

# Booklets

Computational Statistics / Computerintensive Methoden

**3. Mai 2025**

Name: Safouan Er-Ryfy

---

## Inhaltsverzeichnis

<b>1. Verbreitung von Krankheiten</b>	<b>3</b>
<b>A1. SIR Modell</b>	<b>3</b>
a. Der Verlauf der Verbreitung simulieren . . . . .	3
b. Hygienevorschriften . . . . .	4
c. Kontakseinschränkung . . . . .	4
d. Kombination aus HygienemaSSnahmen und Kontaktbeschränkungen . . . . .	6
e. Zeit variieren . . . . .	7
<b>A2. SIS Modell</b>	<b>8</b>
a. SIR und SIS vergleichen . . . . .	8
b. Mögliche Erweiterung: Virusvarianten . . . . .	9
c. Weitere Herausforderungen: Impfungen . . . . .	9
<b>2. Monte Carlo Integration</b>	<b>10</b>
<b>A1. Monte Carlo Approximation von Kreiszahl</b>	<b>10</b>
<b>A2. Monte Carlo Integration</b>	<b>12</b>
a. n gleichverteilte Zufallszahlen . . . . .	12
b. Die Verteilung der Schätzer . . . . .	13
<b>A3. Berechnen einer Fläche</b>	<b>14</b>
a. Graphische Darstellung der Fläche A . . . . .	14
b. Naherungsweise der Flächeninhalt . . . . .	15
<b>3. Simulation von Zufallszahlen</b>	<b>16</b>
<b>A1. Inversions- und Box-Müller-Methode</b>	<b>16</b>
a. Exponentialverteilung und Poissonverteilung . . . . .	16
b. Normalverteilung . . . . .	17
<b>A2. Zufallszahlengeneratoren</b>	<b>18</b>
a. Random Decimal Fraction von Random.org . . . . .	18
b. Mittsquadratverfahren . . . . .	19
c. Kreativer Generator . . . . .	20
d. Mersenne-Twister und Super-Duper . . . . .	21
e. Vergleich der Generatoren . . . . .	21

## 1. Verbreitung von Krankheiten

### A1. SIR Modell

Aufgabestellung:

Wir verwenden SIR (Susceptible, Infectious, Recover) Modell als *stochastic individual contact model (ICM)*.

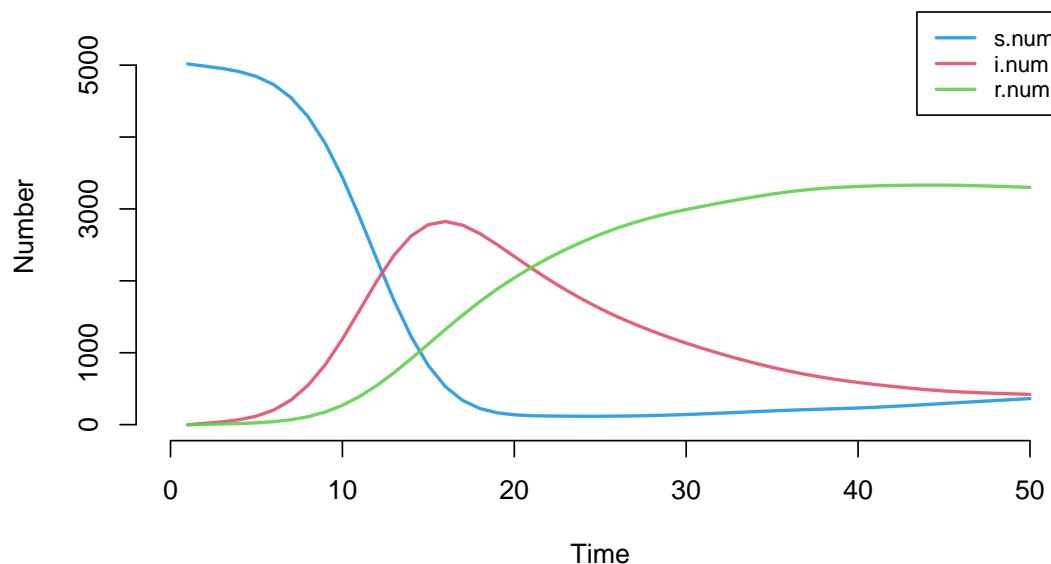
#### a. Der Verlauf der Verbreitung simulieren

Simulieren Sie den Verlauf der Verbreitung in 50 Tagen und stellen Sie die Anzahl von infizierbaren, infizierten und immunisierten Personen in einer geeigneten Graphik dar.

```
param <- param.icm(inf.prob = 0.1,
                     act.rate = 10,
                     rec.rate = 1/14,
                     a.rate = 1/100,
                     ds.rate = 1/90,
                     di.rate = 2/90,
                     dr.rate = 1/90)

init <- init.icm(s.num = 5000, i.num = 10, r.num = 0)
control <- control.icm(type = "SIR", nsteps = 50)

model1 <- icm(param, init, control)
plot(model1)
```



## b. Hygienevorschriften

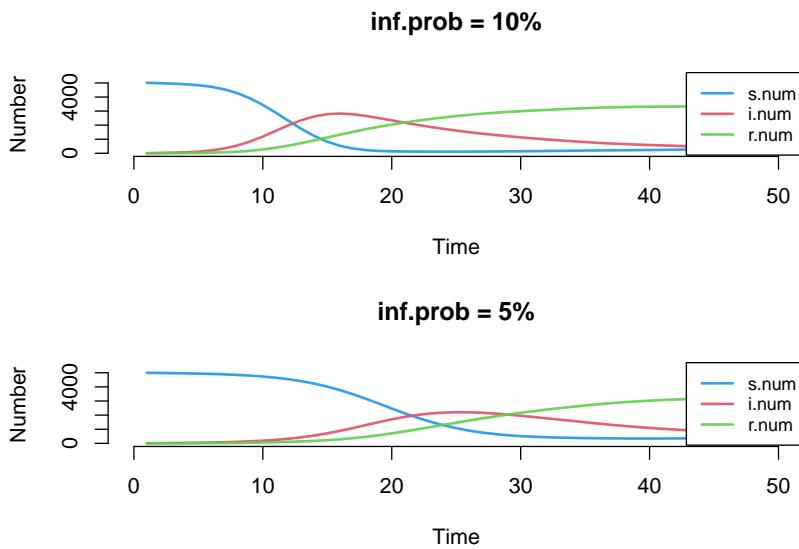
Gehen Sie davon aus, dass zum Zeitpunkt  $t_H = 0$  Hygienevorschriften erlassen wurden (z.B. Händewaschen, Mundschutz etc.). Dies beeinflusst die Ansteckungswahrscheinlichkeit (*inf.prob*) und senkt sie auf 5%. Simulieren Sie die neue Situation und vergleichen Sie diese mit Ihren Ergebnissen aus a).

```
# Die Infektionswahrscheinlichkeit (inf.prob) wird von 0.1 auf 0.05 reduziert.
param <- param.icm(inf.prob = 0.05,
                     act.rate = 10,
                     rec.rate = 1/14,
                     a.rate = 1/90,
                     ds.rate = 1/90,
                     di.rate = 2/90,
                     dr.rate = 1/90)

init <- init.icm(s.num = 5000, i.num = 10, r.num = 0)
control <- control.icm(type = "SIR", nsteps = 50)

model2 <- icm(param, init, control)

par(mfrow = c(2, 1), mar = c(4, 4, 4, 4))
plot(model1, main = "inf.prob = 10%")
plot(model2, main = "inf.prob = 5%")
```



Eine höhere Infektionswahrscheinlichkeit (*inf.prob* = 10%) führt zu einem schnelleren und stärkeren Anstieg der Infektionen. Bei niedriger Wahrscheinlichkeit (5%) verläuft die Ausbreitung deutlich flacher und langsamer.

## c. Kontakseinschränkung

Gehen Sie nun davon aus, dass die Population keine Hygienemaßnahmen ergreift, aber zum Zeitpunkt  $t_K = 0$  den Kontakt einschränkt. D.h. hier wird die Begegnungsrate gesenkt (auf 5 bzw. 2). Vergleichen Sie Ihre Ergebnisse mit a) und b) und interpretieren diese.

```

param1 <- param.icm(inf.prob = 0.1,
                      act.rate = 5,
                      rec.rate = 1/14,
                      a.rate = 1/90,
                      ds.rate = 1/90,
                      di.rate = 2/90,
                      dr.rate = 1/90)

param2 <- param.icm(inf.prob = 0.1,
                      act.rate = 2,
                      rec.rate = 1/14,
                      a.rate = 1/90,
                      ds.rate = 1/90,
                      di.rate = 2/90,
                      dr.rate = 1/90)

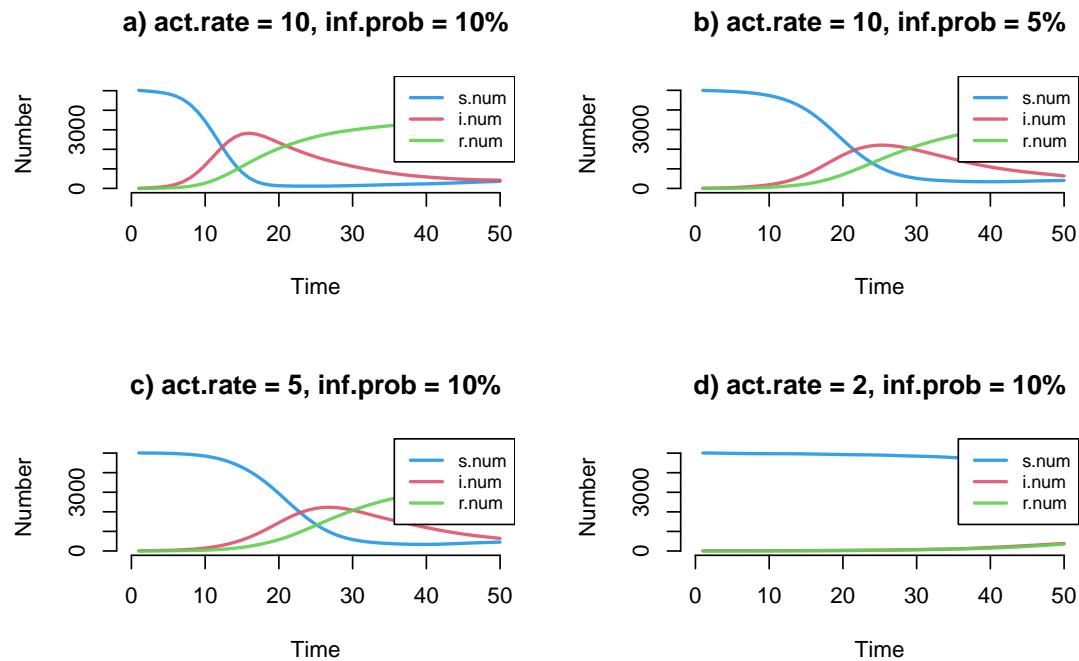
init <- init.icm(s.num = 5000, i.num = 10, r.num = 0)
control <- control.icm(type = "SIR", nsteps = 50)

model_c1 <- icm(param1, init, control)
model_c2 <- icm(param2, init, control)

par(mfrow = c(2, 2))

plot(model1 , main = "a) act.rate = 10, inf.prob = 10%")
plot(model2 , main = "b) act.rate = 10, inf.prob = 5%")
plot(model_c1, main = "c) act.rate = 5, inf.prob = 10%")
plot(model_c2, main = "d) act.rate = 2, inf.prob = 10%")

```



- Hygienemaßnahmen und Kontaktbeschränkungen wirken sich jeweils auf die Ansteckungs-

wahrscheinlichkeit (`inf.prob`) und die Begegnungsrate (`act.rate`) aus. Die Abbildungen b) und c) zeigen: Eine Halbierung der `act.rate` hat einen ähnlichen Effekt wie eine Halbierung der `inf.prob`. Wird `act.rate` sogar auf 2 gesenkt (d)), flacht der Verlauf der Infektion deutlich ab.

## d. Kombination aus HygienemaSSnahmen und Kontaktbeschränkungen

Kombinieren Sie die MaSSnahmen aus b) und c) und vergleichen auch diesen Effekt mit Ihren vorherigen Befunden.

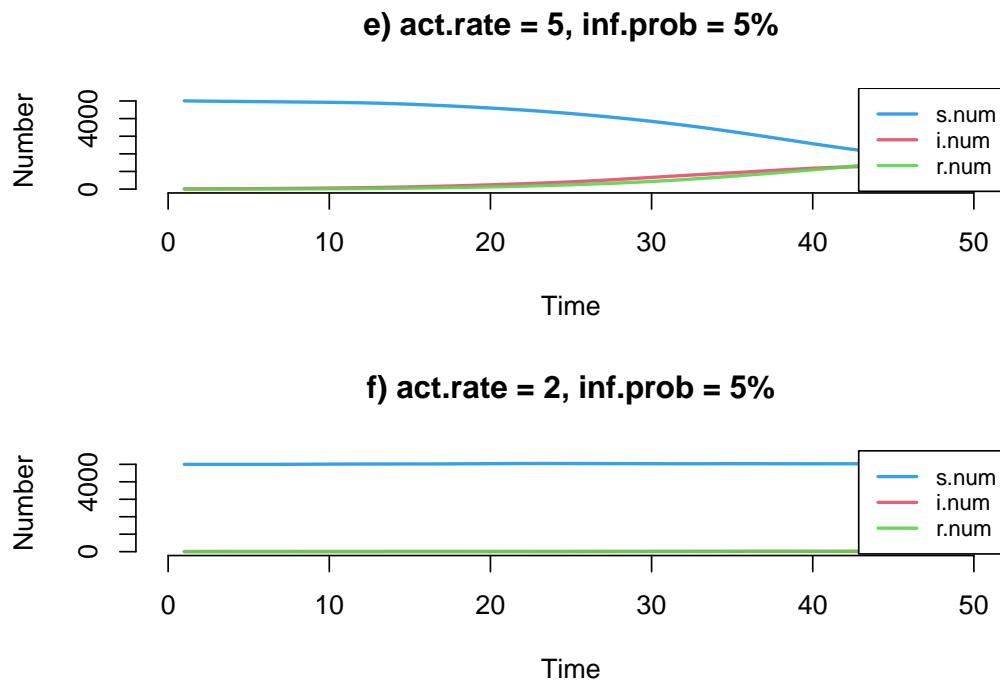
```
param3 <- param.icm(inf.prob = 0.05,
                      act.rate = 5,
                      rec.rate = 1/14,
                      a.rate = 1/90,
                      ds.rate = 1/90,
                      di.rate = 2/90,
                      dr.rate = 1/90)

param4 <- param.icm(inf.prob = 0.05,
                      act.rate = 2,
                      rec.rate = 1/14,
                      a.rate = 1/90,
                      ds.rate = 1/90,
                      di.rate = 2/90,
                      dr.rate = 1/90)

init <- init.icm(s.num = 5000, i.num = 10, r.num = 0)
control <- control.icm(type = "SIR", nsteps = 50)

model_d3 <- icm(param3, init, control)
model_d4 <- icm(param4, init, control)

par(mfrow = c(2, 1), mar = c(4, 4, 4, 4))
plot(model_d3, main = "e") act.rate = 5, inf.prob = 5%
plot(model_d4, main = "f") act.rate = 2, inf.prob = 5%
```



- Die Kombination aus HygienemaSSnahmen und Kontaktbeschränkungen wirkt besonders effektiv – gemeinsam flachen sie die Infektionskurve deutlich stärker ab.

### e. Zeit variieren

Variieren Sie die Zeitpunkte  $t_H$  und  $t_K$  und setzen den MaSSnahmen gegebenenfalls auch ein Ende. Vergleichen Sie auch jetzt die Verläufe und interpretieren diese.

- Ab dem Zeitpunkt  $t_H = 10$  wurde die Ansteckungswahrscheinlichkeit (`inf.prob`) von 10% auf 5% gesenkt. Ab  $t_K = 20$  kam zusätzlich eine Reduktion der Kontaktfrequenz (`act.rate`) von 10 auf 2 hinzu. Beide MaSSnahmen zusammen zeigen eine deutlich stärkere Wirkung.

## A2. SIS Modell

### a. SIR und SIS vergleichen

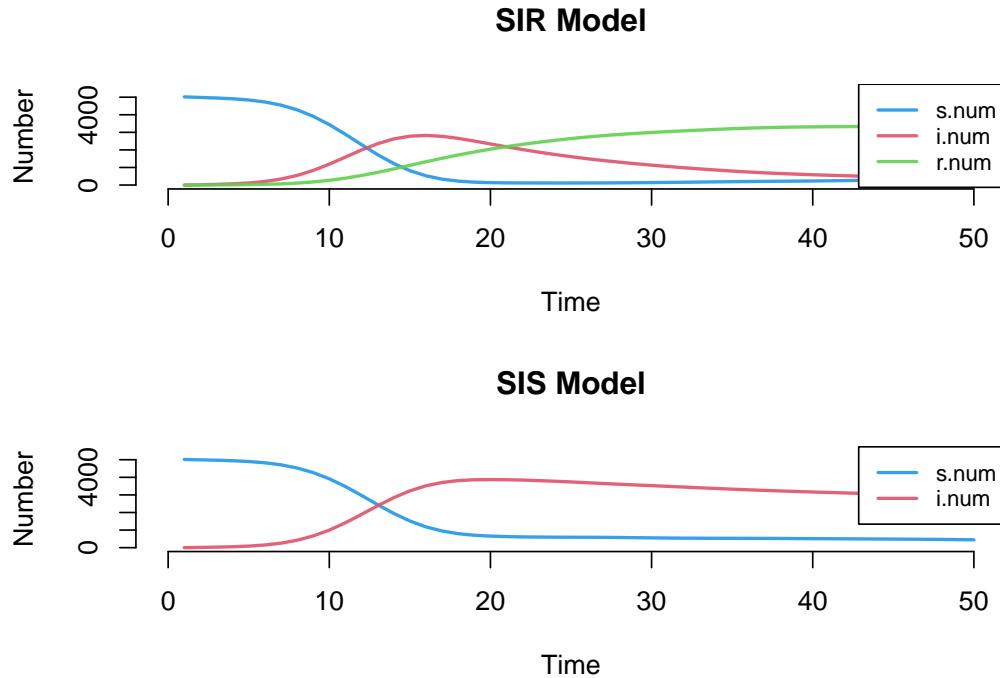
Inzwischen weiß man, dass sich Menschen nach einer gewissen Zeit erneut infizieren können.

- Was bedeutet das für unser Modell? Es gibt keine dauerhafte Immunität nach einer Genesung. Die Gruppe der Genesenen“ verliert damit an Bedeutung – stattdessen kehren Personen nach ihrer Infektion wieder in die Gruppe der Anfälligen zurück. In diesem Szenario erreicht die Anzahl der Infizierten ein stabiles Gleichgewicht, bei dem sich Neuinfektionen und Genesungen die Waage halten.
- Was wäre hier besser? Ein SIS-Modell ist für diesen Fall besser geeignet als das klassische SIR-Modell, da es genau diesen Rückfluss von Genesenen zu Anfälligen abbildet.
- Setzen Sie das neue Modell (auch als stochastic individual contact model (ICM)) in einer Simulation um, ’spielen’ mit den entsprechenden Parametern, stellen die Ergebnisse in geeigneten Graphiken dar und interpretieren Ihre Befunde.

```
param <- param.icm(inf.prob = 0.1,
                     act.rate = 10,
                     rec.rate = 1/14,
                     a.rate = 1/90,
                     ds.rate = 1/90,
                     di.rate = 2/90,
                     dr.rate = 1/90)

init <- init.icm(s.num = 5000, i.num = 10, r.num = 0)
control <- control.icm(type = "SIS", nsteps = 50)

par(mfrow = c(2, 1), mar = c(4, 4, 4, 4))
sis_model <- icm(param, init, control)
plot(model1, main="SIR Model")
plot(sis_model, main="SIS Model")
```



### b. Mögliche Erweiterung: Virusvarianten

Im Verlauf der Pandemie sind viele verschiedene Virusvarianten aufgetreten.

- Was bedeuten diese Varianten für die Modellierung? Jede Variante hat eigene Eigenschaften wie Ansteckungsrate oder Krankheitsverlauf. Daher müsste man für jede Variante eigene Parameter definieren.
- Wie könnte man das modellieren – und wo liegen die Schwierigkeiten? Man könnte mehrere Varianten im Modell abbilden, z.B. mit separaten Zuständen oder Parametern.
- Probleme dabei:
  - Eine Person kann sich mit mehreren Varianten anstecken (Ko-Infektion).
  - Es ist schwierig, realistische Übertragungsraten für jede Variante zu bestimmen.
  - Die Modellierung wird deutlich komplexer, weil mehr Parameter und Zustände nötig sind.

### c. Weitere Herausforderungen: Impfungen

- Wie kann man Impfungen berücksichtigen? Die Infizierbaren (S) könnten in zwei Gruppen unterteilt werden: Geimpfte und Ungeimpfte – mit unterschiedlichen Ansteckungsraten.
- Welche Parameter beeinflussen Impfungen? Besonders relevant sind `inf.prob` (Ansteckungswahrscheinlichkeit) und `rec.rate` (Erholungsrate)
- Sind diese Einflüsse konstant? Nein, sie können variieren, z.B. je nach Alter, Immunsystem, Vorerkrankungen oder Zeitpunkt der Impfung.

## 2. Monte Carlo Integration

### A1. Monte Carlo Approximation von Kreiszahl

In dieser Übung soll die Kreiszahl approximiert werden.

Erzeuge hierzu  $n$  Tupel  $(x_i, y_i)$ ,  $i = 1, \dots, n$  mit

$$x_i, y_i \sim U([1, 1])$$

und definiere daraus die Variable

$$z_i = 4 \cdot \mathbf{1}(x_i^2 + y_i^2 \leq 1).$$

Hierbei ist  $\mathbf{1}$  die Indikatorfunktion, die 1 liefert, wenn die Bedingung erfüllt ist, und 0 sonst.

Approximiert wird durch

$$= \frac{1}{n} \sum_{i=1}^n z_i.$$

- Erzeuge eine Tabelle, in der für  $n = 10^k$ ,  $k \in \{1, \dots, 8\}$  jeweils die Approximation und der absolute Fehler  $| - |$  dargestellt werden.
- Stimmt die Genauigkeit mit der Tschebyscheff-Schätzung aus dem Video überein?

In dieser Aufgabe schätzen wir die Kreiszahl durch zufällig gezogene Punkte innerhalb des Quadrats  $[1, 1]^2$ .

Die Wahrscheinlichkeit, dass ein Punkt innerhalb des Einheitskreises liegt, entspricht dem Verhältnis der Flächen:

$$P(x^2 + y^2 \leq 1) = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

Da  $z_i = 4 \cdot \mathbf{1}(x_i^2 + y_i^2 \leq 1)$ , ist dies eine Bernoulli-Kette mit Erwartungswert:

$$E[z_i] = 4 \cdot \frac{\pi}{4} =$$

Somit ist der Stichprobenmittelwert

$$= \frac{1}{n} \sum_{i=1}^n z_i$$

eine Schätzung von .

```
set.seed(123) # Seed setzen für Reproduzierbarkeit
k = 8
alpha <- 0.05 # 95% Sicherheit

df <- data.frame(
  k = 1:k,
  n = 10^(1:k)
)

df$mean <- sapply(df$n, function(n) {
```

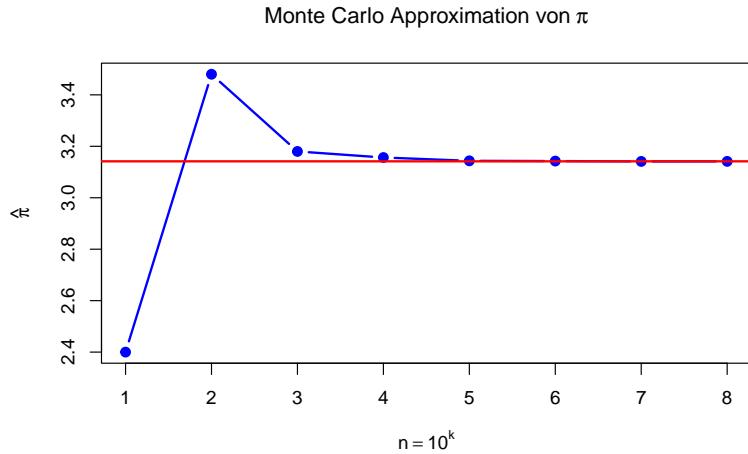
```

x <- runif(n, -1, 1) # n gleichverteilte Zufallszahlen zwischen -1 und 1 erzeugen
y <- runif(n, -1, 1)
mean(4 * (x^2 + y^2 <= 1))
})

df$var <- sapply(df$n, function(n) {
  x <- runif(n, -1, 1)
  y <- runif(n, -1, 1)
  var(4 * (x^2 + y^2 <= 1))
})

df$error <- sqrt(df$var / (alpha * df$n)) # Fehlerabschätzung basierend auf Tschebyscheff-Ungleichung

# Plot
plot(df$k, df$mean, type = "b", lwd = 2, pch = 19, col = "blue",
      xlab = expression(n == 10^k),
      ylab = expression(hat(pi)),
      main = expression("Monte Carlo Approximation von " ~ pi))
abline(h = pi, col = "red", lwd = 2)
    
```



```

df
##   k     n      mean      var      error
## 1 1 1e+01 2.400000 1.600000 1.7888543820
## 2 2 1e+02 3.480000 3.258182 0.8072399666
## 3 3 1e+03 3.180000 2.619644 0.2288948948
## 4 4 1e+04 3.156400 2.698845 0.0734689764
## 5 5 1e+05 3.143560 2.687353 0.0231834103
## 6 6 1e+06 3.142612 2.698049 0.0073458133
## 7 7 1e+07 3.141214 2.698435 0.0023231165
## 8 8 1e+08 3.141456 2.696273 0.0007343396
    
```

- Interpretation: Die Genauigkeit der Monte Carlo-Approximation von  $\pi$  steigt mit wachsender Stichprobengröße  $n$ , während die Varianz der Schätzungen entsprechend sinkt (Gesetz der grossen Zahlen).

## A2. Monte Carlo Integration

In dieser Übung soll der Wert 1.96 als 0.975 *Quantil* der Standardnormalverteilung mit Hilfe von Monte Carlo Integration überprüft werden.

### a. n gleichverteilte Zufallszahlen

Erzeuge hierzu  $n$  gleichverteilte Zufallszahlen auf dem Intervall [1,96, 1,96] und bestimme

$$2 \leq 1,96 \leq (x),$$

wobei die Dichte der Standardnormalverteilung ist, definiert durch

$$(x) = \frac{1}{2} \exp\left(-\frac{x^2}{2}\right).$$

Stelle analog zur Aufgabe A1 für verschiedene Werte von  $n$  die Approximation und ihre Genauigkeit in einer Tabelle dar. Verwende hierfür die gleichen Werte für  $n$  wie in Aufgabe A1, also  $n = 10^k$  für  $k \in \{1, , 8\}$ .

**Hinweis:** Der gesuchte Integralwert ist 0,95!

```
set.seed(123) # Seed setzen für Reproduzierbarkeit
k = 8
alpha <- 0.05
Q <- 1.96 # 0.975-Quantil der Standardnormalverteilung

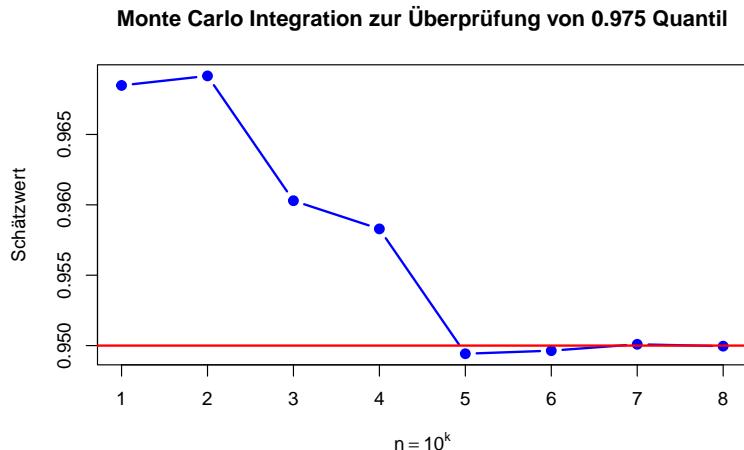
df <- data.frame(
  k = 1:k,
  n = 10^(1:k)
)

df$mean <- sapply(df$n, function(n) {
  x <- runif(n, -Q, Q) # n gleichverteilte Zufallsvariablen zwischen -Q und Q
  mean(2 * Q * dnorm(x))
})

df$var <- sapply(df$n, function(n) {
  x <- runif(n, -Q, Q)
  var(2 * Q * dnorm(x))
})

df$error <- sqrt(df$var / (alpha * df$n)) # Fehlerabschätzung basierend auf Tschebyscheff-Ungleichung

# Plot
plot(df$k, df$mean, type = "b", lwd = 2, pch = 19, col = "blue",
      xlab = expression(n == 10^k),
      ylab = "Schätzwert",
      main = "Monte Carlo Integration zur Überprüfung von 0.975 Quantil")
abline(h = 0.95, col = "red", lwd = 2)
```



```
df
##   k      n      mean      var      error
## 1 1 1e+01 0.9684838 0.2404018 0.6934000096
## 2 2 1e+02 0.9691602 0.1752135 0.1871969795
## 3 3 1e+03 0.9602923 0.1940901 0.0623041169
## 4 4 1e+04 0.9582903 0.1983122 0.0199154330
## 5 5 1e+05 0.9494218 0.1965286 0.0062694272
## 6 6 1e+06 0.9496388 0.1971466 0.0019856818
## 7 7 1e+07 0.9500912 0.1971818 0.0006279837
## 8 8 1e+08 0.9499596 0.1971159 0.0001985527
```

- Interpretation: Mit zunehmender Stichprobengröße  $n$  wird die Monte Carlo Schätzung des 0.975 Quantils der Standardnormalverteilung zunehmend genauer und stabilisiert sich sehr nah am theoretischen Wert von 0.95 (Gesetz der großen Zahlen).

## b. Die Verteilung der Schätzer

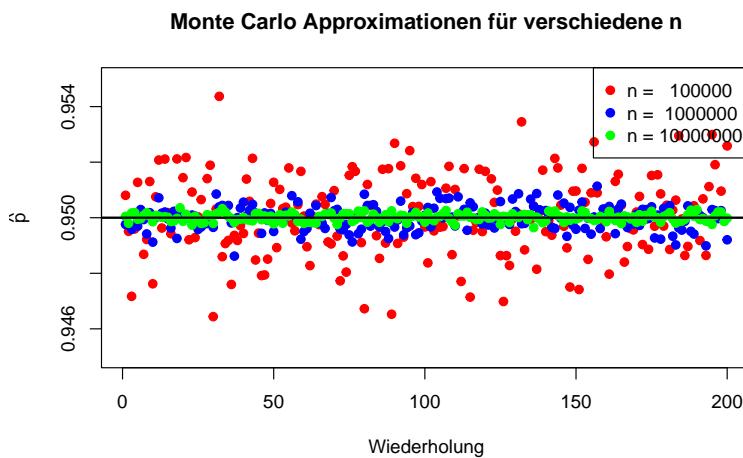
Führe für die Werte  $n = 10^5$ ,  $n = 10^6$  und  $n = 10^7$  jeweils 200 Approximationen durch und stelle die Verteilung der Schätzer für  $p = 0.95$  in einer gemeinsamen Abbildung dar. Interpretiere das Ergebnis.

```
set.seed(123) # Seed setzen für Reproduzierbarkeit

N <- 200 # Anzahl der Wiederholungen
n_values <- c(10^5, 10^6, 10^7)
Q <- 1.96
y_lim <- c(0.945, 0.955)
farben <- c("red", "blue", "green")

results <- list()
for (n in n_values) {
  z_mean <- replicate(N, {
    x <- runif(n, -Q, Q) # n gleichverteilte Zufallsvariablen zwischen -Q und Q
    mean(2 * Q * dnorm(x))
  })
  results[[as.character(n)]] <- z_mean
}
```

```
# Plot
plot(1:N, results[[1]], col = farben[1], pch = 19, ylim = y_lim,
      xlab = "Wiederholung", ylab = expression(hat(p)),
      main = "Monte Carlo Approximationen für verschiedene n")
for (i in 2:length(n_values)) {
  points(1:N, results[[i]], col = farben[i], pch = 19)
}
abline(h = 0.95, col = "black", lwd = 2)
legend('topright', legend = paste("n =", format(n_values, scientific = FALSE)),
       col = farben, pch = 19)
```



- Interpretation: Mit zunehmender Stichprobengröße  $n$  verringert sich die Varianz der Schätzungen signifikant. Bei  $n = 10^7$  (grün) ist die Approximation extrem präzise und praktisch perfekt auf 0,95 zentriert. Die grafische Darstellung bestätigt eindrucksvoll die Konvergenzeigenschaften der Monte Carlo Integration und das Gesetz der grossen Zahlen.

## A3. Berechnen einer Fläche

In dieser Übung soll eine Fläche mithilfe von Monte-Carlo-Integration berechnet werden. Als umgebendes Rechteck wählen wir  $[7, 7] \times [4, 4]$ , die Fläche  $A$  ist durch folgende Ungleichung definiert:

$$\frac{x^2}{49} + \frac{y^2}{9} \leq 1$$

Erzeuge  $n = 10^5$  Tupel  $(x_i, y_i)$  für  $i = 1, \dots, n$  mit  $x_i \sim U([7, 7])$  und  $y_i \sim U([4, 4])$  und bilde hieraus die Bernoulli-Variable  $z_i = \mathbf{1}_A(x_i, y_i)$ .

### a. Graphische Darstellung der Fläche A

Stelle die Fläche A anhand der Punkte, die in A liegen, graphisch dar. Wenn die Ungleichungen richtig umgesetzt wurde, sollte Dir die Fläche bekannt vorkommen.

```
set.seed(123) # Seed setzen für Reproduzierbarkeit
n <- 10^5
x <- runif(n, min = -7, max = 7)
```

```

y <- runif(n, min = -4, max = 4)

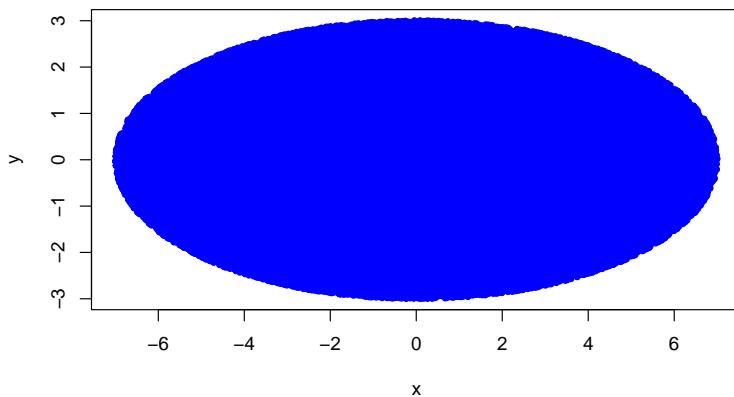
# Berechne die Funktion f(x, y) für jedes Tupel
f <- x^2 / 49 + y^2 / 9 - 1

# Bestimme die Punkte, die in A liegen
indices_A <- which(f <= 0)
x_A <- x[indices_A]
y_A <- y[indices_A]

# Plot der Punkte in A
plot(x_A, y_A, pch = 20, col = "blue",
      main = "Darstellung der Fläche A",
      xlab = "x",
      ylab = "y")

```

Darstellung der Fläche A

**b. Näherungsweise der Flächeninhalt**

Berechne näherungsweise den Flächeninhalt von A mit Hilfe der Formel aus dem Video.

$$\text{Fläche der Ellipse } (7 \times 4) \times \frac{\text{Anzahl der Punkte in } A}{n}$$

```

A <- (7*4) * (length(indices_A)/n)
print(paste("Der Flächeninhalt A beträgt näherungsweise:", A))
## [1] "Der Flächeninhalt A beträgt näherungsweise: 16.51384"

```

### 3. Simulation von Zufallszahlen

#### A1. Inversions- und Box-Müller-Methode

In dieser Aufgabe sollen aus gleichverteilten Zufallszahlen Zufallszahlen beliebiger Verteilung gewonnen werden.

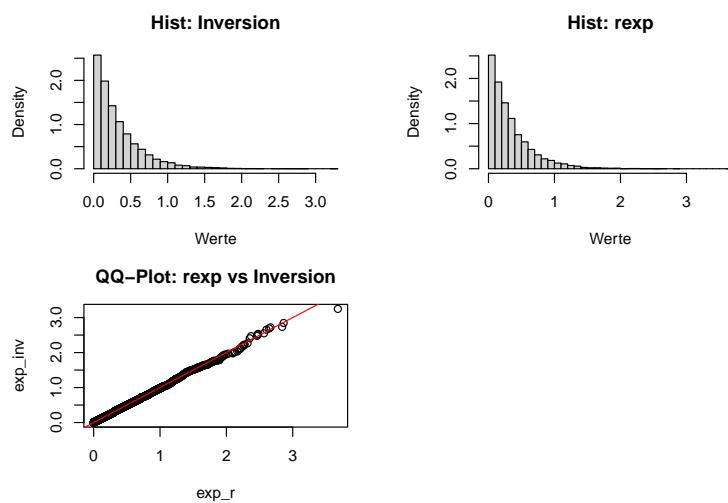
##### a. Exponentialverteilung und Poissonverteilung

Erzeugen Sie mit Hilfe der Inversionsmethode aus jeweils  $n = 10000$  gleichverteilten Zufallszahlen zwischen 0 und 1,  $n = 10000$  exponentialverteilte bzw. poissonverteilte Zufallszahlen mit Parameter = 3. Vergleichen Sie die Verteilungen mit den Verteilungen entsprechender Zufallszahlen, die direkt von R in der gewünschten Verteilung ausgegeben werden (Histogramm, QQ-Plot).

```
set.seed(123) # Seed setzen für Reproduzierbarkeit
n <- 10000
lambda <- 3
u <- runif(n, 0, 1)
exp_inv <- ((-1/lambda) * log(1 - u))
exp_r <- rexp(n, rate = lambda)

par(mfrow = c(2, 2), mar = c(4, 4, 3, 3))
hist(exp_inv, breaks = 30, probability = TRUE, main = "Hist: Inversion", xlab = "Werte")
hist(exp_r, breaks = 30, probability = TRUE, main = "Hist: rexp", xlab = "Werte")

# QQ-Plot: Vergleich von 2 Datensätzen
qqplot(exp_r, exp_inv, main = "QQ-Plot: rexp vs Inversion")
abline(0, 1, col = "red") # Diagonaöe
```



Die erste Darstellung belegt, dass die Inversionsmethode korrekt exponentiell verteilte Zufallszahlen erzeugt.

```
poisson_inver <- function(n, lambda) { # Produkt von Zufallszahlen (Knuths Algorithmus)
  x <- numeric(n)
  for (i in 1:n) {
    L <- exp(-lambda)
    p <- 1
    k <- 0
```

```

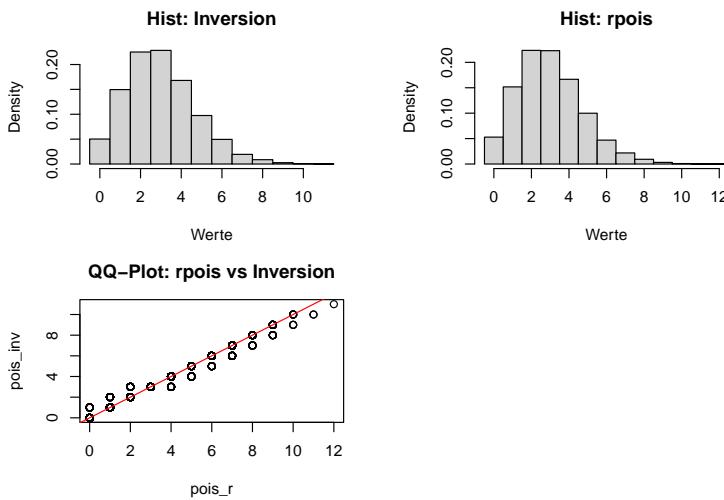
while (p > L) {
  k <- k + 1
  p <- p * runif(1)
}
x[i] <- k - 1
}
return(x)
}

pois_inv <- poisson_inver(n, lambda)
pois_r <- rpois(n, lambda)

par(mfrow = c(2, 2), mar = c(4, 4, 3, 3))
hist(pois_inv, breaks = seq(-0.5, max(pois_inv)+0.5, 1), probability = TRUE, main = "Hist: Inversion")
hist(pois_r, breaks = seq(-0.5, max(pois_r)+0.5, 1), probability = TRUE, main = "Hist: rpois", xlab = "Werte")

# QQ-Plot: Vergleich von 2 Datensätzen
qqplot(pois_r, pois_inv, main = "QQ-Plot: rpois vs Inversion")
abline(0, 1, col = "red") # Diagonale

```



Die zweite Darstellung belegt, dass die Inversionsmethode eine gute Annäherung an die echte Poissonverteilung liefert.

## b. Normalverteilung

Da sich die Verteilung der Normalverteilung nicht ohne Weiteres umkehren lässt, ist die Inversionsmethode aus a) zur Erzeugung normalverteilter Zufallszahlen aus gleichverteilten Zufallszahlen ungeeignet. Verwenden Sie die Box-Müller Methode: Erzeugen Sie  $n = 10000$  zweier Tupel  $(u_i, v_i), i = 1, \dots, n$ , aus gleichverteilten Zufallszahlen. Durch die Transformationsformel

$$z_i = \cos(2u_i) \ln(v_i)$$

lassen sich nun 10000 normalverteilte Zufallszahlen erzeugen. Vergleichen Sie auch hier die Verteilung mit den entsprechenden Zufallszahlen aus R (Histogramm, QQ-Plot).

```

u <- runif(n)
v <- runif(n)

```

```

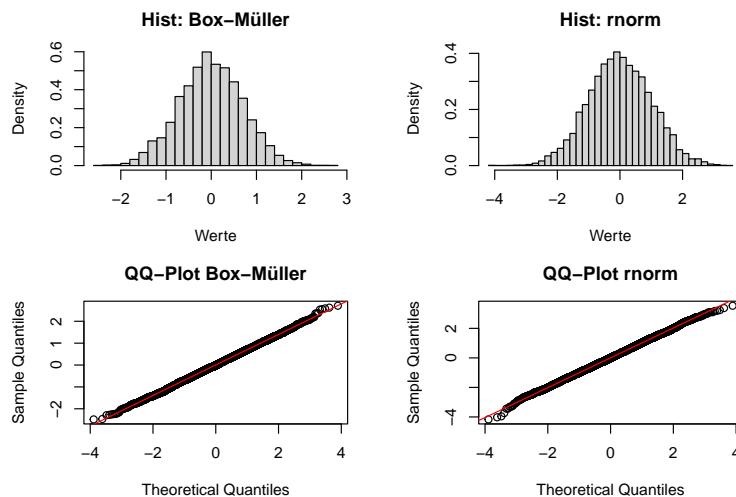
z_boxmuller <- cos(2 * pi * u) * sqrt(-log(v))
z_r <- rnorm(n)

par(mfrow = c(2, 2), mar = c(4, 4, 3, 3))
hist(z_boxmuller, breaks = 30, probability = TRUE, main = "Hist: Box-Müller", xlab = "Werte")
hist(z_r, breaks = 30, probability = TRUE, main = "Hist: rnorm", xlab = "Werte")

# QQ-Plots Normalverteilung
qqnorm(z_boxmuller, main = "QQ-Plot Box-Müller")
qqline(z_boxmuller, col = "red")

qqnorm(z_r, main = "QQ-Plot rnorm")
qqline(z_r, col = "red")

```



Die Inversionsmethode und die Box-Müller-Methode liefern Ergebnisse, die sehr gut mit den direkten R-Funktionen vergleichbar sind. Die Histogramme und QQ-Plots bestätigen die Übereinstimmung der erzeugten Verteilungen.

## A2. Zufallszahlengeneratoren

In dieser Aufgabe sollen gleichverteilte Zufallszahlen auf unterschiedliche Arten erzeugt werden. Zur Überprüfung der Qualität bilden wir jeweils zweier Tupel  $(z_1, z_2), (z_2, z_3), \dots, (z_n, z_n)$  und stellen diese in der zweidimensionalen Ebene dar. **Erkennbare Muster sprechen für eine schlechte Qualität der Zufallszahlen.**

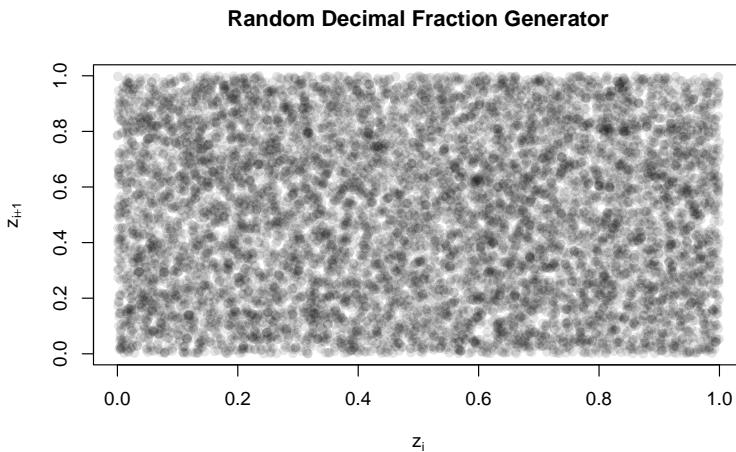
### a. Random Decimal Fraction von Random.org

Lassen Sie sich auf der Seite [Random.org](http://Random.org) 10000 echte gleichverteilte Zufallszahlen aus atmosphärischem Rauschen erzeugen (Random Decimal Fraction Generator)

```

# echte gleichverteilte Zufallszahlen aus atmosphärischem Rauschen
random_org <- scan("random_org.txt")
plot(head(random_org, -1), tail(random_org, -1), # Paare von direkt aufeinanderfolgenden Zufallszahlen
      main = "Random Decimal Fraction Generator", xlab = expression(z[i]), ylab = expression(z[i+1]),
      pch = 19, col = rgb(0, 0, 0, 0.1))

```



## b. Mittquareatverfahren

Erzeugen Sie jeweils 10000 Zufallszahlen mit dem Mittquareat Verfahren von Neumann mit  $k = 8$  mittleren Ziffern.

```

mid_square <- function(n, k = 8) {
  x <- numeric(n)
  seed <- 62459921 # 8-stelliger Startwert
  for (i in 1:n) {
    square <- seed^2
    square_str <- as.character(square)

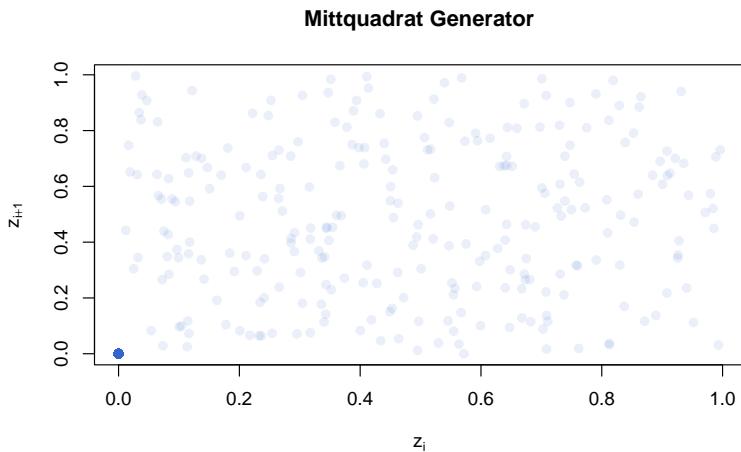
    # Auffüllen mit Nullen, falls square zu kurz ist
    while (nchar(square_str) < 2 * k) {
      square_str <- paste0("0", square_str)
    }

    # mittlere k Ziffern extrahieren
    start <- floor((nchar(square_str) - k) / 2) + 1
    mid <- substr(square_str, start, start + k - 1)

    seed <- as.numeric(mid)
    x[i] <- seed / 10^k # Skaliere auf [0, 1]
  }
  return(x)
}

msq <- mid_square(10000)
plot(head(msq, -1), tail(msq, -1),
  main = "Mittquareat Generator", xlab = expression(z[i]), ylab = expression(z[i+1]),
  pch = 19, col = c(0.2, 0.4, 0.8, 0.1))

```



### c. Kreativer Generator

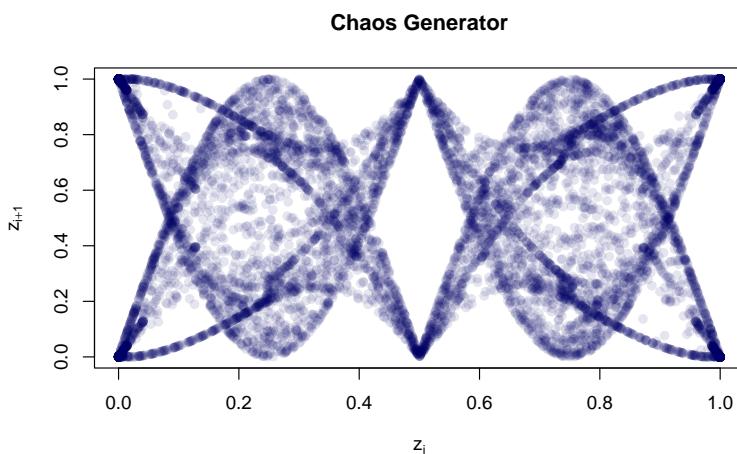
Werden Sie kreativ: Kombinieren Sie die einfachen Generatoren aus der Vorlesung zu einem eigenen Zufallszahlgenerator bzw. denken Sie sich etwas völlig anderes aus - der Generator muss nicht perfekt sein! Geben Sie Ihrem Generator einen liebevollen Namen und erzeugen 10000 Zufallszahlen

```

chaos <- function(n) {
  x <- numeric(n)
  x[1] <- 0.3546 # Startwert
  for (i in 2:n) {
    x[i] <- (sin(x[i - 1] * 2 * pi) * cos(i / 20) + 2) %% 1
  }
  return(x)
}

chaos <- chaos(10000)
plot(chaos[-10000], chaos[-1],
      main = "Chaos Generator", xlab = expression(z[i]), ylab = expression(z[i+1]),
      pch = 19, col = rgb(0, 0, 0.4, 0.1))

```

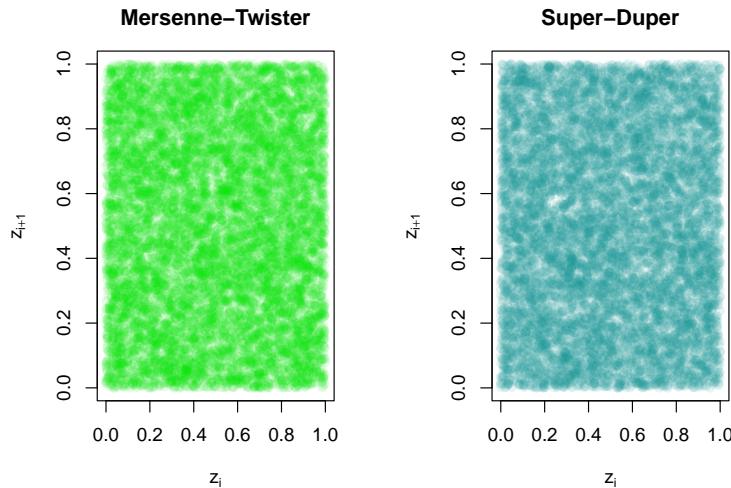


## d. Mersenne-Twister und Super-Duper

Erzeugen Sie mit Hilfe des Mersenne-Twisters (Standard in R) und mit Hilfe eines beliebigen weiteren Generators in R (Funktion *RNGkind*) jeweils 10000 Zufallszahlen.

```
mersenne <- runif(10000) # Standard
RNGkind("Super-Duper")
superduper <- runif(10000)

par(mfrow = c(1, 2), mar = c(4, 4, 3, 3))
plot(head(mersenne, -1), tail(mersenne, -1),
     main = "Mersenne-Twister", xlab = expression(z[i]), ylab = expression(z[i+1]),
     pch = 19, col = rgb(0, 0.9, 0, 0.1))
plot(head(superduper, -1), tail(superduper, -1),
     main = "Super-Duper", xlab = expression(z[i]), ylab = expression(z[i+1]),
     pch = 19, col = rgb(0.1, 0.6, 0.6, 0.1))
```



## e. Vergleich der Generatoren

Vergleichen Sie die erzeugten Zufallszahlen aus a)-d) mit Hilfe entsprechender Plots.

- Random Decimal Fraction: Sehr gute Gleichverteilung, keine Muster sichtbar.
- Mittsquadratverfahren: Geringe Dichte und Häufungen im Plot, Zeichen für mangelnde Zufälligkeit.
- Chaos: Erkennbares Muster, nicht als Zufallszahlengenerator geeignet.
- Mersenne-Twister: Sehr gute Gleichverteilung, keine Muster sichtbar.
- Super-Duper: Sehr gute Gleichverteilung, keine Muster sichtbar.