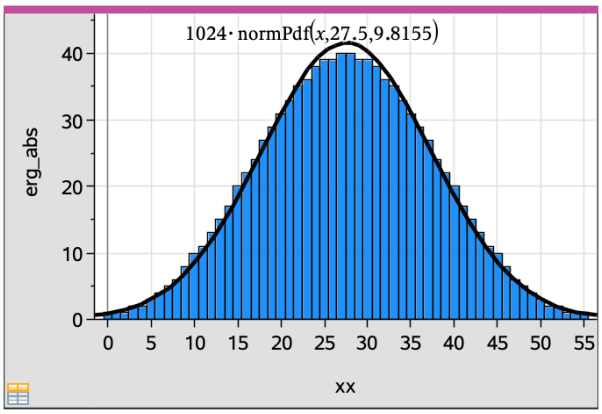
Die Münzen können jeweils zwei Werte annehmen, 0 oder n, wobei n die Position der Münze im Wurf ist. Um die $2^{10} = 1024$ verschiedenen Würfe effizient zu erstellen, werden die

Münzwürfe zunächst als Binärzahl interpretiert. Es ist praktisch, die Binärzahl von hinten nach vorne aufzuschreiben.
Der Fall (0,2,0,4,0,0,0,0,9,0) (also die Münze 2,4, und 9 zeigen ihre Werte, alle anderen zeigen 0) ist also 0101000010, wäre der Fall $100001010_2 = 266_{10}$

```
While n>0
  erg:=augment(erg,{remain(n,2)})
  n:=floor(n/2)
EndWhile
```

Die Abbildung zeigt die Umwandlung in eine Zahl zur Basis 2

Im Anschluss wird jede Ziffer der (umgedrehten) Binärzahl mit ihrem Stellenwert multipliziert und die Summe gebildet.
Eine Schleife bearbeitet so alle möglichen Ausgänge von 0 bis 1023. Es ergibt sich folgende Verteilung:



Nun kann man zur Kontrolle auch eine Überprüfung der bereits ermittelten Lagemaße durchführen.

yy:=allcoins()
{1,1,1,2,2,3,4,5,6,8,10,11,13,15,17,20,22,24,27,29,31,33,35,36,38,39,39,40,40,39,...

dotP(yy,seqn(n−1,56))	28160
$\frac{28160}{1024}$	$\frac{55}{2}$
$\sum_{i=1}^{56}\left(\frac{(xx[i]-27.5)^2\cdot yy[i]}{1024}\right)$	96.25
$\sqrt{96.25}$	9.81071

Lösungsvarianten

Nutzung von Matrizen

Man konstruiert eine Zufallsmatrix mit 10 Zeilen und z.B. 500 Spalten mittels

aa:=constructMat(randInt(0,1)*n,n,m,10,500).

$$aa:=\text{constructMat}(\text{randInt}(0,1)\cdot n,n,m,10,500)$$

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 2 & 2 & 0 & 2 & 2 & 2 & 0 & 2 & 0 & 2 & 0 & 2 \\ 3 & 3 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 & 0 & 3 \\ 4 & 4 & 4 & 4 & 0 & 0 & 4 & 0 & 4 & 4 & 0 & 0 \\ 0 & 5 & 5 & 5 & 5 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Man erkennt z. B. in der dritten Spalte, dass die ersten drei Münzen auf „0“ gefallen sind und die 4. und 5. Münze den Wert 4 bzw. 5 zeigt.

Mit dem Befehl

$$\text{countIf}(\text{sum}(aa,1,10),?\geq 45) \qquad 21$$

wird die Anzahl der Versuche (hier bei 500 Versuchen) gezählt, die eine Summe größer gleich 45 ergeben. Hieße in diesem Fall hätte man eine relative Häufigkeit von 0,042.

Natürlich lassen sich auch die anderen Fälle abzählen und als Ergebnisliste darstellen.

$$\text{frequency}(\text{mat}\blacktriangleright\text{list}(\text{sum}(aa,1,10)),\text{seqn}(n-1,56))$$

Auch eine Abzählvariante mit Matrizen ist möglich, hier soll lediglich der Befehl zur Erzeugung der Matrix genannt werden:

$$\text{constructMat}\left(\text{ifFn}\left(\text{mod}\left(\text{floor}\left(\frac{m-1}{2^{n-1}}\right),2\right)=1,n,0\right),n,m,10,2^{10}\right)$$

Python

Erheblich schneller als die Berechnung in TI-Basic laufen die Progamme zur Simuation und Abzählung in Python (ebenfalls auf dem Nspire). In der beigefügten .tns Datei finden sich diese Varianten.

Es sei an dieser Stelle auf den Artikel von Wilfried Zappe hingewiesen, der das Problem mit der Variante **Lists&Spreadsheet** löst.

Die hier beschriebenen Varianten nutzen bewusst verschiedene Zugänge, die ein digitales Mathematikwerkzeug bietet.
Vorteile der Programmierung (leicht variierbar für beliebige Münzanzahlen und beliebig hohe Simulationsversuche) stehen dem einfacheren Zugang mit Hilfe der Tabellenkalkulation gegenüber.

Autor:
Hubert Langlotz
Sebastian Rauh

Info:
Hubert Langlotz unterrichtet Mathematik *Fach* und *InformatikFach* am *Elisabeth*-Gymnasium in Eisenach
Sebastian Rauh unterrichtet Mathematik und Physik an der Gesamtschule Kamen