# Lösung der Zusatzaufgabe

- Erweitern Sie `lnkng()` um einen Parameter, so dass dass die Anzahl der zurückzugebenden Parameter wählbar ist.

- **Lösung:**

```
lnkng <- function (a, m, c, start, n)
  {
    result <- rep(NA, n)
    result[1] <- (a*start+ c)%% m
    for (i in 2:n) result[i] <- (a*result[i-1]+ c) %% m
    result
  }
```

# Weitere Verfeinerungen in der Konstruktion von LKGen

- Ergebnisse aus der Zahlentheorie schränken die Wahl für gute Parameter weiter ein.
- Ohne Details (s. Knuth!) gilt:
- Ist $m = 2^e$, so muss $a = 3$ oder $a = 5 \mod 8$ (bei $c = 0$) gelten.
- Wählt man $m = 10^e, e > 5$, so erhält man die maximal mögliche Periode $5 \times 10^{e-2}$ für $X_0$ kein Vielfaches von 2 oder 5 und $a \mod 200$ ist einer der Werte 3, 11, 13, 19, 21, 27, 29, 37, 53, 59, 61, 67 , 69, 77, 83, 91, 109, 117, 123, 131, 133, 139, 141, 147, 163, 171, 173, 179, 181, 187, 189 oder 197.

- Heute üblich ist, $m = 2^p - 1, p$ prim, zu wählen, da diese Wahlen zu großen Periodenlängen führen. Die Zahlen $2^p - 1$ sind sehr oft Primzahlen, die sogenannten Mersenne-Primzahlen. Die bisher größte gefunden Primzahl ist auch von dieser Art ist $2^{43112609} - 1$ mit ca. 13 Mio Stellen!

- Problem bei allen LKGen ist, dass nach mehr als $\sqrt{m}$ entnommenen Zufallszahlen die "Zufälligkeit" stark abnimmt.

- Dies ist nur eine Faustregel, in der Literatur findet man auch $m/1000$ als Grenze.

- Deshalb wurden Methoden entwickelt, die Ausbeute an Zufallszahlen weiter zu steigern.

# Andere Methoden der Zufallszahlengenerierung

- Ein verallgemeinerter Ansatz findet sich in den multipel rekursiven Generatoren, die von mehr als einem vorherigen Wert, nämlich einer ganzen Teilmenge der Vergangenheit abhängen:

$$X_n = f(X_{n-i_1}.X_{n-i_2}, \ldots, ).$$

- Das älteste Beispiel für einen solchen Ansatz ist der Fibonacci-Generator (1958)

$$X_n = (X_{n-1} + X_{n-2}) \mod m.$$

  Oft gibt dieser sogar Periodenlängen größer als $m$! Wieso? (Leider scheitert dieser elegante Generator an einfachen statistischen Tests, wie wir noch sehen werden.)

- Andere Verfahren sezten auf die Permutation des Zufallszahlenstroms, den ein Generator erzeugt.

- Weiterhin gibt es nichtlineare Kongruenzgeneratoren, bspw.

- Eichenhauer/Lehn (1986) bei dem $x_{n-1}$ ersetzt wird durch das inverse Element in $\mathcal{Z}_p$:

$$x_n = ax_{n-1}^- + c \mod m.$$

- oder auch Knuth (1998) mit der naheliegenden Idee eines quadratischen Generators

$$x_n = ax_{n-1}^2 + bx_{n-1} + c \mod m.$$

- Der aktuell als das Nonplusultra geltende Generator ist ein auf Mersenne-Primzahlen basierender Generator von Makoto Matsumoto und Takuji Nishimura (1996/7) mit Periodenlänge $2^{19937} - 1$!

- Dieser heißt auch liebevoll "das große Monster" oder Twisted Mersenne Twister!

- Dieser ist auch der Standardgenerator in `R`!

# Aufgabe:

Implementieren Sie den Fibonacci-Generator als `R Funktion`!

- Nutzen Sie hier auch das zweistufige Vorgehen.

  1. Im ersten Schritt **eine** Zufallszahl aus zwei gegebenen.
  2. Im zweiten Schritt $\mathbf{n}$ Zufallszahlen aus zwei Startwerten.

# Lösung:

```
rfib1 <- function ( x1, x2, m) { (x1+ x2) %% m}

rfib <- function (n, x1, x2, m)
  {
    result <- rep(NA, n)
    result[1] <- rfib1( x1,x2,m)
    result[2] <- rfib1(result[1], x2, m )
    for (i in 3:n) result[i] <- rfib1( result[i-1], result[i-2], m)
    result
  }
```

# Zusatzaufgabe: RANDU

- Implementieren Sie RANDU! Für RANDU gilt

$$X_{n+1} = 65539 X_n \mod 2^{31}, X_0 \text{ ungerade.}$$

- RANDU gilt als der vielleicht schlechteste Zufallsgenerator der je kommerziell vertrieben wurde. Er gehörte zur Standardausstattung von IBM Großrechnern seit den 1960er Jahren, bis in die 1970er hinein.

- Viele wissenschaftliche Arbeiten der Zeit, die mit Simulationen gearbeitet haben, gelten seitdem als suspekt.

- Starten Sie RANDU mit Startwert (oder auch *seed* $x_0 = 1$, erzeugen Sie 50 Zahlen. Was beobachten Sie?

- Das Hauptproblem ist allerdings ein anderes: Erzeugt man dreidimensionale Punkte, indem man $x_n, x_{n+1}, x_{n+2}$ als Koordinaten eines Punktes in $\mathcal{R}^3$ auf, so liegen alle Punkte auf genau 15 Ebenen!

- Erzeugen Sie 100000 Punkte in $\mathcal{R}^3$ mit RANDU und einem guten Startwert ($x_0 \neq 1$) und plotten Sie diese Punkte mit Hilfe des `R` Paketes `scatterplot3d`! Können Sie die Ebenen entdecken?

# Tests auf Zufälligkeit - Vorüberlegungen

- Wir haben gesehen, dass die menschliche Intuition sehr schlecht ist, um Zufälligkeit zu beurteilen.

- Der nahe liegende Ausweg sind statistische Tests.

- Da wir aber nur Mimikry mit dem Zufall betreiben, sagt das Bestehen eines statistischen Tests für einen Zufallsgenerator nichts über das Bestehen eines anderen Test aus.

- "Bestehen" bedeutet hier, dass der Erwartungswert einer Teststatistik für die Gleichverteilung gleich dem "Erwartungswert" der entsprechenden Teststatistik für die von einem Generator erzeugten Pseudozufallszahlen ist.

- In der Praxis testet man einen neuen Generator an einer Serie von Standardtests (5-10).

- Bewährt sich der Generator in diesen Tests, gilt er als unverdächtig, bis jemand eine Schwachstelle findet.

- Bespielhaft wird dies am Fibonacci-Generator gezeigt werden.

# Theoretische Tests

- Einen Generator nur *a posteriori* beurteilen zu können, ist eine prinzipiell unbefriedigende Situation.

- Schön wäre, wenn man z.B. für einen linearen Kongruenzgenerator anhand der Paramter direkt seine Güte ablesen könnte.

- Beispiele für solche Ergebnisse wurden für die Zykluslänge gegeben.

- Im Laufe der Jahrzehnte konnten (einige) Tests entwickelt werden, die die *vollständige* Sequenz von Zufallszahlen eines Generators untersuchen.

- Es scheint theoretisch sehr schwer zu sein, sinnvolle Dinge über kurze Sequenzen von Zufallszahlen in Bezug auf ihre "Zufälligkeit" auszusagen. Da die Mathematik dieser Tests sehr aufwändig ist, gehen wir hier nicht weiter in die Tiefe.

# Tests auf Zufälligkeit - Allgemeine Tests

- Die erste Möglichkeit besteht natürlich darin, die allgemein verfügbaren Tests auf Verteilung zu nutzen.
- Namentlich sind dies der $\chi^2$-Test und den Kolmogorov-Smirnoff-Test.
- Zur Erinnerung: $\chi^2$-Test ist für diskrete Verteilungen, KS-Test für stetige Verteilungen. Durch Diskretisierung kann der $\chi^2$-Test auch für jede stetige Verteilung genutzt werden.
- Details siehe Statistik II!

# Beispiel: Das Scheitern des Fibonacci-Generators

- Der Fibonacci Generator scheitert am KS-Test.

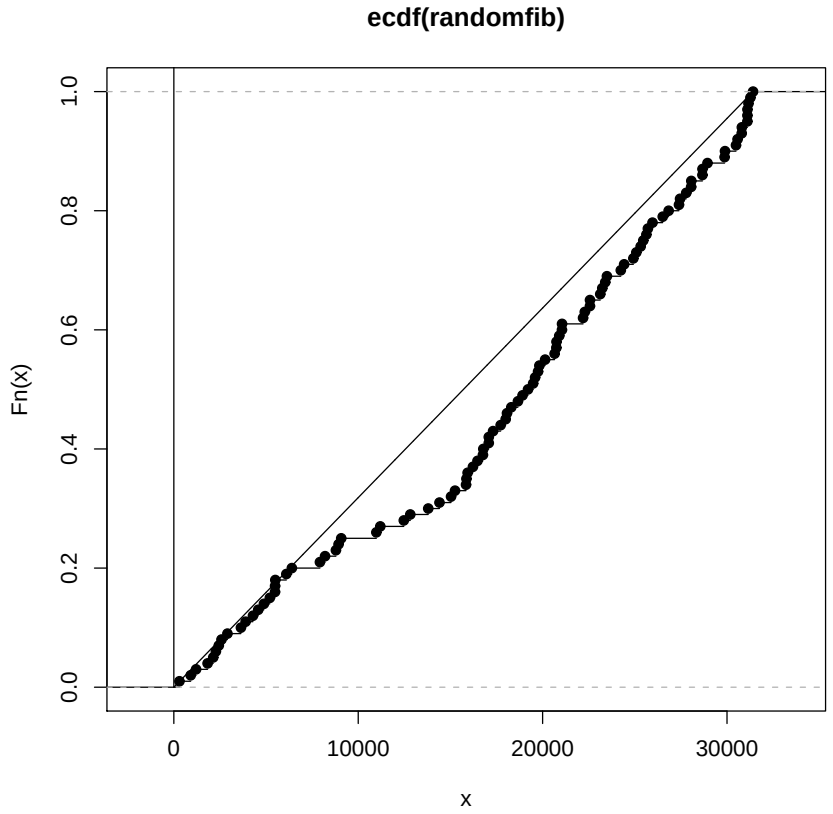
```
> randomfib <- rfib(1000, 1, 1, 31415)
> randomfib <- randomfib[901:1000]
### Die ersten 900 Werte werden als sog. burn-in genutzt
> ks.test(randomfib, "punif", 0, 31414)

One-sample Kolmogorov-Smirnov test

data:  randomfib
D = 0.1745, p-value = 0.004544
alternative hypothesis: two-sided
> ks.test(runif(100), "punif")
> plot(ecdf(randomfib))
> lines(seq(0,31414), seq(0,31414)/31414)
```

# Vergleich der Verteilungsfunktionen

Die emp. Verteilungsfunktion des Fibonacci-Generators und die der Gleichverteilung.



# Würfeln mit Fibonacci-Zahlen

- Generiert man mit denselben Fibonacci-Zahlen Würfelwürfe, so bestehen diese den $\chi^2$-Test!

```
> randomdice <- randomfib %% 6
> table(randomdice)
randomdice
 0  1  2  3  4  5
11 22 13 22 12 20
> chisq.test(c(11,22,13,22,12,20))

Chi-squared test for given probabilities

data:  c(11, 22, 13, 22, 12, 20)
X-squared = 8.12, df = 5, p-value = 0.1497
```

# Empirische Tests - $ad\ hoc$ Methoden

- Im Folgenden ist $(U_n)$ stets eine Folge von unabhängig identisch auf [0, 1] gleichverteilten Zufallszahlen. Für Tests, die auf ganzen Zahlen operieren, sei $(Y_n) = (\lfloor dU_n \rfloor)$, die Folge der auf $\{0, 1, \ldots, d\text{-}1\}$ gleichverteilten ganzen Zahlen, für einen angemessenen Wert von $d$.

- Es gibt einen ganze Reihe derartiger Tests.

- Während der klassische $\chi^2$-Test jeweils nur einzelne Glieder der Folgen $(U_n)$, bzw. $(Y_n)$ betrachtet, erweitern die folgenden Methoden den Ansatz jeweils auf $n$-Tupel von Folgengliedern, betrachten also mehrere Zufallszahlen mit ihrer gemeinsamen Verteilung.

# Der *serial test*

.

- Beim Serial-Test betrachtet man Paare von ganzen Zahlen $(Y_i, Y_{i+1}), (Y_{i+2}, Y_{i+3}), \ldots$ und zählt die Häufigkeit, mit der die Paare $(q, r), q, r \in \{0, 1, \ldots, d-1\}$ auftreten. Mit diesen Paaren wird dann der klassische $\chi^2$-Test durchgeführt.

- Im Wesentlichen handelt es sich also beim Serial-Test um den $\chi^2$-Test für alle $d^2$ möglichen Ausprägungen von Paaren erzeugter Zufallszahlen.

- Der Ansatz lässt sich natürlich auf Tripel, Quadrupel etc. erweitern.

- Bei der Durchführung muss darauf geachtet werden, dass der realisierte Stichprobenumfang $n$ so groß ist, dass man eine valide $\chi^2$-Teststatistik erhält. Z. B. $n > 5d^2$ für Paare. (Besetzung der Zellen!)

- Da der benötigte Stichprobenumfang, um $k$-Tupel von Zufallszahlen zu betrachten, exponentiell mit $k$ wächst, gibt es abgeschwächte Varianten,um die höherdimensionalen Verteilungseigenschaften von Generatoren zu überprüfen.

- Diese abgeschwächten Varianten beschränken sich auf bestimmte Eigenschften der Tupel.

# Empirische Tests - Der Pokertest

- Auch beim Pokertest wird im Grunde ein $\chi^2$ Test durchgeführt.
- Betrachtet werden Quintupel $(Y_i, Y_{i+1}, Y_{i+2}, Y_{i+3}, Y_{i+4})$.
- Dann zählt man im klassischen Pokertest das Auftreten der Ereignisse: Alle verschieden $abcde$, ein Paar $aabcd$, zwei Paare $aabbc$, ein Dreier $aaabc$, *full house* $aaabb$, ein Vierer $aaaab$ oder ein Fünfer $aaaaa$.
- Leichter zu programmieren ist eine abgeschwächte Form, bei der nur gezählt wird, wie viele verschiedene Elemente das Quintupel enthält. Diese Variante läuft auch unter dem Namen *Pokertest*.
- Die Wahrscheinlichkeiten für die einzelnen Felder der $\chi^2$-Tafel lassen sich kombinatorisch herleiten.

- Oder:
  **Aufgabe:** Schreiben Sie ein Programm, dass diese Wahrscheinlichkeiten für den vereinfachten Poker-Test simuliert ($d = 100$ und 10000 Wiederholungen)!
  Tipp: Nutzen Sie die Funktion `unique()`!

# Empirische Tests - Maximum-von-t-Test

- Ein Test, der nicht auf dem $\chi^2$-Test zurückgeführt beruht, ist der Maximum-von-$t$-(Zahlen) Test.

- Bei diesem Test wird ausgenutzt, dass für $0 \leq j < n$ gilt:

$$\text{Wenn } V_j := \max\{U_{tj}, U_{tj+1}, \ldots, U_{tj+t-1}\}, \text{ dann } F(V_j) = V_j^t.$$

- Für die Folge $V_0, , \ldots, V_{n-1}$ kann dann der KS-Test gegen die Verteilung $F(V_t)$ durchgeführt werden.

- Dieser Test beruht darauf, dass

$$P(\max\{U_1, \ldots, U_t\} \leq x) = x^t.$$

- Und viele, viele mehr! Die Thematik ist hier nur angeschnitten worden.

- Stichworte: Kollisonstest, Run-Test, Gap-Test, Korrelationstests ....