Programmierkurs Julia
Johannes Nägele
Lehrstuhl Prof. Dr. Schlather
FSS 2024

Übungsblatt 9

Aufgabe 1 (7 + 5 Punkte)

Zufallszahlen continued.

(a) Wir wollen nun unseren eigenen Zufallszahlengenerator programmieren. Die verwendete Methode nennt man LCG (Linear Congruental Generator). Dabei soll für einen Startwert $x_0 \in \mathbb{R}$ und die Konstanten $a, c, m \in \mathbb{R}$ gelten

$$x_{n+1} = (ax_n + c) \bmod m, \quad \forall n \in \mathbb{N}.$$

Für gute Startwerte sind die x_n/m approximativ uniform verteilt. Schreibe daher eine Funktion `next(x, a, c, m)`, welche die nächste (Pseudo-)Zufallszahl x_{n+1} berechnet.

(b) Berechne 10^7 Zufallszahlen mit den Parameterwerte $a = 69069$, $c = 1$, $m = 2^{32}$ sowie dem Seed $x_0 = 123456789$ und speichere die x_n/m in einem Array ab.

(c) Plotte ein Histogramm und erstelle einen Scatterplot für n vs. x_n/m (Tipp: Verwende nur die ersten 1000 Zahlen). Kombiniere beide Plots in einer Grafik.

(d) Bonusaufgabe (5 Punkte):

Nutze deine Samples um normalverteilte Zahlen zu bekommen (Inversionsmethode; verwende `quantile` und `Normal` aus `Distributions.jl`). Erstelle wieder ein Histogramm und prüfe über einen Verteilungstest, ob die gesampleten Zahlen auh tatsächlich normalverteilt aussehen (verwende dafür `ExactOneSampleKSTest` aus `HypothesisTests`). Teste alternativ die Werte $a = 16807$, $c = 1$, $m = 2^{32}-1$.

Aufgabe 2 (9 Punkte)

Diese Aufgabe dreht sich um die Logistische Gleichung

$$x_{n+1} = r \cdot x_n(1 - x_n),$$

wobei $x_0 \in [0, 1]$, $r \in [0, 4]$ und $n \in N$. Für $r \leq 3.0$ hat diese Differenzengleichung einen Fixpunkt, danach oszilliert x_n zwischen mehreren Werten.

(a) Schreibt eine Funktion `next(r, x)`, welche x_{n+1} in Abhängigkeit von r und x_n berechnet.

(b) Plottet $x_0, ...x_{50}$ für $r = 3.1$ und $x_0 = 0.5$.

(c) Schreibt eine Funktion `oscillations(x, r)` schreiben, welche

(a) zunächst 1000 Iterationen mit dem Startwert `x` durchführt (um ungefähr konvergiert zu sein),

(b) dann die nächsten 500 Folgenglieder in einem Array speichert sowie

(c) diesen Array dann auf mehrfach vorkommende Werte filtert und zurückgibt (Das sind unsere gesuchten Oszillationen. Achtung: Werte kommen nur approximativ doppelt vor; verwende `unique` in Kombination mit `round` und Runde auf 4 Nachkommastellen).

(d) Erstellt einen Scatterplot (ein sog. Bifurkationsdiagramm) für `oscillations` mit `r` aus `0.0:0.001:4.0` (Tipp: Eine kleine `markersize` könnte hilfreich sein; entfernt entweder die Umrisse der Punkte oder sorgt dafür, dass Umrisse und Inneres der Punkte die gleiche Farbe haben).

Aufgabe 3 (3 Punkte)

Schreibe ein Makro `@todolist`. Dieses soll für

```
@todolist Buy groceries, Wash the dishes, Do laundry
```

Folgendes ausprinten:

```
TODO:
1. Buy groceries
2. Wash the dishes
3. Do laundry
```

Tipp: Gebt Euch zunächst via `dump` im Makro aus, wie eigentlich der Input aussieht. Am praktischsten lässt sich die Ausgabe wahrscheinlich mit `enumerate` organisieren.

Aufgabe 4 (Zusatzaufgabe zur Bonusvorlesung, 3 Punkte)

Überlege Dir kurz, warum das Beispiel Typ Rechteck, Subtyp Quadrat wieder ein Beispiel dafür ist, dass Vererbung unpraktisch sein kann.

Viel Erfolg!