Projektaufgaben Block 1

Carlo Michaelis, <Matr.-Nr.>; Lukas Ruff, 572521

15 November 2016

1 Infinite-Monkey-Theorem

1.1 Formulierung und Beweis des Infinite-Monkey-Theorems

Laut dem Infinite-Monkey-Theorem wird ein Affe, der unendlich lange zufällig auf einer Schreibmaschine tippt, fast sicher jede beliebige Zeichenkette unendlich oft schreiben. Diese bildhafte Interpretation des mathematischen Satzes soll der gedanklichen Einordnung von Unendlichkeit dienen. Das Infinite-Monkey-Theorem ist ein Beispiel für die Anwendung des Lemmas von Borel-Cantelli.

Satz 1 (Lemma von Borel-Cantelli)**.** *Sei $(\Omega, \mathcal{F}, P)$ ein Wahrscheinlichkeitsraum.*

a) Ist $(A_n)_{n\in\mathbb{N}}$ eine Folge von Ereignissen mit $\sum_{n\in\mathbb{N}} P(A_n) < \infty$, so gilt

$$P\left(\limsup_{n\to\infty} A_n\right) = 0.$$

b) Ist $(A_n)_{n\in\mathbb{N}}$ eine Folge von unabhängigen Ereignissen mit $\sum_{n\in\mathbb{N}} P(A_n) = +\infty$, so gilt

$$P\left(\limsup_{n\to\infty} A_n\right) = 1.$$

Um das Infinite-Monkey-Theorem formulieren zu können, modellieren wir zunächst einen geeigneten Wahrscheinlichkeitsraum. Definiere dazu ein endliches Alphabet $\Omega := \{\text{a, b, c, } \ldots\}$, d.h. Ω ist eine Menge von Zeichen (Buchstaben, Satzzeichen, Zahlen, etc.) mit $|\Omega| = n$ für ein $n \in \mathbb{N}$. Setze weiter die zugehörige σ-Algebra als Potenzmenge $\mathcal{F} := \mathcal{P}(\Omega)$ und definiere das Wahrscheinlichkeitsmaß P als diskrete Gleichverteilung, d.h. $P(A) = \frac{|A|}{|\Omega|} = \frac{|A|}{n}$ für $A \in \mathcal{F}$. Somit modelliert $(\Omega, \mathcal{F}, P)$ das Ziehen von Zeichen aus einem Alphabet mit gleicher Wahtscheinlichkeit. Den Übergang zu Zeichenketten (Folgen von Zeichen aus dem Alphabet Ω) können wir nun mittels des (stets existierenden und eindeutigen) Produktraumes $(\Omega^{\mathbb{N}}, \mathcal{F}^{\mathbb{N}}, \mathbb{P})$ mit $\mathbb{P} := P^{\mathbb{N}}$ bewerkstelligen. Hiermit können wir das Infinite-Monkey-Theorem wie folgt formulieren und beweisen:

Satz 2 (Infinite-Monkey-Theorem)**.** *Betrachte den oben definierten Produktraum $(\Omega^{\mathbb{N}}, \mathcal{F}^{\mathbb{N}}, \mathbb{P})$ von (unendlich langen) Zeichenketten und sei $s = (s_1, \ldots, s_m)^\top \in \Omega^m$ eine beliebige Zeichenkette der Länge $m \in \mathbb{N}$ (String). Definiere zu $k \in \mathbb{N}$ das Ereignis*

$$A_k := \left\{\omega = (\omega_j)_{j\in\mathbb{N}} \in \Omega^{\mathbb{N}} : \omega_k = s_1, \ldots, \omega_{k+m-1} = s_m\right\},$$

d.h. A_k ist das Ereignis, dass String s der Länge m an der Stelle k in einer Zeichenkette beginnt. Dann ist $(A_{jm+1})_{j\in\mathbb{N}_0}$ eine Folge unabhängiger Ereignisse und es gilt

$$\mathbb{P}\left(\limsup_{j\to\infty} A_{jm+1}\right) = 1.$$

Beweis. Es sei $\pi_i : \Omega^{\mathbb{N}} \to \Omega, (\omega_j)_{j\in\mathbb{N}} \mapsto \omega_i$ die i-te Koordinatenprojektion auf dem kartesischen Produkt $\Omega^{\mathbb{N}}$. Weiter sei $(I_j)_{j\in\mathbb{N}_0}$ eine Zerlegung von $\mathbb{N}$ in disjunkte Blöcke der Länge m, d.h.

$$\mathbb{N} = \bigcup_{j\in\mathbb{N}_0}^{\cdot} I_j \qquad \text{mit} \qquad I_j := \bigcup_{k=1}^{m} \{jm + k\},\ j \in \mathbb{N}_0.$$

Dann folgt für $j \in \mathbb{N}_0$

$$\begin{aligned}\mathbb{P}(A_{jm+1}) &= \mathbb{P}\left(\left\{\omega : \omega_{jm+1} = s_1, \ldots, \omega_{(j+1)m} = s_m\right\}\right) \\ &= \mathbb{P}\left(\bigcap_{k=1}^{m} \pi_{jm+k}^{-1}(\{s_k\})\right) \\ &= \prod_{k=1}^{m} P(\{s_k\}) = \left(\frac{1}{n}\right)^m,\end{aligned}$$

da $\mathbb{P} = P^{\mathbb{N}}$ Produktmaß ist. Weiter folgt für $j_1, j_2 \in \mathbb{N}_0, j_1 \neq j_2$:

$$\begin{aligned}\mathbb{P}\left(A_{j_1 m+1} \cap A_{j_2 m+1}\right) &= \mathbb{P}\left(\left\{\omega : \omega_{j_1 m+1} = s_1, \ldots, \omega_{(j_1+1)m} = s_m\right\} \cap \left\{\omega : \omega_{j_2 m+1} = s_1, \ldots, \omega_{(j_2+1)m} = s_m\right\}\right) \\ &= \mathbb{P}\left(\left(\bigcap_{k=1}^{m} \pi_{j_1 m+k}^{-1}(\{s_k\})\right) \cap \left(\bigcap_{k=1}^{m} \pi_{j_2 m+k}^{-1}(\{s_k\})\right)\right) \\ &= \left(\prod_{k=1}^{m} P(\{s_k\})\right)^2 = \left(\frac{1}{n}\right)^m \left(\frac{1}{n}\right)^m = \mathbb{P}(A_{j_1 m+1}) \cdot \mathbb{P}(A_{j_2 m+1}),\end{aligned}$$

da $I_{j_1} \cap I_{j_2} = \varnothing$ für $j_1 \neq j_2$. Somit ist $(A_{jm+1})_{j \in \mathbb{N}_0}$ eine Folge unabhängiger Ereignisse mit

$$\sum_{j \in \mathbb{N}_0} P(A_{jm+1}) = \sum_{j \in \mathbb{N}_0} \left(\frac{1}{n}\right)^m = +\infty.$$

D.h. mit Teil b) des Lemma von Borel-Cantelli folgt die Behauptung. □

Es gilt also

$$\begin{aligned}\mathbb{P}\left(\limsup_{j \to \infty} A_{jm+1}\right) &= \mathbb{P}\left(\bigcap_{j \in \mathbb{N}_0} \bigcup_{k \geq j} A_{km+1}\right) \\ &= \mathbb{P}\left(\left\{\omega \in \Omega^{\mathbb{N}} : \forall j \in \mathbb{N}_0\, \exists k \geq j : \omega \in A_{km+1}\right\}\right) \\ &= \mathbb{P}\left(\left\{\omega \in \Omega^{\mathbb{N}} : \omega \in A_{jm+1} \text{ für unendlich viele } j \in \mathbb{N}_0\right\}\right) = 1,\end{aligned}$$

d.h. ein beliebiger String s der Länge m erscheint fast sicher unendlich oft.

1.2 Simulation eines Infinite-Monkey Experiments

In diesem Abschnitt wollen wir das Infinite-Monkey-Theorem experimentell simulieren. Wir schreiben hierzu eine R-Funktion die solange zufällig Zeichen aus dem Alphabet $\Omega = \{a, b, \ldots, y, z\}$ mit gleicher Wahrscheinlichkeit (d.h. $\frac{1}{26}$) auswählt, bis eine vorgegebene Zeichenkette $s = (s_1, \ldots, s_m)^\top \in \Omega^m, m \in \mathbb{N}$, vollständig erschienen ist. Die R-Funktion gibt dann die Anzahl der bis zum Erscheinen der Zeichenkette s gezogenen Zeichen zurück. Damit simulieren wir die Zufallsvariable $X : \Omega^{\mathbb{N}} \to \mathbb{N}$ mit

$$X(\omega) = \min\{k + m - 1 : \omega_k = s_1, \ldots, \omega_{k+m-1} = s_m \text{ für } k \in \mathbb{N}\}$$

Wir haben die Funktion wie folgt in R implementiert:[1]

[1] Eine effizientere Implementierung wäre je Schleifendurchlauf eine größere Anzahl an Zeichen zu generieren und den resultierenden Zeichen-Block mit einem Fenster nach `strTarget` zu durchsuchen. Dabei muss beachtet werden, dass `strTarget` auch in der Überlappung zweier Blöcke erscheinen könnte. Für unsere Untersuchung ist eine vereinfachte (ineffiziente) Implementierung jedoch ausreichend. Für große Samples oder einen langen String `strTarget` sollte jedoch auf jeden Fall eine effizientere Implementierung herangezogen werden.

```r
fnInfiniteMonkey <- function(strTarget) {
  # This function is a simulation of the Infinite-Monkey-Theorem. It generates a
  # random sequence of letters until a given target string appears.
  #
  # Args:
  #   strTarget: The target string which should be matched
  #
  # Returns:
  #   The number of generated letters until the target string appeared

  # Split target string to vector of chars
  vecCharTarget <- strsplit(strTarget, "")[[1]]
  # Get the number of letters in target string
  nTarget <- length(vecCharTarget)

  # Set counting variable (at least nTarget letters needed)
  nCounter <- nTarget

  # Switch on the monkey (i.e. sample the first nTarget letters)
  vecLetterSeqTail <- sample(letters, nTarget, replace = TRUE)

  # Let the monkey type until target string was written
  while(!identical(vecCharTarget, vecLetterSeqTail)) {

    # Let the monkey hit another key (sample next letter) and store in
    # vecLetterSeqTail (first in, first out)
    if (nTarget == 1) {
      vecLetterSeqTail <- sample(letters, 1)
    } else {
      vecLetterSeqTail <- c(vecLetterSeqTail[2:nTarget],
                            sample(letters, 1, replace = TRUE))
    }
    # Count
    nCounter <- nCounter + 1
  }

  # Return the length of the generated letter sequence
  return(nCounter)
}
```

In Figure 1 haben wir ein Histogramm der simulierten Anzahlen von Zeichen und die durchschnittliche Anzahl von Zeichen, bis die Zeichenkette `'ab'` vollständig erschienen ist, für 10^4 Samples geplottet. Wir können sehen, dass die Anzahl von Zeichen X einer rechtsschiefen Verteilung folgt. Die durch das starke Gesetz der großen Zahlen motivierte Monte-Carlo-Approximation von $\mathbb{E}[X]$ liegt bei 686.9719.

Zur Simulation von Infinite-Monkey Experimenten wollen wir abschließend noch folgende Überlegung durchführen: würden wir anstatt einer fortlaufenden Zeichenfolge, bei welcher je Schleife nur ein weiterer Letter generiert wird, je Schleifendurchlauf einen Zeichen-Block der Länge m generieren und mit s vergleichen, so entspräche jeder Schleifendurchlauf einem Bernoulli-Experiment mit Erfolgswahrscheinlichkeit $p = \left(\frac{1}{n}\right)^m$. In diesem Fall wäre die Anzahl der Blöcke, die notwendig sind um eine Übereinstimmung mit s zu erhalten, gerade geometrisch-verteilt mit Erfolgswahrscheinlichkeit $p = \left(\frac{1}{n}\right)^m$. Diese Zerlegung in Blöcke der Länge m entspricht genau der Folge unabhängiger Ereignisse aus dem Beweis des Theorems.

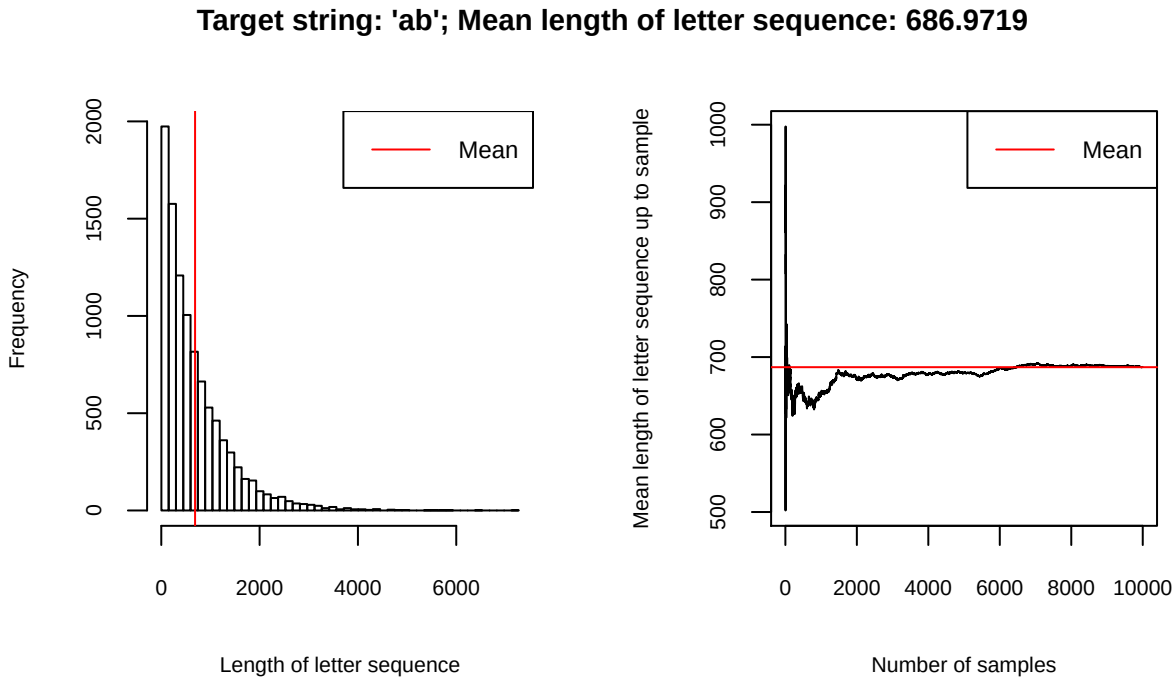

Figure 1: Histogram of letter sequence lengths and mean letter sequence length by number of samples.

2 Monte-Carlo-Approximationen

3 Eine (naive) Datenanalyse

4 Normalverteilung

In diesem Abschnitt wollen wir verschiedene Zufallsgeneratoren, d.h. deterministische Algorithmen zur Erzeugung von pseudo-zufälligen Zahlen, vergleichen. Unser Ziel ist die Erzeugung von unabhängigen standard-normalverteilten (Pseudo-)Zufallszahlen $Z \sim N(0,1)$.

Um die „Güte“ der erzeugten Zufallszahlen zu überprüfen verwenden wir Histogramme, QQ-Norm-Plots und zwei statistische Tests auf Normalverteilung (Kolmogorov-Smirnov-Test[2] und Shapiro-Wilk-Test[3]). Wir haben diese Überprüfungen in einer R-Funktion zusammengefasst:

```r
fnCheckNormality <- function(vecSample) {
  # This function helps to check a sample of generated random numbers for
  # normality. It helps comparing different random number generators by using
  # histograms, qq-norm-plots, Kolmogorov-Smirnov-tests, and Shapiro-Wilk-test.
  #
  # Args:
  #   vecSample: A sample of generated random numbers
  #
  # Returns:
  #   -

  # Split plot panel
  par(mfrow = c(1,2), ps = 9,
      cex.axis = 0.9,
      cex.lab = 0.9)

  # Plot histogram
  hist(vecSample,
       breaks = seq(min(vecSample), max(vecSample), length = 25),
       probability = TRUE,
       xlim = c(-5, 5),
       ylim = c(0, 0.5),
       xlab = "Samples from random number generator",
       main = "Histogram of random samples")

  # Add density of standard normal to the plot
  vecNorm <- seq(-5, 5, by = 0.1)
  vecNormDensity <- dnorm(vecNorm)
  lines(vecNorm, vecNormDensity, col = "blue")
  legend("topright", legend = "Density of standard normal",
         lty = 1, col = "blue", cex = 0.75)

  # QQ-norm-plot
  qqnorm(vecSample, ylim = c(-5, 5), cex = 0.8)
  qqline(vecSample)

  # Perform Kolmogorov-Smirnov test
  print(ks.test(vecSample, "pnorm"))

  # Perform Shapiro-Wilk test for normality
  print(shapiro.test(vecSample))
}
```

[2] https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test
[3] https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test

Als Benchmark überprüfen wir zunächst die in R mit der Funktion `rnorm` standardmäßig implementierte Methode zur Erzeugung standard-normalverteilter Zufallsvariablen:

```
# Test with rnorm-function
set.seed(42)
fnCheckNormality(rnorm(5000))
```

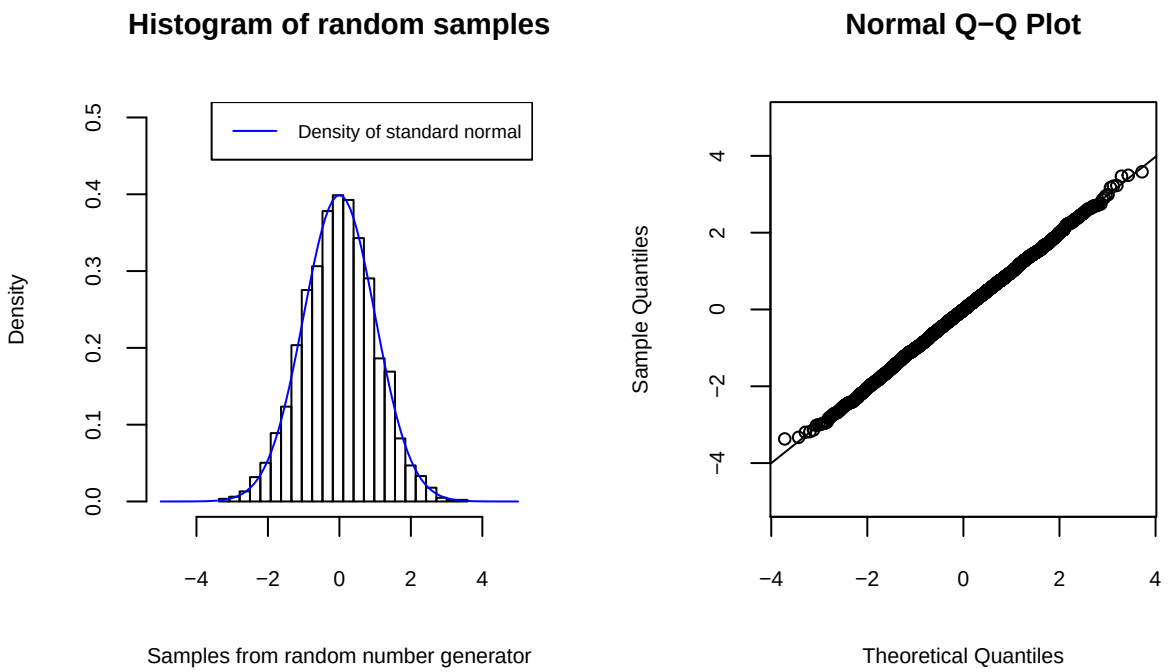

Figure 2: Histogram and QQ-Norm-plot for random numbers generated with `rnorm`.

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  vecSample
## D = 0.0089805, p-value = 0.8148
## alternative hypothesis: two-sided
##
##
##  Shapiro-Wilk normality test
##
## data:  vecSample
## W = 0.99971, p-value = 0.744
```

Wir können sehen, dass Histogramm und Dichtefunktion der Standard-Normalverteilung sowie die theoretischen mit den empirischen Quantile sehr gut übereinstimmen. Dieser optische Eindruck wird durch die beiden statistischen Tests unterstützt: in beiden Tests liegen die p-Werte deutlich über den gewöhnlichen Signifikanzniveaus α von 0.05 oder 0.01. D.h. in beiden Tests kann die Nullhypothese, dass die Zufallszahlen normalverteilt sind, nicht abgelehnt werden.

Im Folgenden wollen wir drei verschiedene Methoden zur Erzeugung von standard-normalverteilten Zufallsvariablen vergleichen. Alle drei Methoden setzen die Erzeugung von unabhängigen $U([0,1])$-verteilten Zufallsvariablen voraus, wovon wir annehmen, dass diese zuverlässig in R erzeugt werden können.[4]

[4]Die Methoden zur Erzeugung von Zufallszahlen in R finden sich in der Dokumentation unter „RNG“ (Random Number Generation), welche auch unter https://stat.ethz.ch/R-manual/R-devel/library/base/html/Random.html eingesehen werden kann.

4.1 Box-Muller-Methode

Die Box-Muller-Methode nutzt aus, dass für $U_1, U_2 \overset{iid}{\sim} U([0,1])$ die Transformationen $Z = Z_1 := \sqrt{-2\log U_1}\cos(2\pi U_2), Z_2 := \sqrt{-2\log U_1}\sin(2\pi U_2)$ unabhängige standard-normalverteilte Zufallsvariablen sind.[5] Wir überprüfen $Z = Z_1$:

```
# Generate random numbers from the standard normal using the Box-Muller-Method
set.seed(42)
nSamples <- 5000

vecU1 <- runif(nSamples)
vecU2 <- runif(nSamples)
vecZ1 <- sqrt((-2)*log(vecU1)) * cos(2*pi*vecU2)

# Test for normality
fnCheckNormality(vecZ1)
```

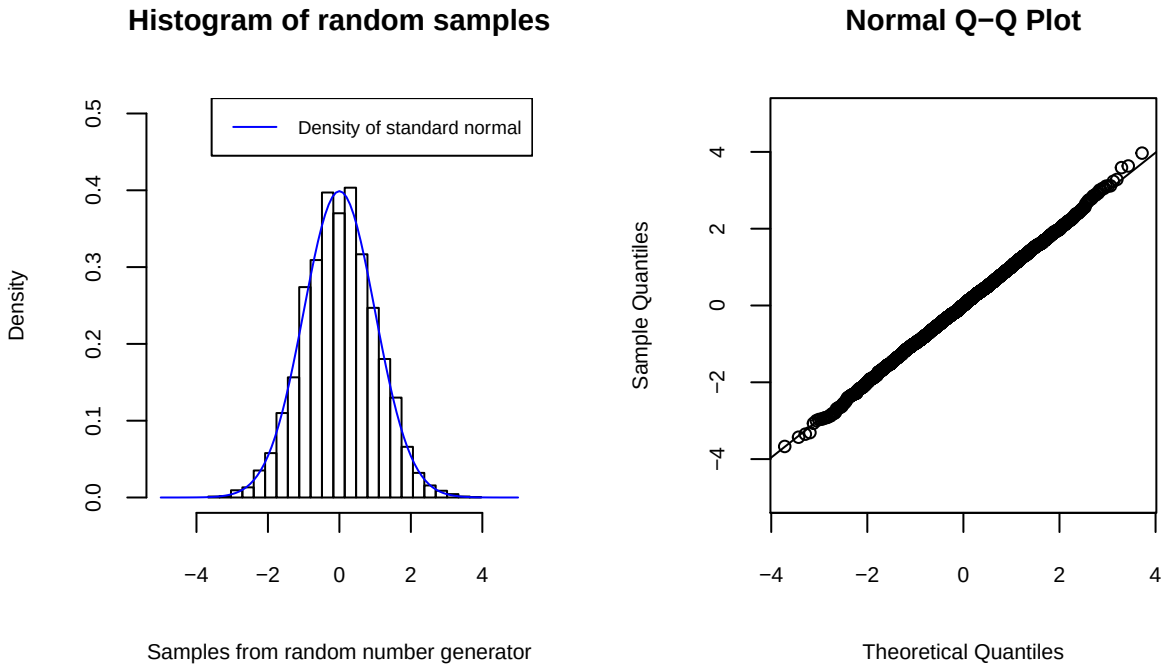

Figure 3: Histogram and QQ-Norm-plot for random numbers generated with the Box-Muller method.

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  vecSample
## D = 0.009996, p-value = 0.6998
## alternative hypothesis: two-sided
##
##
##  Shapiro-Wilk normality test
##
## data:  vecSample
## W = 0.99974, p-value = 0.8267
```

[5] Es ist $(Z_1, Z_2) = (R\cos(2\pi U_2), R\sin(2\pi U_2))$ die Polarkoordinatentransformation mit Radius $R := \sqrt{-2\log U_1}$. Die Aussage lässt sich dann mit Hilfe des Dichtetransformationssatzes nachweisen.

Wiederum stimmen Histogramm und Dichtefunktion sowie die theoretischen mit den empirischen Quantilen sehr gut überein. Auch die Tests können zum Signifikanzniveau $\alpha = 0.05$ oder $\alpha = 0.01$ nicht abgelehnt werden, da die p-Werte deutlich größer sind.

4.2 Summe stetig gleichverteilter Zufallsvariablen

Als zweite Methode wollen wir gleichverteilte Zufallszahlen $U_1, \ldots, U_m \overset{iid}{\sim} U([-1/2, 1/2])$ für ein $m \in \mathbb{N}$ erzeugen und $Z = \sum_{i=1}^{m} U_i$ setzen. Es ist $\mathbb{E}[U_1] = 0$ und $\mathrm{Var}(U_1) = 1/12$. Damit gilt nach dem zentralen Grenzwertsatz, dass die standardisierte Summe in Verteilung gegen die Standard-Normalverteilung konvergiert:

$$\sqrt{\frac{12}{m}} \sum_{i=1}^{m} U_i \xrightarrow{d} N(0,1).$$

Damit stimmt die standardisierte Summe mit der Summe $Z = \sum_{i=1}^{m} U_i$ für $m = 12$ überein. Wir werden nun überprüfen, ob $m = 12$ bereits hinreichend groß ist, um normalverteilte Zufallszahlen durch Z zu generieren[6]:

```
# Generate random numbers from the standard normal as sum of i.i.d.
# U([-0.5, 0.5])-distributed random variables

set.seed(42)
nSamples <- 5000
m <- 12  # number of i.i.d. random variables

matUnif <- matrix(runif(nSamples*m, -0.5, 0.5), ncol = m)
vecZ2 <- apply(matUnif, 1, sum)

# Test for normality
fnCheckNormality(vecZ2)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  vecSample
## D = 0.011851, p-value = 0.4837
## alternative hypothesis: two-sided
##
##
##  Shapiro-Wilk normality test
##
## data:  vecSample
## W = 0.9995, p-value = 0.2187
```

Das Histogramm und Dichtefunktion sowie die theoretischen mit den empirischen Quantilen stimmen wiederum gut überein. Auch die Tests können zum Signifikanzniveau $\alpha = 0.05$ oder $\alpha = 0.01$ nicht abgelehnt werden, da die p-Werte größer sind. Im Vergleich zur Box-Muller-Methode sind die p-Werte der Tests jedoch kleiner. Der QQ-Norm-Plot lässt des Weiteren vermuten, dass die Tails nicht so gut übereinstimmen.

4.3 Accept-Reject-Algorithmus mit Laplace-Kandidatendichte

Als letzte Methode wollen wir den Accept-Reject-Algorithmus mit der Dichte der Laplace-Verteilung

[6] Ein bekanntes Resultat über die Güte der Konvergenz im zentralen Grenzwertsatz ist der Satz von Berry-Esseen. Dieser besagt (in einfacher Form), dass wenn das dritte absolute Moment existiert (was in unserem Fall zutrifft), die Konvergenzrate der Ordnung $n^{-\frac{1}{2}}$ ist. (https://en.wikipedia.org/wiki/Berry%E2%80%93Esseen_theorem)

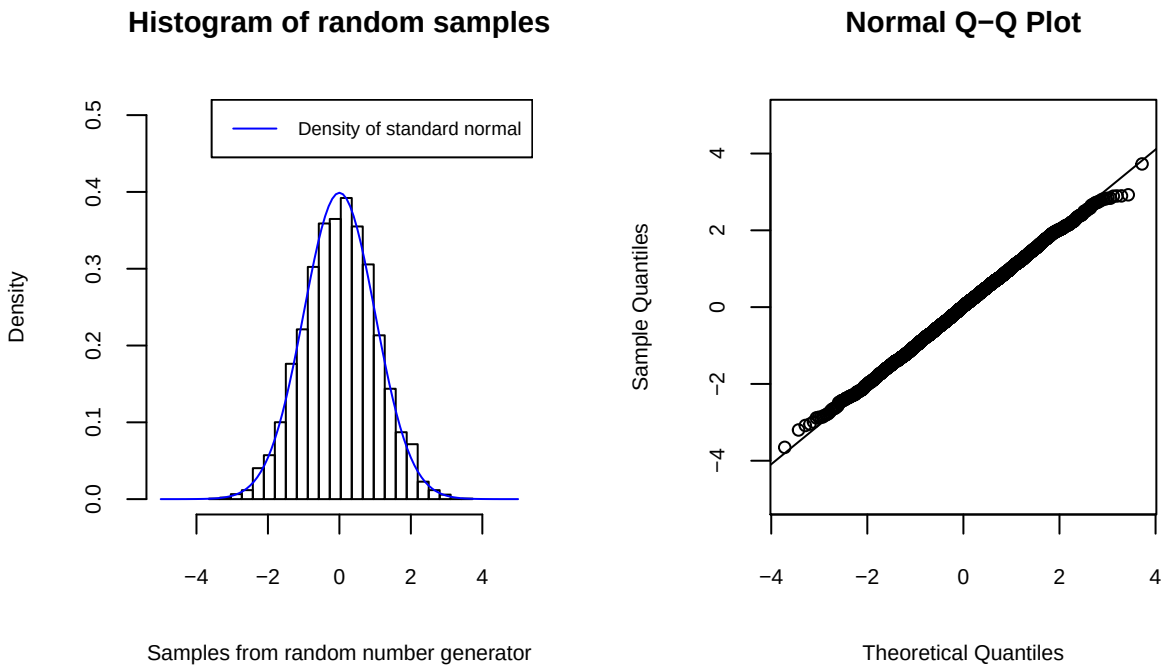

Figure 4: Histogram and QQ-Norm-plot for random numbers generated from the sum of $m = 12$ uniform random variables.

$$g_\alpha(x) = \frac{\alpha}{2}\exp(-\alpha|x|), \quad \alpha > 0,$$

als Kandidatendichte implementieren. Um Zufallszahlen der Laplace-Verteilung zu generieren, wenden wir die Inversionsmethode an. Für die Verteilungsfunktion F_α der Laplace-Verteilung gilt:

$$F_\alpha(x) = \int_{-\infty}^{x} \frac{\alpha}{2}\exp(-\alpha|y|)\,dy = \begin{cases} \int_{-\infty}^{x} \frac{\alpha}{2}\exp(\alpha y)\,dy = \frac{1}{2}\exp(\alpha x) & , x < 0, \\ \frac{1}{2}\exp(0) + \int_0^x \frac{\alpha}{2}\exp(-\alpha y)\,dy = 1 - \frac{1}{2}\exp(\alpha x) & , x \geq 0. \end{cases}$$

Somit folgt für die Inverse F_α^{-1}:

$$F_\alpha^{-1}(u) = \begin{cases} \frac{1}{\alpha}\log(2u) & , u < \frac{1}{2}, \\ -\frac{1}{\alpha}\log(2(1-u)) & , u \geq \frac{1}{2}. \end{cases}$$

Um Zufallszahlen einer Zielverteilung mit Dichte f zu erzeugen, muss für die Kandidatendichte g_α und eine Konstante M gelten, dass $\frac{f(x)}{g(x)} \leq M$ für alle $x \in \mathbb{R}$. Da die Akzeptanzwahrscheinlichkeit im Accept-Reject-Sampling durch $\frac{1}{M}$ gegeben ist, macht es Sinn M möglichst „scharf“ (d.h. klein) zu wählen, sodass weniger Samples abgelehnt werden. Definiere

$$h_\alpha(x) := \frac{f(x)}{g_\alpha(x)} = \sqrt{\frac{2}{\pi}} \cdot \frac{1}{\alpha} \cdot \exp\left(\alpha|x| - \frac{x^2}{2}\right), \quad x \in \mathbb{R},$$

da f in unserer Anwendung die Dichte der Standard-Normalverteilung ist. Dann ist h_α für $|x| = \alpha$ maximal, denn

$$\frac{\partial}{\partial x} h_\alpha(x) = h_\alpha(x) \cdot (\alpha \mathbb{1}_{(0,\infty)}(x) - \alpha \mathbb{1}_{(-\infty,0)}(x) - x) \overset{!}{=} 0$$
$$\iff x = \alpha \mathbb{1}_{(0,\infty)}(x) - \alpha \mathbb{1}_{(-\infty,0)}(x),$$

da $h_\alpha(x) > 0$ für alle $x \in \mathbb{R}$.

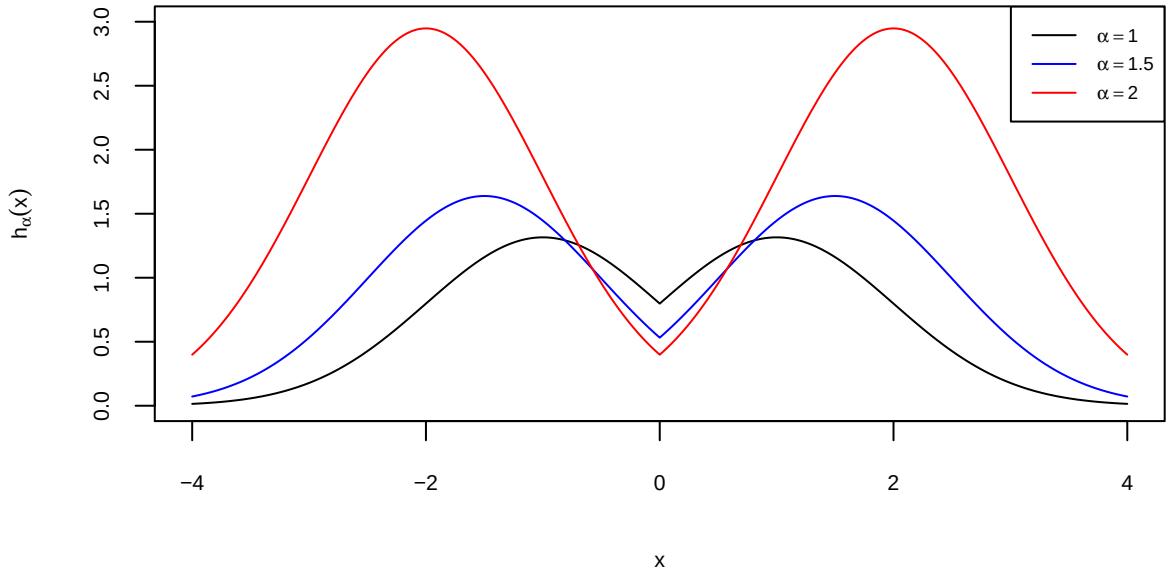

Figure 5: The function h_α for different values of α.

Somit ist h_α uniform durch $h_\alpha(\alpha) = \sqrt{\frac{2}{\pi}} \cdot \frac{1}{\alpha} \cdot \exp\left(\frac{\alpha^2}{2}\right)$ beschränkt. Weiter können wir diese Schranke in $\alpha > 0$ minimieren:

$$\frac{\partial}{\partial \alpha}\left(\sqrt{\frac{2}{\pi}} \cdot \frac{1}{\alpha} \cdot \exp\left(\frac{\alpha^2}{2}\right)\right) = \sqrt{\frac{2}{\pi}}\left(1 - \frac{1}{\alpha^2}\right)\exp\left(\frac{\alpha^2}{2}\right) \stackrel{!}{=} 0$$
$$\iff \alpha = 1.$$

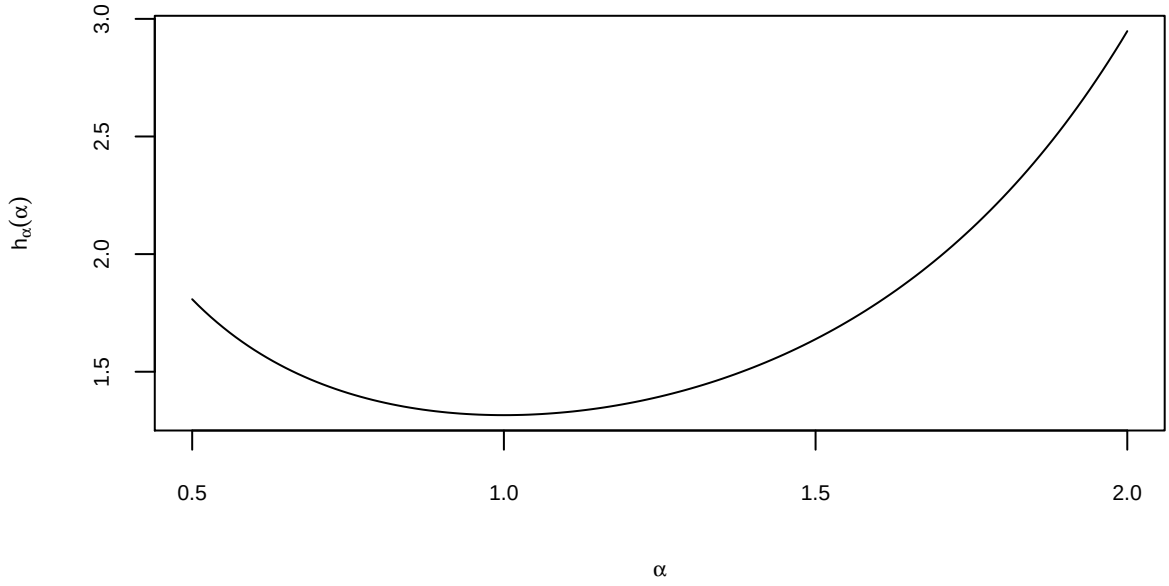

Figure 6: $h_\alpha(\alpha)$ as a function in α.

Somit wählen wir die scharfe Schranke $M = \sqrt{\frac{2}{\pi}} \exp(0.5) \approx 1.3155$ und können den Accept-Reject-Algorithmus wie folgt implementieren:

```r
# Generate random numbers from the standard normal using the
# accept-reject-algorithm with Laplace proposal density

# Set seed and number of samples
set.seed(42)
nSamples <- 5000

# Define Laplace density
g <- function(x, alpha = 1) {
  return((alpha/2) * exp((-alpha)*  abs(x)))
}

# Define bound M efficiently
M <- sqrt(2/pi) * exp(0.5)

# Sample from U([0,1])
vecU1 <- runif(nSamples)

# Sample from the Laplace distibution using the inversion-method
alpha <- 1
vecU2 <- runif(nSamples)
vecLaplace <- rep(0, nSamples)
vecLaplace[vecU2 < 0.5] <- (1/alpha) * log(2*vecU2[vecU2 < 0.5])
vecLaplace[vecU2 >= 0.5] <- (-1/alpha) * log(2*(1-vecU2[vecU2 >= 0.5]))

# Accept or reject?
vecZ3 <- vecLaplace[vecU1 <= dnorm(vecLaplace)/(M * g(vecLaplace))]

# Acceptance ratio
fAcceptRatio <- length(vecZ3) / nSamples

# Test for normality
fnCheckNormality(vecZ3)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  vecSample
## D = 0.015448, p-value = 0.3289
## alternative hypothesis: two-sided
##
##
##  Shapiro-Wilk normality test
##
## data:  vecSample
## W = 0.99968, p-value = 0.8549
```

Die Akzeptanzrate liegt bei 0.7546. D.h. es müssen im Schnitt etwa vier Zufallszahlen bzgl. g_1 erzeugt werden um drei standard-normalverteilte Zufallszahlen zu erhalten.

Wir sehen wiederum, dass Histogramm und Dichtefunktion sowie die theoretischen und empirischen Quantile sehr gut übereinstimmen. Auch die Tests können zum Signifikanzniveau $\alpha = 0.05$ oder $\alpha = 0.01$ nicht abgelehnt werden, da die p-Werte größer sind.

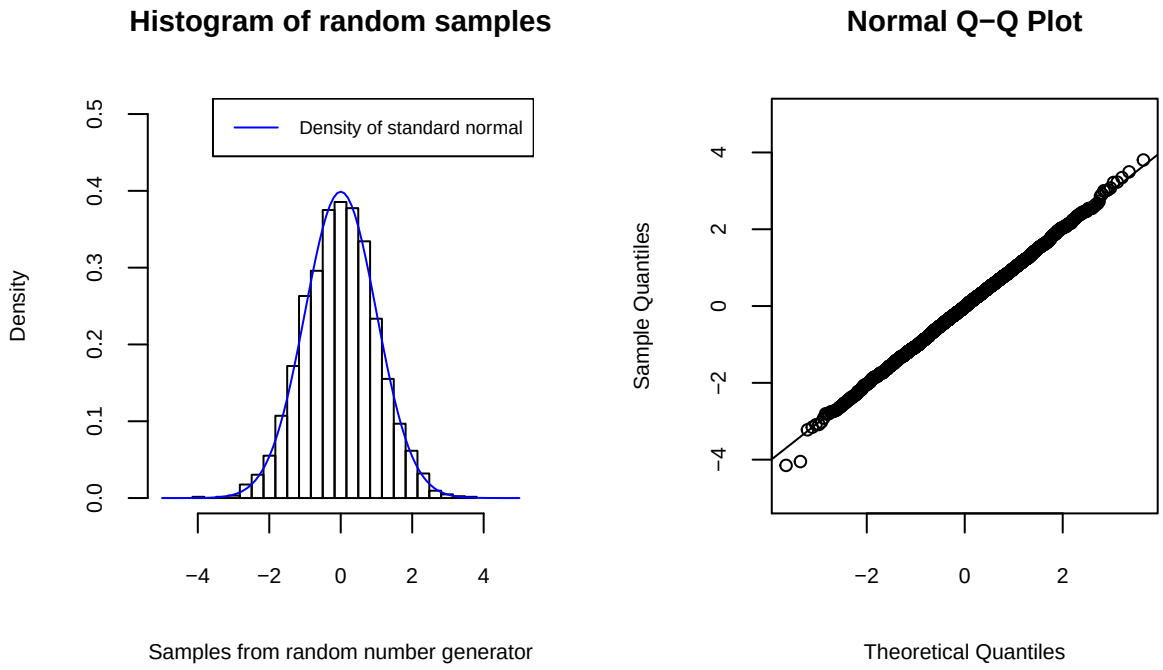

Figure 7: Histogram and QQ-Norm-plot for random numbers generated using Accept-Reject-Sampling with Laplace proposal density with $\alpha = 1$.