Lösung zur Aufgabe zum Pokertest

```
set.seed(1)
results <- rep (NA, 10000)
for (i in 1:10000){
  results[i] <- length(unique(floor(100*runif(5))))
}
table(results)
>results
   3    4    5
  25  958 9017
```

Lösung zur RANDU Aufgabe

```
randu <- function(n,start){
result <- rep(NA,n)
result[1] <- start
for (i in 1:(n-1)) result[i+1] <- (65539*result[i])%%2^31
result
}

rtest <- randu(9999,5)
rtest.m <- matrix(rtest,ncol=3,byrow=TRUE)
scatterplot3d(x=rtest.m[,1],y=rtest.m[,2],z=rtest.m[,3])
```

- Leider kann man mit `scatterplot3d` keine Struktur erkennen ...

Empirische Tests - Spektraltest

- Die bisherigen Untersuchungen zeigen, dass die Verteilung von Tupeln konsekutiver Zufallszahlen eines Generators das entscheidende Kriterium bei der Beurteilung der Güte eines Generators ist.

- Der Spektraltest untersucht nun die Eigenschaften der Folge (U_n) mit bekannter Periode m, indem für alle m Punkte $0 \leq n < m$ die Menge aller t-Tupel $\{(U_n, U_{n+1}, \ldots, U_{n+t-1})\}$ betrachtet wird.

- Der Spektraltest setzt hier direkt bei den Eigenschaften der gemeinsamen Verteilung t-Tupel an.

- Dieser Test ist besonders deshalb bedeutend, weil bisher alle als schlecht erkannten Zufallsgeneratoren an ihm scheitern!

- Er ist eine Art Zwitter zwischen theoretischem und empirischem Test.

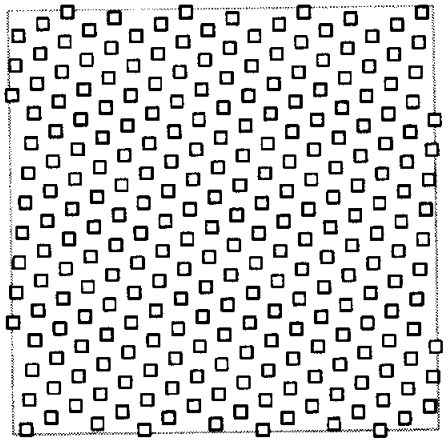
- Auf der einen Seite leitet Eigenschaften für die komplette Periode eines Generators her, auf der andere Seite benötigt man einen Rechner, um die kritischen Größen zu berechnen.

- Man betrachte den Generator $s(x) = (137x + 187) \mod 256$.

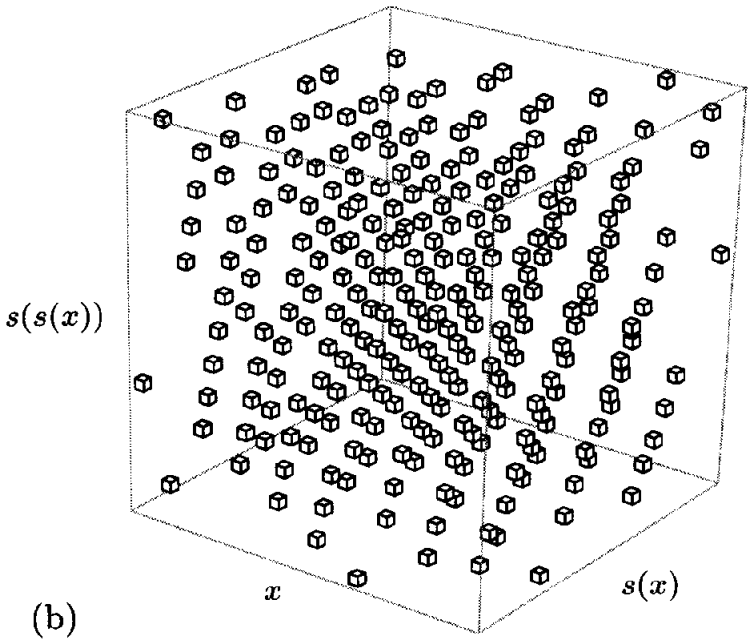
- Ein t-Tupel dieses Generators läßt sich dann schreiben als

$$(x, s(x), s(s(x)), \ldots, s^{t-1}(x)).$$

- Die Eigenschaft, bei der der Spektraltest ansetzt, kann am Besten in einem Bild gesehen werden.



(a)



(b)

Fig. 8. (a) The two-dimensional grid formed by all pairs of successive points (X_n, X_{n+1}), when $X_{n+1} = (137X_n + 187) \bmod 256$. (b) The three-dimensional grid of triplets (X_n, X_{n+1}, X_{n+2}).

- Auf den ersten Blick scheint es, als könnte etwas, das ein so regelmässiges Muster zeigt, nicht zufällig sein.

- Überlegt man aber, dass die Daten als Vielfache von $\frac{1}{256}$ (bzw. im allgemeinen Fall $\frac{1}{m}$) erzeugt werden, dann ist klar, dass für jeden linearen Kongruenzgenerator ein solches Muster zu erwarten ist.

- Entsprechend würden sich auch "echte" Zufallszahlen, die man auf eine bestimmte Stelle rundet auf einem ähnlichen Gitter realisieren. Jede Zufallszahl wäre nach Rundung ein Vielfaches einer Grundeinheit $\frac{1}{\nu}$ für irgendeine ganze Zahl ν. Diese könnte z.B. $\nu = 2^w$ sein, wenn w die Wortlänge des Rechners ist.

- Wenn man das Muster beschreiben möchte, dann kann man leicht sehen, dass es sich um Anordnungen von Punkten handelt, die im $\mathcal{R}^2$ von Familien von Geraden überdeckt werden und im $\mathcal{R}^3$ von Ebenen.

- Allgemein, wenn man in höhere Dimensionen geht, kann man jeweils Familien von Hyperbenen finden, die alle Punkte enthalten.

- Definiert man $\frac{1}{\nu_2}$ als den maximalen Abstand paralleler Geraden, die in $\mathcal{R}^2$ **alle** Punkte der Form $\{$(x/m, s(x)/m) $\}$ überdecken. Dann nennt man ν_2 die *Genauigkeit* oder *Auflösung* des Generators $s()$ in $\mathcal{R}^2$.

- Entsprechend definiere man für größere t den Wert $\frac{1}{\nu_t}$ als den maximalen Abstand von Hyperebenen die alle Punkte der Form $\{(x, s(x), s(s(x)), \ldots, s^{t-1}(x)\}$ enthalten. Dann heißt ν_t die t-dim. Auflösung des Generators $s()$.

- Der entscheidende Unterschied zwischen "echten" Zufallszahlen und Pseusozufallszahlen ist nun, dass für "echte" Zufallszahlen die Auflösung in allen Dimensionen gleich bleibt.

- Bei Pseudozufallszahlen jedoch wird durch die Periodenlänge m festgelegt, dass zum einen die Auflösung abnimmt, wenn t wächst, und dass zum anderen die t-dim. Auflösung begrenzt ist durch ca. $m^{1/t}$.

- Die Periodenlänge beschränkt also direkt die Auflösung.

- Bei der praktischen Durchführung des Tests beschränkt man sich auf Dimensionen $t < 10$, oft sogar auf $2 \leq t < 6$!

- Achtung! Es reicht nicht, den Test nur für eine passende hohe Dimension t durchzuführen. Die Tests umschliessen sich nicht!

- Die Idee des Spektraltests ist nun, die empirische Auflösung eines Generators mit der theoretischen Auflösung zu vergleichen.

- Aber wie berechnet man diese Werte?

- An dieser Stelle wäre ein ziemlich langer Beweis nötig. (s. Knuth, Vol 2, P. 98ff)

- Hier soll reichen: Es geht! Und zwar sogar ohne die ganze Sequenz der Zufallszahlen explizit zu betrachten.

- In der Praxis ist der Algorithmus für Dimensionen $t > 9$ nicht anwendbar, da die Rechenzeit mit $O(3^t)$ wächst.

- Das Gute ist, man muss diesen Alg. ja nicht von Grund auf implementieren!

- Außerdem sind etliche Standardgeneratoren bereits untersucht!

- Bespielsweise gilt für RANDU:
  $\nu_2^2 = 536936458, \nu_3^2 = 118, \nu_4^2 = \nu_5^2 = \nu_6^2 = 116$

- Als Kriterium für einen guten Generator hat sich eine Grenze von

$$\nu_t \geq 2^{30/t} \text{ für } 2 \leq t \leq 6.$$

  Also z.B. $\nu_3^2 = 2^{20} \approx 10^6 >> 118$.

- Der Spektral-Test für Zufallszahlen ist also nicht im Sinne der Statisitk ein Test, sondern eine von der Zeit geprüfte Heuristik!

- In der Literatur findet sich anstelle der Auflösung oft auch die Zahl von Hyperebenen, auf denen die Zufallszahlen liegen. Diese Zahl ist aber nicht rotationsinvariant und deshalb schlecht zu interpretieren.

Erzeugung nicht gleichverteilter Zufallszahlen

- Im Folgenden wird nun vorrausgesetzt, dass ein perfekter Generator für Folgen (U_n) und (Y_n) aus der stetigen bzw. diskreten Gleichverteilung vorliegt. In `R` ist dies `runif()`.

- Wie kommt man nun von der stetigen Gleichverteilung zu beliebigen Verteilungen?

- Zunächst zu Erinnerung einige Eigenschaften der stetigen Gleichverteilung auf [0, 1], $U_{[0,1]}$.

- Ist $X \sim U_{[0,1]}$, dann gelten

$$f(x) = 1 \ \ \text{, wenn } 0 \leq x < 1, 0 \ \text{ sonst.}$$

$$F(x) = P(X \leq x) = x, \text{, wenn } 0 \leq x \leq 1.$$

Beliebige diskrete Verteilungen

- Zur Erzeugung beliebiger diskreter Verteilungen exisitiert ein triviales Konzept.
- Betrachte eine diskrete ZV X mit Ausprägungen $x_1, x_2, \ldots, x_k$, die mit W'keiten $p_1, p_2, \ldots, p_k$ auftreten.
- Zu einer Realisierung u einer $U_{[0,1]}$ ZV wählt man nun als Realisierung von X: x_1 für $0 \leq u < p1$, x_2 für $p_1 \leq u < p_1 + p_2, \ldots,$ und x_k für $\sum_1^{k-1} p_i \leq u < 1$.
- Für spezielle Verteilungen gibt es oft geschickt angepasste Verfahren, die die nötige Summation effizient umsetzen.

Aufgaben zu diskreten Zufallsgeneratoren

- Zeigen Sie, dass eine gemäß vorstehendem Algorithmus erzeugte Zufallszahl der gewünschten Verteilung gehorcht!

- Implementieren Sie einen Zufallsgenerator für einen Münzwurf, bei dem Kopf, Zahl und Kante mit vorzugebenden Wahrscheinlichkeiten realisiert werden!

Stetige Verteilungen mit invertierbarer Verteilungsfunktion

- Es gilt für jede stetige Zufallsvariable X mit Verteilungsfunktion F, dass

$$F(X) \sim U_{[0,1]}.$$

- Beweis:
  $F_{F(X)}(x) = P(F(X) < x) = P(F^{-1}(F(X)) < F^{-1}(x)) =$
  $= P(X < F^{-1}(x)) = F(F^{-1}(x)) = x\square.$

- X besitze eine invertierbare Verteilungsfunktion F, so dass geschrieben werden kann

$$y = F(x) \Leftrightarrow x = F^{-1}(y).$$

- Dann gilt
$$F^{-1}(U) \sim F_X.$$

- Beweis:
$$P(F^{-1}(U) < x) = P(U < F(x)) = F(x)\Box.$$

- Die Methode, diese Eigenschaft auszunutzen, heißt *Inversionsmethode*.

- Optimal ist natürlich, wenn man die Umkehrfunktion F^{-1} analytisch herleiten und angeben kann. Im Prinzip funktioniert die Methode aber auch, wenn man nur mit hinreichender Genauigkeit die Inverse numerisch berechnen kann!

Beispiele:

- Betrachten Sie die Verteilung des Maximum-von-t-Tests.
- Mit $P(U < x) = x$ für $U \sim U_{[0,1]}$ sieht man leicht, dass

$$P(X =: \max(U_1, U_2, \ldots, U_t) \leq x) = x^t = F(X)$$

  für i.i.d. ZV U_i.
- Also ist $F^{-1}(y) = y^{1/t}$.
- Für den Spezialfall $t = 2$ sieht man also, dass

$$X := \sqrt{U} \text{ und } X := \max\{U_1, U_2\}$$

  dieselben Verteilungen besitzen, was nicht offensichtlich ist.

Aufgabe zur Inversionsmethode

- Leiten Sie einen Generator für exponentialverteilte ZVen her!
- Zur Erinnerung die Verteilungsfunktion lautet in diesem Fall

$$F(x) = 1 - e^{-\lambda x}, x \geq 0.$$

Acceptance-Rejection-Method (ARM)

- Diese Methode geht ebenfalls direkt auf John von Neumann zurück.
- Ziel ist eine ZV X gemäß der Dichte $f_X()$ zu erzeugen. Es sei nun $g()$ eine andere Dichte, für die man einen effizienten Generator vorliegen hat.
- Weiterhin exisitiere ein Konstante $c \in \mathcal{R}$, so dass für alle t gelte:

$$f(t) \leq cg(t).$$

- Es gibt also eine Konstante c, so dass f vollständig von cg *dominiert* dominiert wird.
- Die ARM beschreibt nun einen Algorithmus, mittels Zufallszahlen aus dem bekannten Generator für G und aus $U_{[0,1]}$ einen Generator für F zu erzeugen.

Acceptance-Rejection-Method (ARM): Algorithmus

1. Man erzeugt eine Zufallsgröße x aus G und ein u aus $U[0,1]$.

2. Wenn $u \geq f(x)/cg(x)$, dann wird die Zufallszahl x verworfen und bei Schritt 1 neu begonnen.

3. Wenn $u < f(x)/cg(x)$ wird x als ZV aus der Verteilung F akzeptiert.

- Es ist relativ leicht zu zeigen, dass eine so gewählte Zahl tatsächlich der entsprechenden Verteilung angehört.

- Je kleiner c gewählt werden kann, desto effizienter ist diese Methode.

Beispiel: Normalverteilung mit der ARM

- Die Inversionsmethode ist für die Normalverteilung nicht anwendbar, da die Verteilungsfunktion keine geschlossen angebbare Gestalt besitzt.

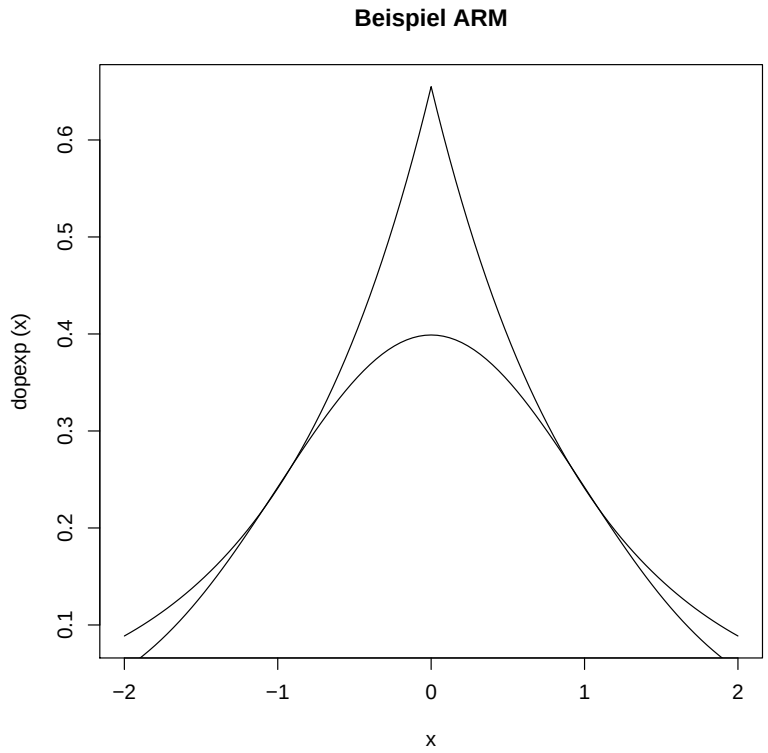
- Bekanntlich ist $f(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}x^2)$ die Dichte der Standardnormalverteilung. Aufgrund der Symmetrie kann man sich auf den Teil $x > 0$ beschränken und nachträglich mit einer weiteren Zufallszahl das Vorzeichen bestimmen.

- Nun wähle man $g(x) = \frac{1}{2} \exp(-|x|)$. $g(x)$ ist die Dichte der sog. *doppelten Expoentialverteiltung*.

- Dann wähle dominierendes c mittels:

$$\frac{f(x)}{g(x)} \leq \sqrt{\frac{2e}{\pi}} \approx 1.31 < 1.32 =: c.$$

Dichte und dominierende Funktion



Beispiel: Normalverteilung mit der ARM

- Der eigentliche Algorithmus lässt sich dann sehr kompakt darstellen:

  1. Generiere drei Zufallszahlen u_1, u_2, u_3 aus $U[0, 1]$.
  2. Generiere eine ZV x gemäß g mit der Inversionsmethode: $x = -ln(u_1)$.
  3. Verwerfe x falls $u_2 > \frac{f(x)}{1.32g(x)}$. Gehe zu Schritt 1.
  4. Falls $u_3 < 0.5$, setze $x = -x$.
  5. x ist eine Zufallszahl gemäß N(0, 1)!
  6. **Aufgabe:** Nutzen Sie `runif()`, um einen Generator für die Standardnormalverteiliung zu erzeugen!

Andere Verteilungen

- Die hier vorgestellten Methoden sind sehr breit verwendbar.
- Sie nutzen keine speziellen Eigenschaften der zu generierenden Verteilungen, sondern nutzen ganz allgemeine Prinzipien.
- Es ist oft möglich zu einer bestimmten Verteilung effizienter Zufallszahlen zu erzeugen, wenn man "Tricks" für den Spezialfall entwickelt!
- Ein Blick in die Literatur lohnt immer, sobald es laufzeitkritisch wird, die Zufallszahlen schnell zu erzeugen!