Das Rosinenbrötchenproblem

25 Rosinen werden in einen Kuchenteig gegeben und gut mit diesem vermischt. Wir gehen davon aus, dass dabei die Rosinen zufällig in die etwa 25 gleich großen Teigstücke verteilt werden, die dann zum Backen der Brötchen verwendet werden. Untersuche, wie viele Brötchen keine, genau eine, zwei, drei oder mehr als drei Rosinen enthalten.

(Quelle**: privat)**

Dieses stochastische Problem wird im Folgenden mit zwei verschieden Technologien gelöst. Im ersten Teil zeigt Wilfried Zappe anhand eines wissenschaftlichen Taschenrechners (WTR), hier der TI-30X Prio MathPrint™, verschiedene Lösungsmöglichkeiten auf. Im zweiten Teil nutzen Hubert Langlotz und Sebastian Rauh ein MMS, hier den TI-Nspire™ CX II-T CAS, und zeigen, wie effektiv Simulationen mit einem solchen Gerät umgesetzt werden.

Teil 1: Lösungsvariationen mit dem TI-30X Prio MathPrint™

Wilfried Zappe

Simulation mit Zufallszahlen

Grundgedanken zur Vorgehensweise

Anstelle der 25 Brötchen wird ein Raster aus 25 Feldern angelegt. Jedes Feld bekommt eine zweistellige Zahl als Nummer. Das gelb gefärbte Feld hat z. B. die Nummer 34, das blaue Feld die Nummer 52.

5					
4					
3					
2					
1					
	1	**2**	**3**	**4**	**5**

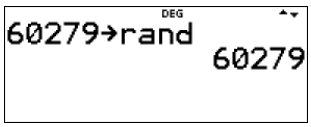
Mit dem Zufallsgenerator des TI-30X Prio MathPrint™ werden 25 Paare von ganzzahligen Zufallszahlen erzeugt, von denen jede im Intervall [1; 5] liegt. Jedes dieser Paare stellt eine Feldnummer dar.

Zunächst werden einige Möglichkeiten erläutert, die der TI-30X Prio MathPrint™ bietet, um solche Paare von ganzzahligen Zufallszahlen zu erzeugen.

Unter [2nd] [!nCr] findet man die beiden Anwendungen zum Erzeugen von Zufallszahlen:

rand erzeugt eine rationale (Pseudo-) Zufallszahl zwischen 0 und 1

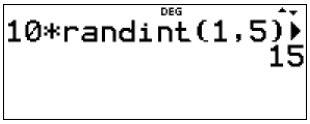
randint(a, b) erzeugt eine ganzzahlige (Pseudo-) Zufallszahl aus dem Intervall [a; b]

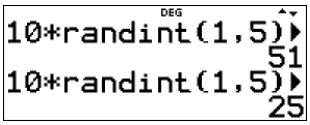


Vor der Anwendungen der beiden Optionen sollte zunächst der Zufallsgenerator durch Eingabe einer individuell gewählten positiven ganzen Zahl und Drücken von [sto→] sowie *rand* ([2nd][!nCr][1]) und [enter] initialisiert werden. (Hinweis: Jeder in der Lerngruppe sollte möglichst eine andere Zahl eingeben, bewährt hat sich das Geburtsdatum in Form von TagMonatJahr, z.B. 1122023. Damit bekommt jeder andere (Pseudo-)Zufallszahlen.)

Für die Erzeugung von passenden Zufallszahlen für die oben beschriebene Simulation braucht man zweistellige Zufallszahlen, bei denen sowohl die erste Ziffer als auch die zweite Ziffer eine ganzzahlige Zufallszahl aus dem Intervall [1; 5] ist.

1. Eine zweistellige natürliche Zahl ist von der Form $\overline{ab} = 10a + b$.
   Im einfachsten Fall kann man also für die obige Simulation im Hauptbildschirm eingeben: $10 \cdot randint(1,5) + randint(1,5)$
   Jedes weitere Drücken von [enter] erzeugt eine neue Zufallszahl mit diesen Eigenschaften.
   Nachteil dieser Variante: Man sieht nicht auf dem Bildschirm, wie viele Zufallszahlenpaare schon erzeugt worden sind.

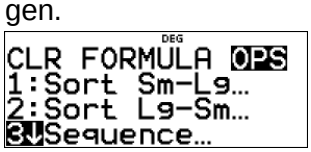


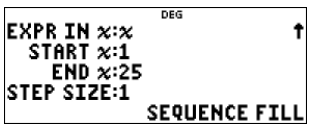


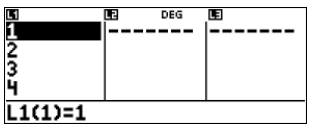
2. Wird der Term $10 \cdot randint(1,5) + randint(1,5)$ als Operation mit Hilfe von [set op] ([2nd][×]) gespeichert und jedes Mal mit [op] ([2nd][)]) aufgerufen, wird auch die Folge der erzeugten Zufallszahlenpaare durchnummeriert.

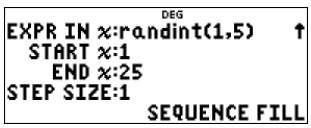
   Nachteil: Relativ viele Tastendrücke.

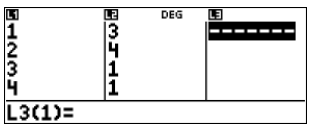
3. Verwendet man [data], so lassen sich eine Liste mit den natürlichen Zahlen von 1 bis n (mit $n \leq 50$) sowie zwei weiteren Listen mit jeweils einer Zufallszahl randint(1,5) erzeugen.

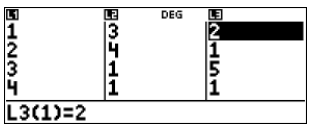






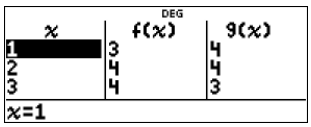


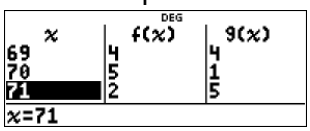




   Nachteil: Die Listen umfassen höchstens 50 Elemente.

4. Mit [table] den Funktionseditor öffnen und ***1: Add/Edit Func*** wählen.
   Unter f(x) mit [2nd][!nCr][2] eingeben: randint(1,5) [enter].
   Unter g(x) mit [2nd][!nCr][2] eingeben: randint(1,5) [enter].
   Die Funktionstabelle zeigt die Zufallspaare an. Sie ist nicht auf 50 Paare beschränkt.

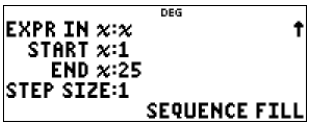




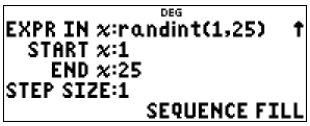
Konkrete Lösung

Zuerst wird etwas ausführlicher die Variante mit [data] gezeigt.

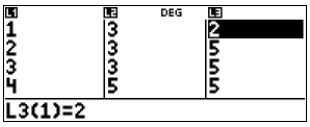
1. Zunächst wird der Zufallsgenerator durch Eingabe einer individuell gewählten positiven ganzen Zahl und Drücken von [sto→] sowie der Eingabe von *rand* ([2nd] [!nCr] [1]) und [enter] initialisiert.
2. [data] [data] öffnen und *OPS 3: Sequence* wählen.
   Vereinbarung für Liste L1 wie nebenstehend mit [enter] bestätigen.
   [data] [data] [▶] [▶] [3] [enter] [x^{yzt}_{abcd}] [enter] [1] [enter] [2] [5] [enter] [enter] [enter]



3. Wieder [data] öffnen, *OPS 3: Sequence* wählen und die Vereinbarungen für die Option [randint(1, 5)] wie nebenstehend für Liste L2 treffen. Analog die Einstellungen auch für die Liste L3 wie bei L2 vereinbaren, mit [enter] bestätigen.
   [data] [▶] [▶] [3] [▶] [enter] [2nd] [!nCr] [2] [1] [2nd] [·] [5] [)] [enter] [enter] [enter] [enter] [enter]



4. Die angezeigten Paare von Zufallszahlen werden als Nummern für die 25 Felder interpretiert. Das entsprechende Feld wird in der Tabelle angekreuzt.



5			xxx	xx	xx	Kreuze pro Feld	Anzahl Felder
4	xxx	xx	x	x		0	9
3				x	xx	1	9
2	x		xx	x		2	5
1	x	x		x	x	3	2
	1	**2**	**3**	**4**	**5**	mehr als 3	0

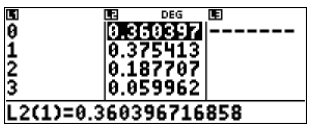
Wir zählen, wie viele Felder mit 0, 1, 2, 3 oder mehr als 3 Kreuzchen es gibt.

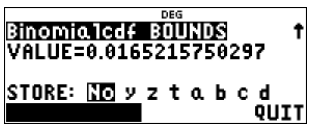
Theoretische Lösung:

Die Wahrscheinlichkeit, dass ein bestimmtes Feld genau k Kreuzchen enthält, ist

$P_{25}(k) = \binom{25}{k} \cdot \left(\frac{1}{25}\right)^k \cdot \left(\frac{24}{25}\right)^{25-k}$, also binomialverteilt mit n = 25 und p = 1/25 = 0,04.

Für k = 0, 1, 2, 3 mit Binompdf; für k > 3 mit Binomcdf Bounds 4 bis 25:

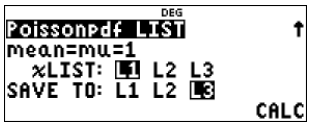


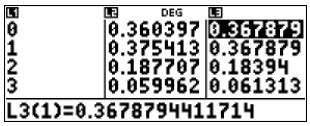


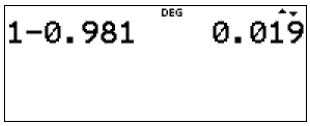
Die Wahrscheinlichkeit kann näherungsweise auch mit der Poisson-Verteilung berechnet werden.

$P(X = k) = \frac{\mu^k}{k!} \cdot e^{-\mu}$ mit $\mu = n \cdot p = 25 \cdot \frac{1}{25} = 1$

Für k = 0, 1, 2, 3 mit Poissonpdf; für k > 3 mit 1 – Poissoncdf(3)







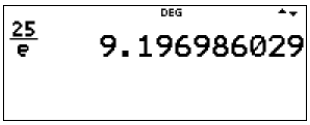
Gegenüberstellung der Ergebnisse:

	Simulation/ relative Häufigkeit	Wahrscheinlichkeit	
Wert für k		binomial	Poisson
0	0,36	0,360	0,369
1	0,36	0,375	0,368
2	0,2	0,188	0,182
3	0,08	0,060	0,061
mehr als 3	0	0,017	0,019

Verallgemeinerung:

Wegen $\left(\frac{24}{25}\right)^{25} = \left(1 - \frac{1}{25}\right)^{25} \approx e^{-1} = \frac{1}{e}$ bleiben im Mittel ungefähr $\frac{25}{e} \approx 9$ Felder leer.

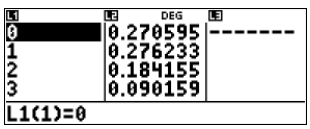
Hinweis auf das 1/e-Gesetz:
Verteilt man n Objekte zufällig auf n Felder, so bleiben ungefähr $\frac{n}{e}$ Felder leer.

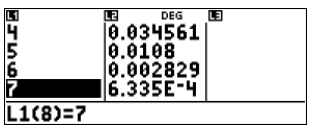


Vertiefung 1
Werden 25 Rosinen auf 25 Brötchen zufällig verteilt, dann bleiben nach den oben getroffenen Betrachtungen mehr als ein Drittel der Brötchen ohne eine einzige Rosine. Das ist für die Kunden des Bäckers wenig befriedigend. Wir untersuchen deshalb, wie sich die zufällige Verteilung von mehr als 25 Rosinen auf 25 Brötchen auswirkt.

Wir beginnen diesmal mit der theoretischen Berechnung für die zufällige Verteilung von n Rosinen auf 25 Felder und schließen eine Simulation an.

Rechnung für n = 50 mit Binomialverteilung: $P_{50}(k) = \binom{50}{k} \cdot \left(\frac{1}{25}\right)^k \cdot \left(\frac{24}{25}\right)^{50-k}$

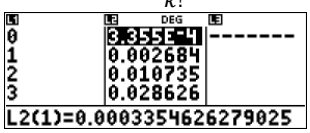


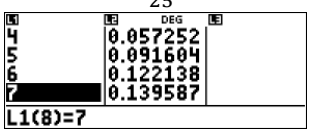


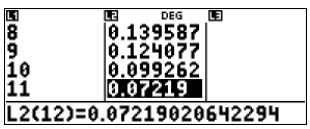
Die Wahrscheinlichkeit für keine Rosinen in einem zufällig ausgesuchten Brötchen ist um etwa 10 % gesunken, aber immer noch größer als ein Viertel.

Rechnung für n = 200 mit Poisson-Verteilung: Wahrscheinlichkeiten für 200 Rosinen auf 25 Brötchen.

$P(X = k) = \frac{\mu^k}{k!} \cdot e^{-\mu}$ mit $\mu = n \cdot p = 200 \cdot \frac{1}{25} = 8$



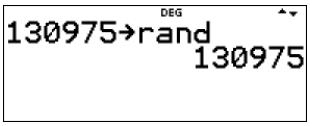




Die Wahrscheinlichkeit für keine Rosine in einem zufällig ausgesuchten Brötchen ist nun fast null. Die größten Wahrscheinlichkeiten liegen bei 7 bis 8 Rosinen pro Brötchen (siehe Erwartungswert).

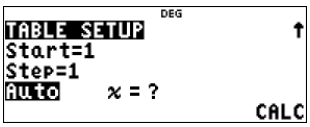
Simulation für n = 100 für n = 100 und $p = \frac{1}{25}$:

Die Erzeugung von ganzzahligen Zufallszahlen wird wieder mit randint(1,5) realisiert. Anders als auf Seite 1 wird die die Wiedergabe der Paare von Zufallszahlen durch [table] ermöglicht. Damit entfällt die Beschränkung auf 50 Elemente pro Liste, die es bei der Option [data] gibt.

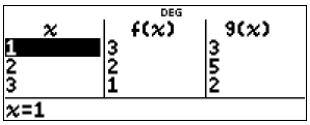


1. Zunächst wird der Zufallsgenerator durch Eingabe einer individuell gewählten positiven ganzen Zahl und Drücken von [sto→] mit *rand* ([2nd] [!nCr] [1]) und [enter] initialisiert.

2. [table] öffnen und *1: Add/Edit Func* wählen.
   Unter f(x) mit [2nd] [!nCr] [2] eingeben: randint(1,5) [enter].
   Unter g(x) mit [2nd] [!nCr] [2] eingeben: randint(1,5) [enter].



3. Nach dem [enter] gelangt man zum TABLE SETUP.
   Alle Einträge wie nebenstehend mit [enter] bestätigen.



4. Die Funktionstabelle zeigt als Funktionswerte unter f(x) und unter g(x) je eine ganzzahlige Zufallszahl aus dem Intervall [1; 5]. Diese Paare werden als Nummern für die 25 Felder interpretiert. Das entsprechende Feld wird in der Tabelle angekreuzt.

5	xxx	xxxx	xxx	x	xxx
4	xx	xx	xxxxx	xxxxxxx	xxxxx
3	xxx	xxxxxxxx	xxxxx	xx	xxxxx
2	xxxxxx	xxx	xxxx	x	xxxxx
1	xxxxxx	xxxxx	xxxx	xxx	xxxxx
	1	**2**	**3**	**4**	**5**

Wir zählen die Kreuzchen in jedem Fach:

5	3	4	3	1	3
4	2	2	5	7	5
3	3	8	5	2	5
2	6	3	4	1	5
1	6	5	4	3	5
	1	**2**	**3**	**4**	**5**

Wir zählen, wie viele Felder mit 0, 1, 2, 3, 4, 5, 6, 7 oder mehr als 7 Kreuzchen es gibt.

Kreuze pro Feld	Anzahl Felder	Relative Häufigkeit	Theoretische Wkt.
0	0	0	0,0169
1	2	0,08	0,0703
2	3	0,12	0,1450
3	6	0,24	0,1973
4	3	0,12	0,1994
5	7	0,28	0,1595
6	2	0,08	0,1052
7	1	0,04	0,0589
mehr als 7	1	0,04	0,0285

Theoretische Wahrscheinlichkeit mit Binomialverteilung (siehe Tabelle, letzte Spalte):

$P_{100}(k) = \binom{100}{k} \cdot \left(\frac{1}{25}\right)^k \cdot \left(\frac{24}{25}\right)^{100-k}$

Vertiefung 2[1)]

Wie viele Rosinen muss man in 1250 g Teig tun, damit ein 50 g-Brötchen mit 99%iger Sicherheit

a) mindestens eine Rosine enthält,
b) mindestens fünf Rosinen enthält?

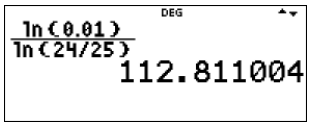
Aus 1250 g Teig kann man 25 Brötchen zu je 50 g backen. Wenn der Teig gut gemischt wird, ist die Annahme gerechtfertigt, dass jede Rosine mit der Wahrscheinlichkeit $p = \frac{1}{25} = 0{,}04$ in jedes der 25 Brötchen kommt. Das ist gleichbedeutend damit, dass jede der n Rosinen mit der Wahrscheinlichkeit $\frac{24}{25}$ nicht in dieses Brötchen kommen.

Lösung zu a)

Gesucht ist $P(X \geq 1) > 0{,}99$. Es gilt $P(X \geq 1) = 1 - P(X = 0)$.

$1 - P(X = 0) = 1 - \binom{n}{0} \cdot \left(\frac{1}{25}\right)^0 \cdot \left(\frac{24}{25}\right)^{n-0} > 0{,}99$

$\left(\frac{24}{25}\right)^n < 0{,}01 \Rightarrow n > \frac{\ln(0{,}01}{\ln(\frac{24}{25})}$



Man muss etwa 113 Rosinen in den Teig tun und hat dann im Mittel 4 bis 5 Rosinen pro Brötchen.

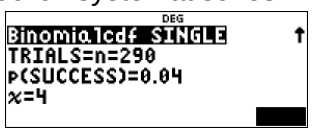
Lösung zu b)

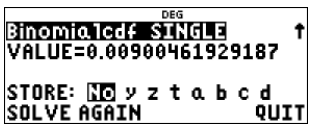
Gesucht ist $P(X \geq 5) > 0{,}99$. Es gilt $P(X \geq 5) = 1 - P(X \leq 4)$.

$1 - P(X \leq 4) = 1 - \sum_{k=0}^{4} \binom{n}{k} \cdot \left(\frac{1}{25}\right)^k \cdot \left(\frac{24}{25}\right)^{n-k} > 0{,}99$

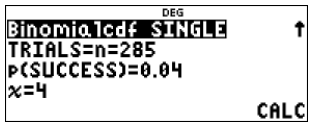
$\sum_{k=0}^{4} \binom{n}{k} \cdot \left(\frac{1}{25}\right)^k \cdot \left(\frac{24}{25}\right)^{n-k} < 0{,}01 \Rightarrow binomcdf(n, 0{,}04, 0, 4) < 0{,}01$

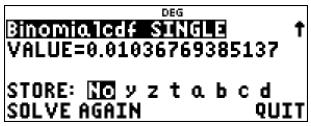
Lösung durch systematisches Probieren:

n = 290:


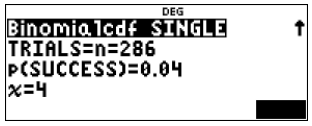


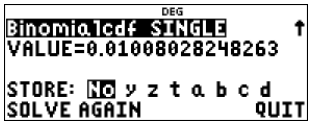
p ist etwas kleiner als 0,01

n = 285:


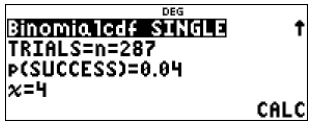


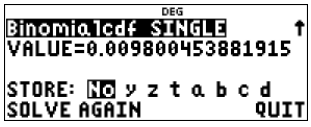
p ist etwas zu groß

n = 286:




p ist immer noch etwas zu groß

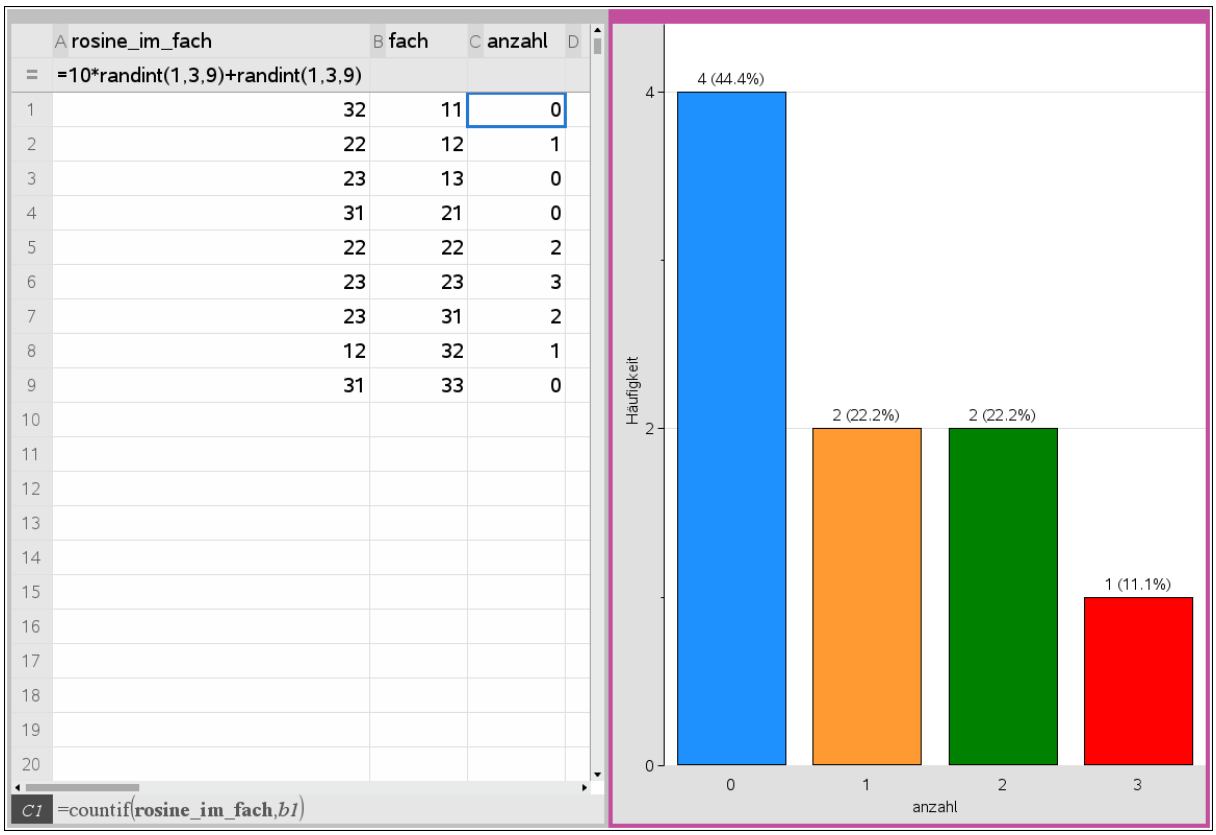
n = 287:




der erste Wert mit p < 0,01

Man muss etwa 287 Rosinen in den Teig tun und hat dann im Mittel 11 bis 12 Rosinen in jedem Brötchen.

Hinweis:
Das hier bei Verwendung des WTR notwendige halbhändische Verfahren des Auszählens der Anzahl der Brötchen mit k Rosinen ($0 \leq k \leq n$) kann mithilfe einer Tabellenkalkulation (z. B. die des CAS-Rechners TI-Nspire, aber auch mit Excel) automatisiert und auf größere Anzahlen von Rosinen bzw. Brötchen erweitert werden. Hier wird eine einfache Anwendung gezeigt, die wegen der besseren Übersicht und Nachvollziehbarkeit mit neun Rosinen, die auf neun Brötchen (Fächer) verteilt werden, auskommen soll.



Die neun Brötchen kann man sich als neun Fächer in einem 3x3-Raster vorstellen. Jedes Fach bekommt eine zweistellige Nummer von 11, 12, 13, 21, ... bis 33 (Spalte B).
Die Rosinen werden per Zufallszahl auf diese Fächer verteilt (Spalte A).
In Spalte C wird z. B. mit dem Befehl = countif(rosine_im_fach,b1) gezählt, wie oft in Spalte A die Nummer 11 auftaucht. Kopiert man diesen Befehl als relativen Zellbezug nach unten bis in die Zelle C9, so wird diese Abfrage nach der Anzahl der möglichen Zufallszahlen für alle Fächer automatisch durchgeführt.
Die Ergebnisse in der Spalte C werden als Schnellgraph veranschaulicht.
Setzt man den Cursor auf den Kopf von Spalte A, dann lässt sich mit [ctrl] [R] sofort eine neue Simualtion erzeugen.
Im Prinzip wird damit auch klar, wie man das Verfahren auf größere Anzahlen von Rosinen und Brötchen erweitern kann.

Ausführliche Hinweise zur Verwendung des TI-Nspire™ CX II-T CAS in diesem Sachzusammenhang werden im nachfolgenden Teil 2 gegeben.

Teil 2: Lösungsmöglichkeiten mit einem MMS

Sebastian Rauh, Hubert Langlotz

Es werden drei verschieden Applikationen genutzt, um die Lösung zu automatisieren und einfache grafische Darstellungen zu erzeugen:

a) die Tabellenkalkulation Lists&Spreadsheet,
b) Programmierung mit Python und
c) die Applikation Notes in Verbindung mit anderen Applikationen.

Prinzipielle Umsetzung in der Tabellenkalkulation

Aufgabe: Es sollen 20 Rosinen auf 10 Brötchen verteilt werden. Untersuchen Sie durch Simulation, wie viele Brötchen keine, genau eine, zwei, drei oder mehr Rosinen enthalten.

In Spalte A mit dem Titel „rosine“ wird mit der Anweisung *=randint(1,10,20)* simuliert, in welchem der Brötchen Nr. 1 bis Nr. 10 die 20 Rosinen zu liegen kommen. Im Screenshot ist z. B. zu erkennen, dass die Rosinen 1, 10, 15, 19 und 20 im Brötchen Nr. 10 landen.

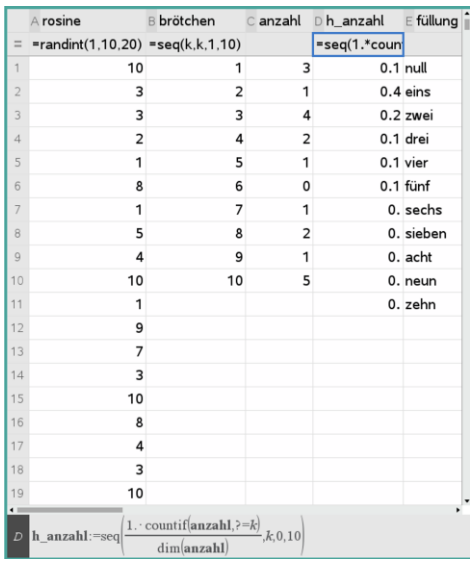
Spalte B mit dem Titel „brötchen“ wird mithilfe der Anweisung *=seq(k, k, 1, 10)* mit den natürlichen Zahlen von 1 bis 10 gefüllt, die für die zehn Brötchen stehen.

Spalte C erhält den Titel „anzahl“. In der Zelle C1 wird mithilfe der Anweisung *=countif(rosine,b1)* ermittelt, in wie vielen Fällen eine Rosine im Brötchen Nr. 1 landet. Diese Anweisung wird als relativer Zellbezug bis in die Zelle C10 kopiert.

In der Spalte D mit dem Titel „h_anzahl“ wird mit der Anweisung

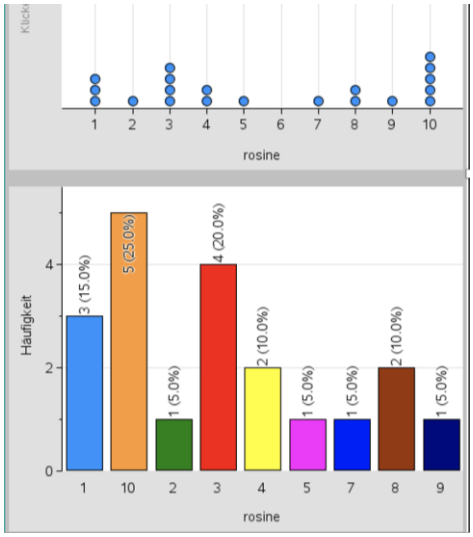
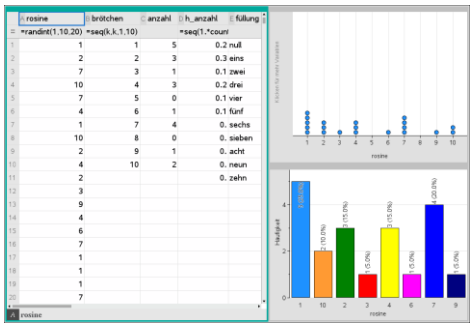
$= seq\left(\frac{1 \cdot countif(anzahl,?=k)}{\dim(anzahl)}, k, 0{,}10\right)$ der Anteil (die relative Häufigkeit) der Brötchen mit 0, 1, 2, …, 10 Rosinen bestimmt.

Spalte E ermöglicht eine Zuordnung der relativen Häufigkeiten zu den Brötchen mit null, eins, zwei, …, zehn Rosinen als Füllung.



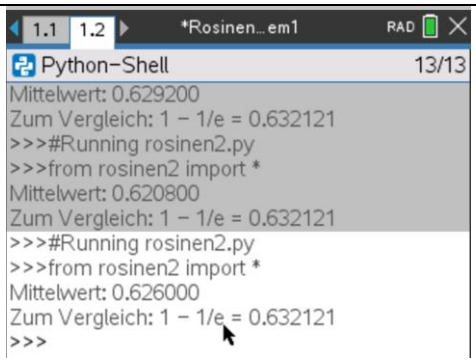
	A rosine	B brötchen	C anzahl	D h_anzahl	E füllung
=	=randint(1,10,20)	=seq(k,k,1,10)		=seq(1.*coun	
1	10	1	3	0.1	null
2	3	2	1	0.4	eins
3	3	3	4	0.2	zwei
4	2	4	2	0.1	drei
5	1	5	1	0.1	vier
6	8	6	0	0.1	fünf
7	1	7	1	0.	sechs
8	5	8	2	0.	sieben
9	4	9	1	0.	acht
10	10	10	5	0.	neun
11	1			0.	zehn
12	9				
13	7				
14	3				
15	10				
16	8				
17	4				
18	3				
19	10				

D h_anzahl:=seq$\left(\frac{1 \cdot \text{countif}(\text{anzahl},?=k)}{\dim(\text{anzahl})}, k, 0, 10\right)$

Die grafischen Darstellungen sind entstanden als Schnellgraphen der Werte aus Spalte A, oben als Punktdiagramm und unten als Säulendiagramm. Beim Säulendiagramm kommt auf der waagerechten Achse der Wert 6 nicht vor, weil bei der hier ausgewerteten Simulation das Brötchen Nr. 6 keine Rosine abbekommen hat.	
Befindet man sich im *Lists&Spreadsheetfenster*, kann man mit <ctrl> <R> die Simulation wiederholen.	

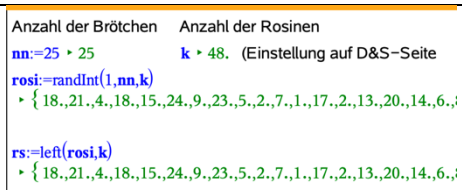
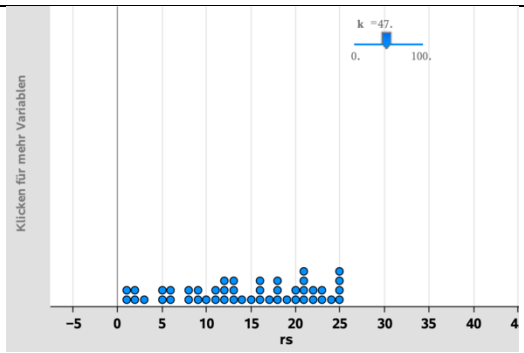
Programmierung am Beispiel von „Mindestens eine Rosine“ mit Python

Es werden n Rosinen auf n Brötchen verteilt. Im Array r „wird für n Rosinen ein Platz in n Brötchen gesucht“. Sobald in den geschachtelten for-Schleifen eine Rosine in einem Brötchen gefunden wird, wird die Schleife verlassen und der Zähler z um „1“ erhöht und die relative Häufigkeit z/n übergeben.	``` from random import * from math import exp def rosinen(n): r = [randint(0, n - 1) for i in range(n)] z = 0 for j in range(n): k = 0 for i in range(n): if r[i] == j: k = 1; break z += k return z / n ```
In der eigentlichen Simulation kann man die Anzahl der Durchläufe m wählen. Es wird dann der Mittelwert aller Durchläufe berechnet und im Vergleich zum theoretischen Wert ausgegeben.	``` def experiment(m): z = 0 for i in range(m): z += rosinen(100) mw = z / m #Mittelwert print("Mittelwert: %1.6f" % mw) ew= 1 - 1 / exp(1) #Erwartungswert print("Zum Vergleich: 1 - 1/e = %1.6f" % ew) experiment(35) ```

Die Simulation liefert gute Näherungswerte für die Wahrscheinlichkeit, dass ein Brötchen mindestens eine Rosine enthält, da der theoretische Wert bei $1 - \frac{1}{e} \approx 63\ \%$ liegt.	

Nutzung von *Notes* in Verbindung mit weiteren Applikationen

Notes in Verbindung mit *Data&Statistics*

Im Notes-Fenster können einfache Änderungen der Anzahl der Brötchen vorgenommen werden, die Anzahl der Rosinen wird im *D&S*-Fenster eingestellt. Die Variable rosi speichert die zufällige Verteilung von k Rosinen auf die nn Brötchen. Die Liste rs dient dazu, nur die ersten k Rosinen darzustellen.	
In der grafischen Darstellung ist die Verteilung der Rosinen auf die Brötchen gut zu erkennen. Durch das Auftragen der Häufigkeit von rs kann die Verteilung nach und nach aufgebaut werden.	

***Notes* mit dem *frequency()* Befehl.**

Der *frequency()* Befehl benötigt eine Urliste und eine Liste, mit den möglichen Anzahlen um eine Häufigkeitsliste zu erzeugen.

Als erstes werden die Eckdaten der Simulation definiert, hier ist die Anzahl der Brötchen auf 25 und die Anzahl der Rosinen auf 100 festgelegt. Dann wird die Simulation selbst durchgeführt, es werden 100 Zufallszahlen erzeugt, die zwischen 1 und 25 liegen und als ls abgespeichert. Im Folgenden muss bestimmt werden, wie oft jede der Zahlen in der Liste ls auftritt. Jedes Auftreten einer Zahl ist eine Rosine in dem Brötchen.	**Per "Hand"** – **Mindestens eine Rosine in jedem Brötchen** Anzahl der Brötchen **br**:=25 ▸ 25 Anzahl der Rosinien **ro**:=100 ▸ 100 ©Erzeugung der Verteilung der Rosinen **ls**:=randInt(1,**br**,**ro**) ▸ { 21,21,14,25,22,24,8,18,13,25,17,11,21,20,13,11,10,3,5,9,9,25,25,19,2,
Jetzt wird mit dem *frequency()*-Befehl die Urliste in eine Häufigkeitsliste zusammengefasst. Der Befehl benötigt die auszuzählende Liste **ls** und eine Liste von sogenannten „Bins“ (***seq(n,n,1,br)***={1,2,3,..,br)**,** also eine Liste mit möglichen Ergebnissen. Die Ausgabe ist eine Liste, die die Häufigkeiten zählt. Das letzte Element ist die Anzahl der Elemente, die in kein Bin passen, also in unserem Fall immer 0.	**anz_br**:=frequency(**ls**,seq(*n*,*n*,1,**br**)) ▸ { 1,4,5,4,4,2,4,5,4,4,4,1,5,5,2,4,5,8,3,6,5,2,0,6,7,0 }
Nun soll die Anzahl der Brötchen mit 0, 1, 2, ..., 9 Rosinen bestimmt werden. Dazu wird erneut der *frequency()* Befehl benutzt, auf **anz_br** ohne das letzte Element (die Anzahl der Elemente, die in kein Bin passen).	**zz**:=frequency(left(**anz_br**,**br**),seq(*n*,*n*,0,9))
Das Ergebnis ist eine Liste mit den Anzahlen der Brötchen mit 0...9 Brötchen, also hier 1 Brötchen ohne Rosine, 2 Brötchen mit genau einer Rosine…	{ 1,2,3,1,8,6,2,1,1,0,0 }

Mit einer leichten Änderung kann man nun die Zahl $\frac{1}{e}$ experimentell bestimmen:

Durch die Definition der Anzahl der Brötchen = Rosinenanzahl erhält man einen interessanten Zusammenhang bei steigender Rosinen/Brötchenzahl: Der Anteil von Brötchen ohne Rosine nähert sich der Zahl $\frac{1}{e}$, wenn die Anzahl der Brötchen der Anzahl der Rosinen entspricht, dies wird hier über den Schieberegler na realisiert.	**Per "Hand"** Anzahl der Brötchen **br**:=**na** ▸ 550. Anzahl der Rosinien **ro**:=**na** ▸ 550. **ls**:=randInt(1,**br**,**ro**) ▸ {136.,217.,156.,550.,517.,358.,172.,528.,378.,303.,204.,482.,229.,32 **anz_br**:=frequency(**ls**,seq(*n*,*n*,1,**br**)) ▸ {3,0,0,0,1,0,1,0,1,3,0,0,0,0,2,1,1,1,0,2,1,2,3,0,1,0,0,1,0,2,0,1,1,1,3,0,0,1 **zz**:=frequency(left(**anz_br**,**br**),seq(*n*,*n*,0,9)) ▸ {201,196,111,37,4,1,0,0,0,0,0} $\frac{zz[1]}{1.\cdot br}$ ▸ 0.365455 $e^{-1.}$ ▸ 0.367879 na =550. 100. 800.

Ein weiteres Verfahren erlaubt eine systematischere Untersuchung des Sachverhaltes. Dazu wird ein neues Problem und eine leere Notes-Seite angelegt.

Die Urliste **br** ist die zufällige Verteilung (von hier 50) Rosinen auf die Brötchen 1...25.	**br**:=randInt(1,25,50) ▸ {6,16,7,1,9,14,18,
Die List **bin** sind die 25 verschiedenen Brötchen.	**bin**:=seq(*n*,*n*,1,25) ▸ {1,2,3,4,5,6,7,8,9,10,
Mit dem *frequency()* wird die Anzahl der Rosinen pro Brötchen gezählt. Der Befehl liefert eine Liste von 26 Zahlen, jeweils die Anzahl von Rosinen pro Brötchen und die Anzahl der Elemente der Urliste, die in kein Bin passen.	**fr**:=frequency(**br**,**bin**) ▸ {3,1,1,3,2,1,4,3,4,1,1,0,0,6,2,2,4,3,0,0,3,1,1,1,3,0}
Nun wird die Anzahl der Nullen gezählt und eins subtrahiert, da die letzte Null nicht mitgezählt werden darf. In diesem Fall sind vier Brötchen ohne Rosine.	**br**:=randInt(1,25,50) ▸ {6,16,7,1,9,14,18,14,15,3,9,5,5,7,16,11,1,7,9,4,8,7,25,17,21,24,4,8,14,2 **bin**:=seq(*n*,*n*,1,25) ▸ {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25} **fr**:=frequency(**br**,**bin**) ▸ {3,1,1,3,2,1,4,3,4,1,1,0,0,6,2,2,4,3,0,0,3,1,1,1,3,0} countIf(**fr**,?=0)−1 ▸ 4
Durch das wiederholte Drücken auf die oberste Zeile lässt sich das Experiment neu durchführen.	**br**:=randInt(1,25,50) ▸ {18,8,1,15,20,3,23,21,24,25,18,25,11,10,15,15,20,13 **bin**:=seq(*n*,*n*,1,25) ▸ {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20, **fr**:=frequency(**br**,**bin**) ▸ {2,4,1,1,1,3,2,3,1,1,1,0,2,4,4,1,1,2,2,5,1,3,1,1,3,0} countIf(**fr**,?=0)−1 ▸ 1

Die oben vorgestellte Methode ist enorm mächtig und erlaubt uns die Untersuchung einer leicht modifizierten Fragestellung:

Wie viele Rosinen müssen in den Teig, damit mit einer Wahrscheinlichkeit von 99% in jedem Brötchen mindestens eine Rosine ist?

Dazu wird die Urliste und die Häufigkeitsliste in eine Funktion umdefiniert, damit bei jedem Aufruf die Zählung einer neuen Liste durchgeführt wird. Mit fr(10) werden 10 Rosinen verteilt und die Anzahl der Rosinen in jedem Brötchen angezeigt.	$br(x):=randInt(1,25,x)$ ▸ *Fertig* $bin:=seqn(n,25)$ ▸ {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25} $fr(x):=frequency(br(x),bin)$ ▸ *Fertig* $fr(10)$ ▸ {0,1,0,1,0,0,0,1,0,1,0,2,0,1,0,0,1,0,0,0,1,0,0,0,1,0} $countIf(fr(10),?=0)$ ▸ 17
Die Funktion *countIf()* bestimmt die Anzahl der Elemente einer Liste, die mit einer Bedingung übereinstimmen. In diesem Fall kommt die null 17mal vor.	
Nun sollen alle Brötchen ohne Rosine gezählt werden, da nur die Durchläufe von Interesse sind, in dem in jedem Brötchen mindestens eine Rosine liegt. Dazu wird eine Zählfunktion definiert Die Zählfunktion liefert 1 wenn x > 0, sonst 1.	$z(x):=\begin{cases}0, x>0\\1, x=0\end{cases}$ ▸ *Fertig*
Nun wird die Simulation mit 50 Rosinen 1000mal durchgeführt. In diesem Fall gab es 14 Durchläufe, bei denen in jedem Brötchen mindestens eine Rosine war.	$\sum_{k=1}^{1000}\left(z(countIf(fr(k),?=0)-1)\right)$ ▸ 14
Um die Frage „Wie viele Rosinen müssen in den Teig, damit mit einer Wahrscheinlichkeit von 99% in jedem Brötchen mindestens eine Rosine ist?“ zu beantworten, erhöht man einfach die Zahl der Brötchen, bis man in die Nähe der gewünschten Prozentzahl kommt. Dies ist ab etwa 190 Brötchen der Fall.	$br(x):=randInt(1,25,191)$ ▸ *Fertig* $bin:=seq(k,k,1,25)$ ▸ {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25} $frequency(br(x),bin)$ ▸ {9,6,8,11,9,7,7,9,9,12,11,7,7,4,5,6,6,8,7,6,7,3,11,6,10,0} $fr(x):=frequency(br(x),bin)$ ▸ *Fertig* $countIf(fr(x),?=0)-1$ ▸ 0 $z(x):=\begin{cases}0, x>0\\1, x=0\end{cases}$ ▸ *Fertig* $\sum_{k=1}^{100}\left(z(countIf(fr(x),?=0)-1)\right)$ ▸ 99

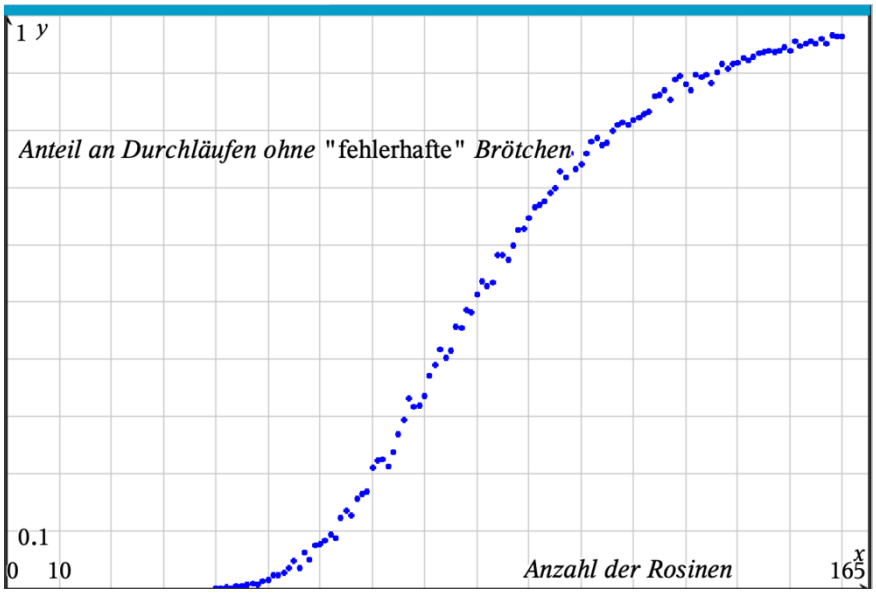
Hinweis: Beachten Sie, dass die Fragestellung: Wie viele Rosinen muss man in 25 Brötchen tun, damit ein (bestimmtes) Brötchen mit 99%iger Sicherheit mindestens eine Rosine enthält, zu einem anderen Ergebnis führt, da es um eine andere Wahrscheinlichkeit geht.

$$\text{solve}\left(1-\left(\frac{24}{25}\right)^{n}>0.99,n\right) \blacktriangleright n>112.81$$

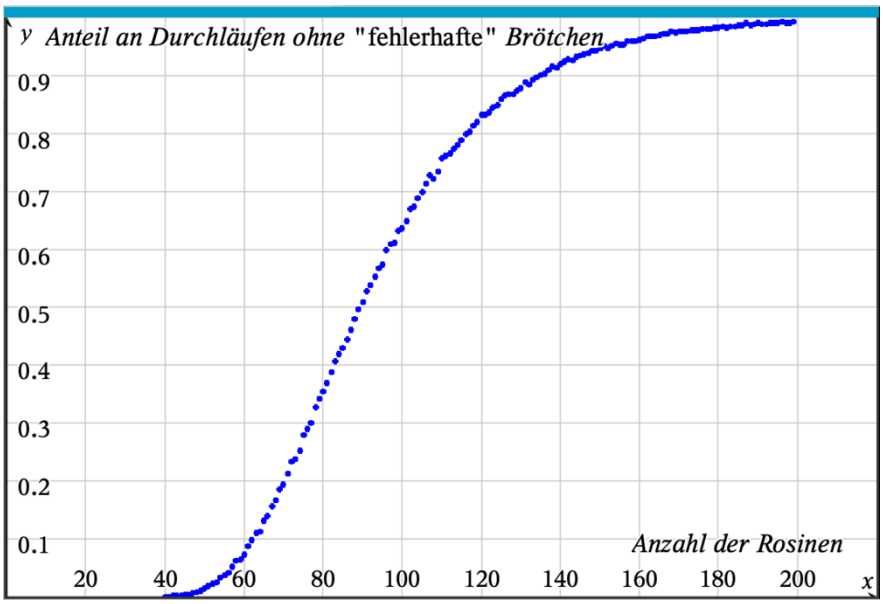
Aufbauend auf dieser Simulation lässt sich das Experiment nun für verschiedene Anzahlen von Rosinen durchführen.

Dazu wird der Sequenz seq() Befehl benutzt.

Die Anzahl der Rosinen wird in der erzeugenden Funktion für die Urliste eingebaut.	$\mathbf{br}(x):=\text{randInt}(1,25,x)$
Um eine Folge zu erhalten, wird der seq() Befehl auf die Summe angewendet. Die Laufvariable n des seq()-Befehls ist die Anzahl der Rosinen. Beachten Sie, dass die Anzahl der Durchgänge hier auf 100 reduziert wurde und nur 50, 51, ...,60 Rosinen verteilt werden, da die Ausführung des Befehls viel Zeit in Anspruch nehmen kann.	$\text{seq}\left(\sum_{k=1}^{100}\left(\mathbf{z}\left(\text{countIf}\left(\mathbf{fr}(n),?=0\right)-1\right)\right),n,50,60\right)$ $\{0,1,1,2,3,3,6,7,5,9,9\}$
Um relative Häufigkeiten zu erhalten, wird die Liste mit 0,01 multipliziert	$\text{seq}\left(\sum_{k=1}^{100}\left(\mathbf{z}\left(\text{countIf}\left(\mathbf{fr}(n),?=0\right)-1\right)\right),n,50,60\right)\cdot 0.01$ $\blacktriangleright \{0.03,0.03,0.04,0.04,0.02,0.01,0.02,0.03,0.04,0.1,0.07\}$
Nun sollen alle Werte von 40 bis 160 Rosinen untersucht werden. Dazu wird der Befehl in ein *Calculatorfenster* kopiert, um die mehrfache Ausführung zu verhindern. Die Anzahl der Simulationen wird auf 1000 gesetzt.	
Nach einer Rechenzeit von etwa 2min lassen sich die Ergebnisse in einem Diagramm auftragen. Dazu werden die Ergebnisse der Simulation unter pp, und eine Liste der Zahlen von 40...160 unter nn gespeichert.	$pp:=\{0.,0.001,0.002,0.$ $\{0.,0.001,0.002,0.001,$ $nn:=\text{seq}(n,n,40,160)$ $\{40,41,42,43,44,45,46,$



Mit einer leistungsfähigeren Programmiersprache wie Python lässt sich das Verfahren auch für erheblich mehr Durchgänge realisieren. Hier mit jeweils 10000 Durchläufen.



Es ergibt sich, dass im Bereich zwischen 190 und 200 Rosinen die Anzahl liegt, bei der mit einer Wahrscheinlichkeit von mehr als 99% in jedem Brötchen mindestens eine Rosine liegt.

Dies kann auch durch eine Rechnung bestätigt werden:

$$\left(\text{binomCdf}\left(nn,\frac{1}{25},1,nn\right)\right)^{25} \blacktriangleright 0.99018$$

nn =192.

1. 300.

$$\text{solve}\left(\left(1-\left(\frac{24}{25}\right)^{n}\right)^{25}>0.99,n\right) \blacktriangleright n>191.54$$

Mit dem $1/\sqrt{n}$-Gesetz ergibt sich, dass – um auf ein 95% Progonoseintervall von 0,1% zu kommen – mindestens 1000000 Simulationen benötigt werden.
Mit einem veränderten python-Programm lässt sich die Speicherbeschränkung der Software umgehen und die Simulation beliebig oft wiederholen. Allerdings nehmen die Rechenzeiten stark zu. Rechts ist das Ergebnis von 10^6 Durchgängen zu sehen.

	A rosinen	B pp2	C
=			
1	189	0.988944	
2	190	0.989395	
3	191	0.989765	
4	192	0.990216	
5	193	0.99058	

Literaturverzeichnis

1) *Engel, Arthur*: „Wahrscheinlichkeitsrechnung und Statistik, Band 1“, Klett Studienbücher, Stuttgart 1973, Seite 49/ 50
2) *Borovcnik, Manfred*: „Das Sammelbildproblem – Rosinen und Semmeln und Verwandtes: Eine rekursive Lösung mit Irrfahrten“ in: *Stochastik in der Schule,* Heft 2/2007, S. 19-24

Autoren:
Dr. Hubert Langlotz
Sebastian Rauh
Dr. Wilfried Zappe

