Goldmünzen – Wahrscheinlichkeiten mit CAS simulieren

In dem Artikel „Goldmünzen – Wahrscheinlichkeiten simulieren“[1)] hat W. Zappe zum Goldmünzenproblem eine Simulationsstrategie vorgestellt, die den WTR TI-30X Prio MathPrint™ nutzt. In diesem Beitrag wird das Goldmünzenproblem als Ausgangspunkt genutzt, um den Mehrwert des TI-Nspire™ CX-T CAS gegenüber einem einfachen WTR herauszuarbeiten. Im Mittelpunkt steht nicht nur das numerische Ergebnis, sondern vor allem der didaktische Weg: Die Simulation soll nachvollziehbar aufgebaut, mit zentralen Befehlen begründet und anschließend durch CAS-spezifische Möglichkeiten erweitert werden.

Im oben genannten Artikel wird die folgende Problemstellung mit einem WTR bearbeitet:

> *Der königliche Münzpräger soll je 100 Goldmünzen in 100 Kisten packen. Er ersetzt in jeder Kiste eine Goldmünze durch eine falsche Münze. Dabei meint er, der Betrug werde nicht auffallen, weil es sehr unwahrscheinlich sei, dass der König beim Inspizieren ausgerechnet eine falsche Münze entdeckt. Der Münzpräger weiß nämlich, dass der faule König aus jeder der 100 gelieferten Kisten genau eine Münze entnimmt und überprüft. Der Betrug gilt als aufgeflogen, wenn der König in mindestens einer der 100 Kisten eine falsche Münze findet. Ermittle das Risiko für den Münzpräger, dass der König den Betrug bemerkt.*

Zentraler Punkt des Artikels ist es, mit den „beschränkten“ Möglichkeiten eines WTR eine Simulation zu diesem Problem darzustellen. Dies gelingt sehr gut.

Es soll hier allerdings gezeigt werden, welche Möglichkeiten sich bieten, dieses Problem mit einem CAS zu bearbeiten und dabei auch konzeptionell Wege aufzeigen, wie ähnliche Probleme mit etwas mächtigeren Werkzeugen befriedigender gelöst werden können.

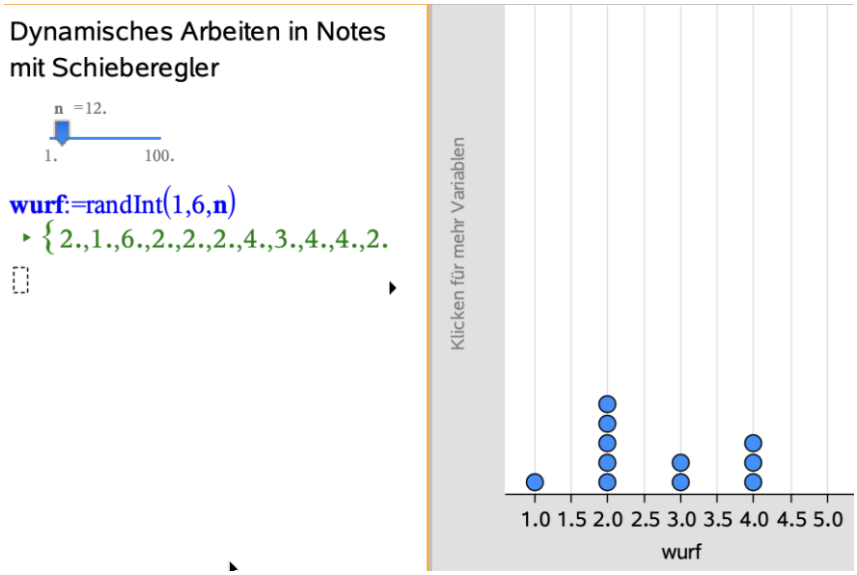
Im Einzelnen soll eingegangen werden auf:

- die Bedeutung von Notes
- Erläuterung einiger wesentliche Befehle für Simulationen
- Möglichkeiten von abschnittsweise definierten Funktionen.

Die Bedeutung der Applikation Notes für Simulationen

Notes bietet im Gegensatz zum Calculator die Möglichkeit des dynamischen Arbeitens mit mathematischen Befehlen und Definitionen. Dadurch kann z. B. mit Unterstützung eines Schiebereglers in Verbindung mit weiteren Applikationen schnell simuliert werden. Weitere Anwendungen sollen in einem extra Heft zu Notes vorgestellt werden, dieses wird demnächst erscheinen.

© T3 Deutschland 2026



Einige wesentliche Befehle für die folgenden Simulationen

1. seq()
   Wenn nicht bereits an anderer Stelle genutzt, so bietet sich hier die Möglichkeit, diesen Befehl, der eine Liste von Werten nach bestimmten Kriterien erzeugt, einzuführen.
2. randint() zum Erzeugen gleichverteilter ganzrationaler Zufallszahlen
3. countif() (Zähle Wenn) zum Suche nach bestimmten Werten in einer Liste
4. augment – Befehl zum Zusammenfügen zweier Listen
5. randsamp – Befehl zum Ziehen von Elementen aus einer Liste
6. Piecewise-Befehl zum Erzeugen einer abschnittsweise definierten Funktion

Diese Liste ist nicht vollständig, um auch andere Simulationen durchzuführen, genügt aber hier für den Einstieg.

1. Lösung des Problems

Mit dem augment-Befehl wird zunächst eine Kiste modelliert, die aus 99 Goldmünzen und einer falschen Münze besteht. Mit Countif wird in 100 Kisten mit dem randsamp-Befehl nach der fehlerhaften Münze gesucht. Hier findet man bei drei von 100 Versuchen die fehlerhafte. Mit Hilfe des Seq-Befehls wird dieses Experiment nun 200-mal durchgeführt. Schließlich nutzt man wieder den Countif-Befehl, um die relative Häufigkeit zu bestimmen	Goldmünzen Lösung 1 kiste:=augment({f},seq(g,k,1,99)) {f,g,g,g

2. Lösung des Problems

Mit dem countif-Befehl wird ermittelt, ob und wie oft bei 100 Ziehungen aus den Zahlen 1 bis 100 die Zahl 1 gezogen wird, hier traf dies einmal zu. Mit Hilfe des Folgebefehls wird dieser Versuch 100-mal durchgeführt und die „Treffer“ ermittelt, dies wird in der Liste gespeichert. Mittels countif wird überprüft, wie oft es zu „Treffern kam. Um aussagefähige relative Häufigkeiten zu bekommen, bindet man diesen letzten Befehl nun in eine Summe ein, bei der dass Experiment 500-mal durchgeführt wird. Das Ergebnis liegt erwartungsgemäß nah bei der theoretischen Wahrscheinlichkeit.	countIf(randInt(1,100,100),?=1) ▸ 1 seq(countIf(randInt(1,100,100),?=1),n,1,100) ▸ {2,0,0,0,1,3,0,1,3,2,1,1,1,2,1,0,0,0,2,1,0,0,1,2,1,1,2,1,0,0,1,1,0,1,1,0,' countIf(seq(countIf(randInt(1,100,100),?=1),n,1,100),?≥1) ▸ 56 $\frac{1}{500.}\cdot\sum_{k=1}^{500}(\text{countIf}(\text{seq}(\text{countIf}(\text{randInt}(1,100,100),?=1),n,1,100),?\geq 1))$ ▸ 63.338 Es gilt $P(\text{entdeckt})=1-\left(\frac{n-1}{n}\right)^k \quad 1-\left(\frac{99}{100}\right)^{100}$ ▸ 0.63397

3. . Lösung des Problems – Nutzung der piecewise-Funktion

Mit Hilfe der piecewise-Funktion wird eine Funktion w definiert, die den Wert 1 liefert, falls x größer oder gleich 1 ist; andernfalls wird 0 zurückgegeben. Nun fügt man den bereits bekannten Befehl als Argument in die Funktion ein und nutzt erneut die Summenbildung. Kleine Änderungen lassen sich in diesem Ansatz leicht umsetzen: Wird der Münzprägер beispielsweise gieriger und ersetzt zwei Münzen pro Kiste, muss nur die entsprechende Zahl angepasst werden.	Goldmünzen Lösung 3 $w(x):=\begin{cases}1, & x\geq 1\\ 0, & \end{cases}$ ▸ *Fertig* w(countIf(randInt(1,100,100),?=1)) ▸ 1 $\sum_{k=1}^{10000}(w(\text{countIf}(\text{randInt}(1,100,100),?=1)))$ ▸ 6352 $\sum_{k=1}^{10000}(w(\text{countIf}(\text{randInt}(1,100,100),?\leq 2)))$ ▸ 8655 $1-\left(\frac{98}{100}\right)^{100}$ ▸ 0.86738

Literaturhinweis

1) Zappe, W.: Goldmünzen – Wahrscheinlichkeiten simulieren. In: https://ti-unterrichtsmaterialien.net/materialien?resource_id=4019&cHash=ec53721f3b0f9e93aa1211e1f74a6173

Autor:
Dr. Hubert Langlotz

© T3 Deutschland 2026