Stochastik-Praktikum, WS 2023/23

Woche 18.-22.12.2023

Themen:

- Illustrationen des zentralen Grenzwertsatzes
- eine „Pfad-Version“ des zentralen Grenzwertsatzes und die Brownsche Bewegung

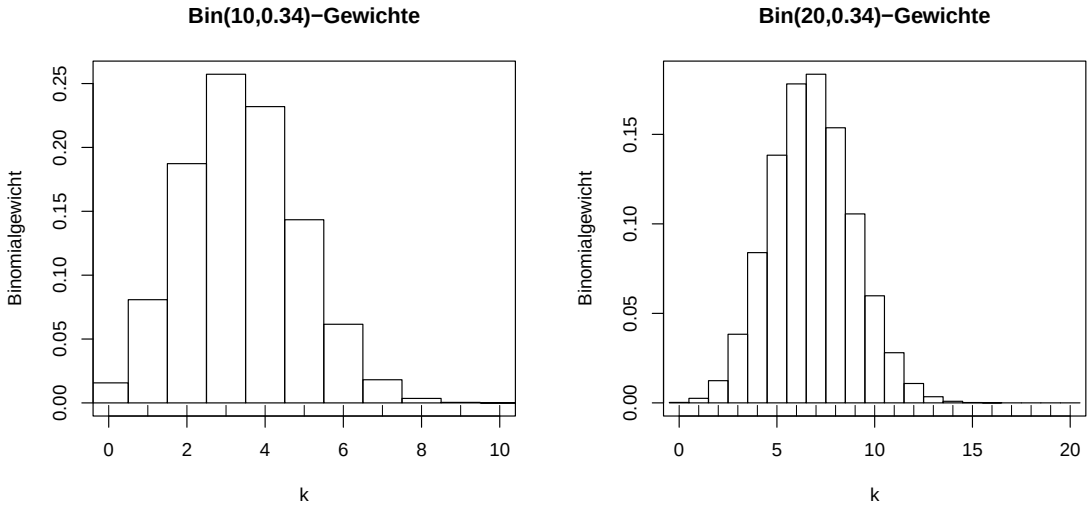
1 Illustrationen des zentralen Grenzwertsatzes

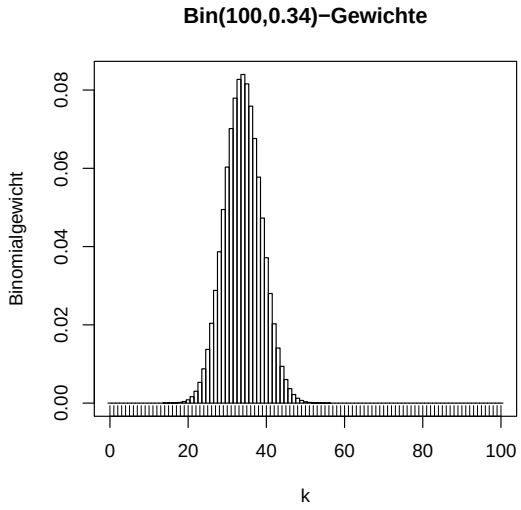
In der Vorlesung lernen wir den zentralen Grenzwertsatz kennen, siehe Kapitel 5 der Vorlesungsnotizen. Hier illustrieren wir das dort als mathematische Grenzwertaussage beschriebene Phänomen, dass Summen vieler zufälliger, unabhängiger Summanden (unter geeigneten Bedingungen) approximativ normalverteilt sind, für konkrete Beispiele (z.T. via Simulation).

Spezialfall: Illustration des Satzes von de Moivre und Laplace Wir betrachten zunächst die Approximation von der Gewichte Binomialverteilungen $\mathrm{Bin}_{n,p}$ durch die Normalverteilung, den Satz von de Moivre und Laplace (vgl. Satz 5.1 der Vorlesungsnotizen):

```
zeichne.binomial.hist <- function(n, p) {
# zeichnet die Gewichte von Bin(n,p)
# als Balkendiagramm in einen Plot
  binomialgewichte <- dbinom(0:n, size=n, prob=p)
  k.werte <- (0:n)
  plot(k.werte, binomialgewichte, type="n",
       xlab="k", ylab="Binomialgewicht",
       main=paste("Bin(",n,",",p,")-Gewichte",sep=""))
  rect(k.werte-0.5, 0, k.werte+0.5, binomialgewichte)
  rug(k.werte)
}
```

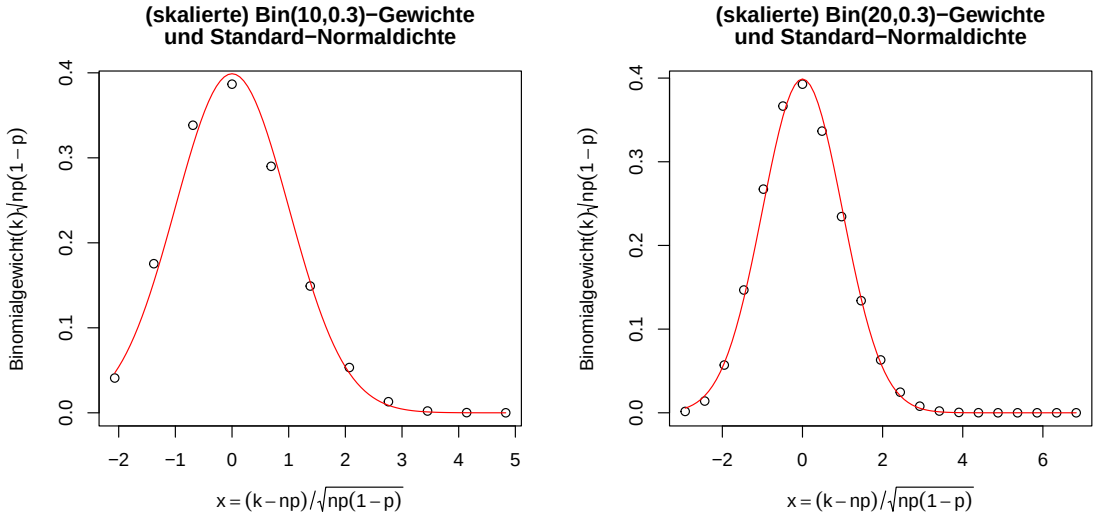
```
p <- 0.34  # variieren Sie gerne das p

zeichne.binomial.hist(10,p)
zeichne.binomial.hist(20,p)
zeichne.binomial.hist(100,p)
```

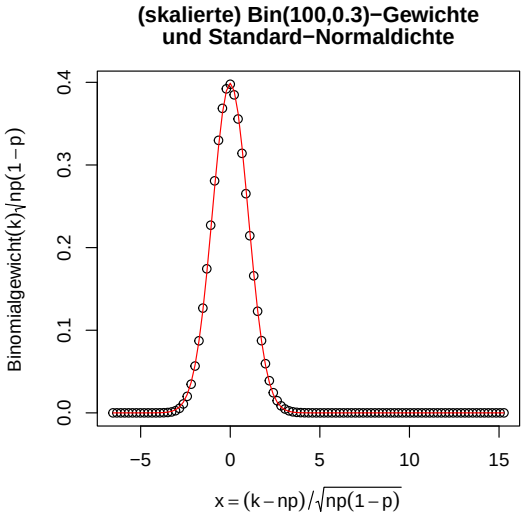




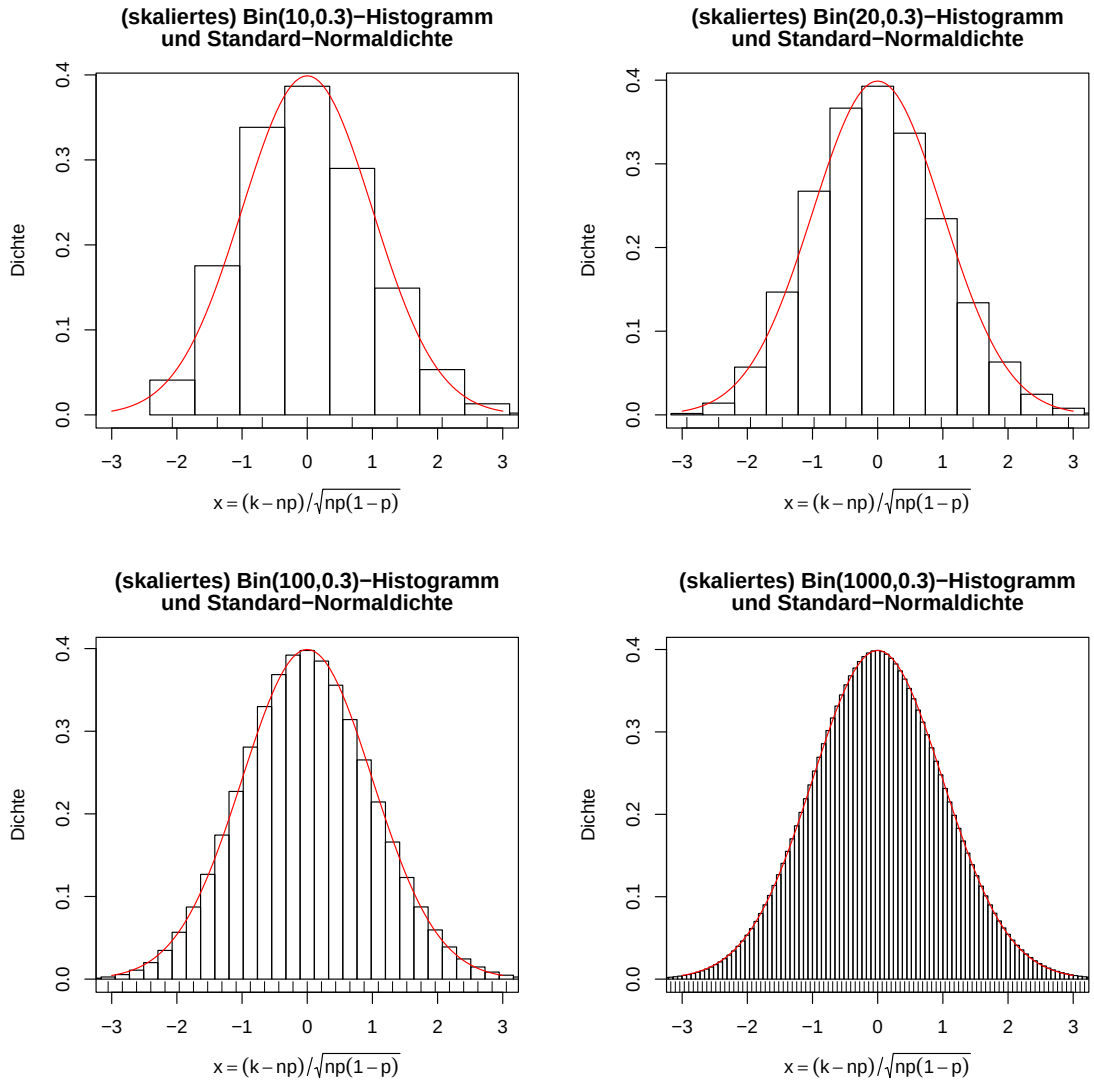

```
# (Wir sehen: die Form stabilisiert sich, Lage und Höhe nicht,
#  daher skalieren wir um und betrachten die Verteilung von
#  (X-n*p)/sqrt(n*p*(1-p)). )

# Lokale Normalapproximation der Binomialverteilung,
# Konvergenz der (skalierten) Gewichte

# X Bin(n,p)-verteilt, für n, k gegen unendlich
# (so dass |k-n*p| <= C*sqrt(n)) konvergiert
# sqrt(n*p*(1-p))*P(X=k) gegen die Standard-Normaldichte an der
# Stelle (k-n*p)/sqrt(n*p*(1-p))

illustriere.ldML <- function(n, p) {
  # zeichne skalierte Binomialgewichte und Normaldichte
  x.werte.skaliert <- ((0:n)-n*p)/sqrt(n*p*(1-p))
```

```r
  binomialgewichte <- dbinom(0:n, size=n, prob=p)

  plot(x.werte.skaliert, sqrt(n*p*(1-p))*binomialgewichte,
       main=c(paste("(skalierte) Bin(",n,",",p,")-Gewichte",sep=""),
              "und Standard-Normaldichte"),
       xlab=expression(x==(k-n*p)/sqrt(n*p*(1-p))),  ## Bem.: Math. Formel in Beschriftung
       ylab=expression(Binomialgewicht(k)*sqrt(n*p*(1-p))))

  curve(dnorm, col="red", add=TRUE)
}


n <- 10; p <- 0.3 # gerne variieren

illustriere.ldML(n,p)
illustriere.ldML(20,p)
illustriere.ldML(100,p)
```





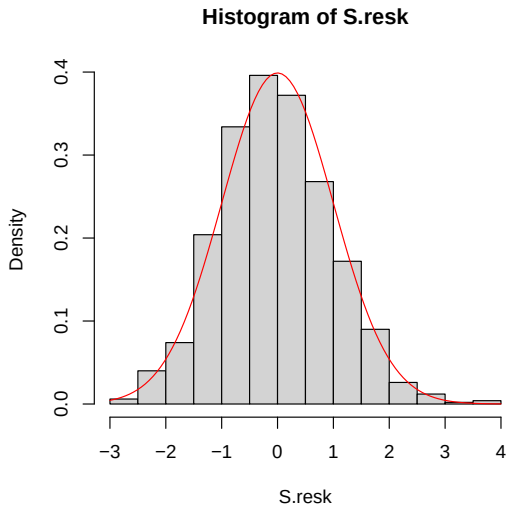

```r
# Integrale Normalapproximation der Binomialverteilung,
# Konvergenz von (Intervall-)Wahrscheinlichkeiten

illustriere.dML.hist <- function(n, p) {
  # zeichne skaliertes Binomialhistogramm und Normaldichte
  x.werte.skaliert <- ((0:n)-n*p)/sqrt(n*p*(1-p))
  binomialgewichte <- dbinom(0:n, size=n, prob=p)

  plot(x.werte.skaliert, sqrt(n*p*(1-p))*binomialgewichte, type="n",
       xlim=c(-3,3),
       main=c(paste("(skaliertes) Bin(",n,",",p,")-Histogramm",sep=""),
              "und Standard-Normaldichte"),
       xlab=expression(x==(k-n*p)/sqrt(n*p*(1-p))),
       ylab="Dichte")

  rect(((0:n)-n*p-0.5)/sqrt(n*p*(1-p)), 0,
       ((0:n)-n*p+0.5)/sqrt(n*p*(1-p)), sqrt(n*p*(1-p))*binomialgewichte)

  rug(x.werte.skaliert)

  curve(dnorm, col="red", add=TRUE)
}

n <- 10; p <- 0.3 # gerne variieren

illustriere.dML.hist(n,p)
illustriere.dML.hist(20,p)
illustriere.dML.hist(100,p)
illustriere.dML.hist(1000,p)
```



Zentraler Grenzwertsatz (allgemein)

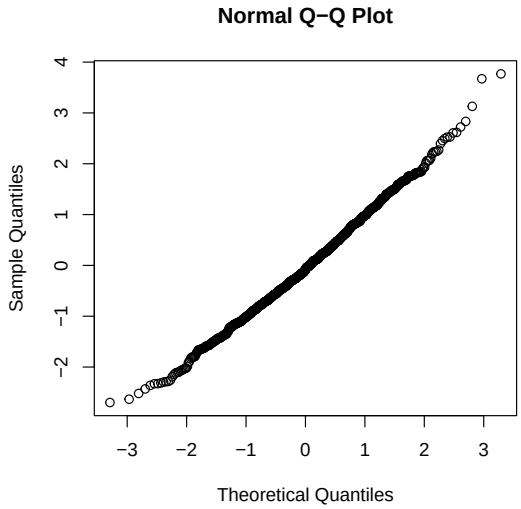
```
# X[1], X[2], ... u.i.v. mit Erwartungswert mu und
# endlicher Varianz sigma^2, so ist
# (S[n]-n*mu)/(sigma*sqrt(n))
# ungefähr (standard-)normalverteilt,
# wo S[n]:=X[1]+...+X[n]

# Bsp.: Exponentialverteilte Summanden:
n <- 100 # variieren: n <- 500;  n <- 1000
lambda <- 2.0
mu <- 1/lambda
sigma <- 1/lambda

versuche <- 1000
```

```
S.beob <- replicate(versuche, sum(rexp(n, rate=lambda)))
S.resk <- (S.beob-n*mu)/(sigma*sqrt(n))

hist(S.resk, prob=TRUE)
curve(dnorm, col="red", add=TRUE)
```



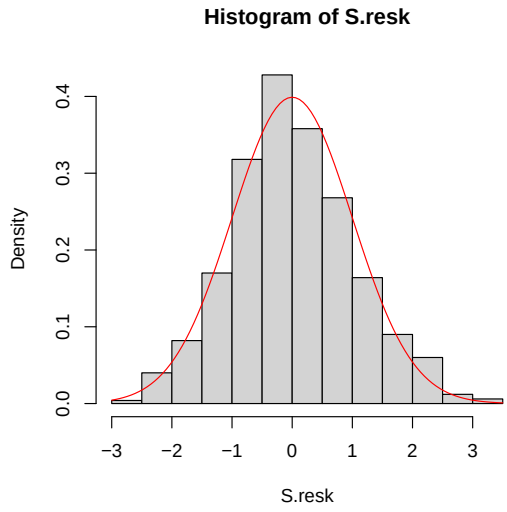
Der Befehl folgende `qqnorm` erstellt einen sogenannten Q-Q-Plot (Quantil-Quantil-Plot, siehe auch die Online-Hilfe z.B. via `?qqnorm`), ein graphisches Diagnosewerkzeug um einzuschätzen, ob Beobachtungsdaten plausiblerweise aus einer Normalverteilung stammen. In diesem Fall sollten die Punkte im Q-Q-Plot (ungefähr) auf einer Geraden liegen:

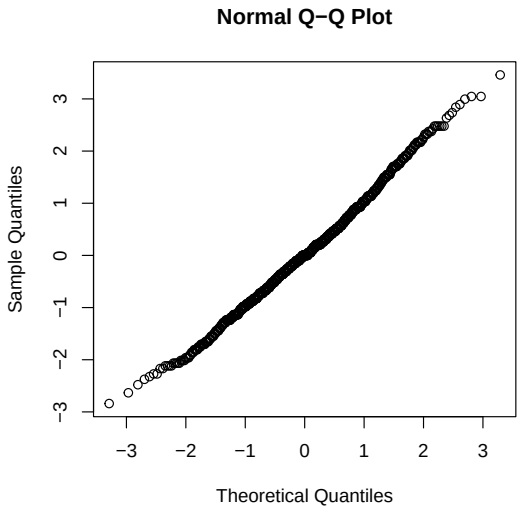
```
qqnorm(S.resk)
```



(Wir sehen: das passt hier recht gut.)

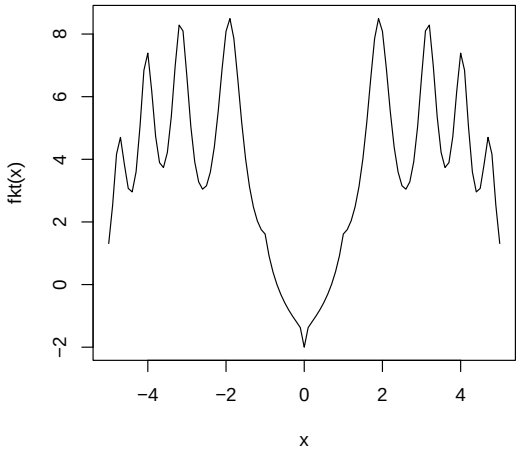
```r
# Bsp.: Geometrisch verteilte Summanden:
# P(X=k)=p*(1-p)^k, k=0,1,2,... : ersetze oben
geom.par <- 0.4
mu <- 1/geom.par - 1
sigma <- sqrt(1-geom.par)/geom.par
S.beob <- replicate(versuche, sum(rgeom(n, prob=geom.par)))

S.resk <- (S.beob-n*mu)/(sigma*sqrt(n))

hist(S.resk, prob=TRUE)
curve(dnorm, col="red", add=TRUE)

qqnorm(S.resk)
```

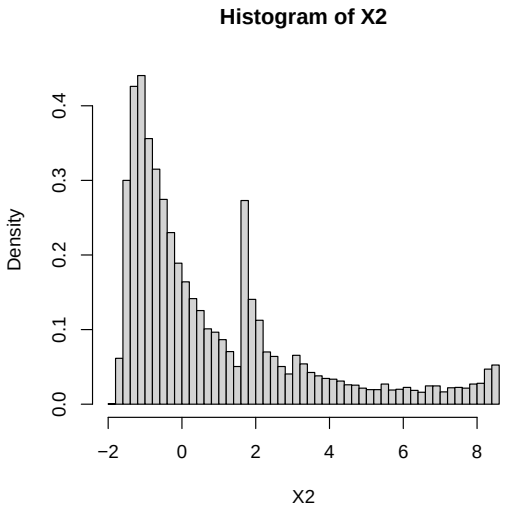


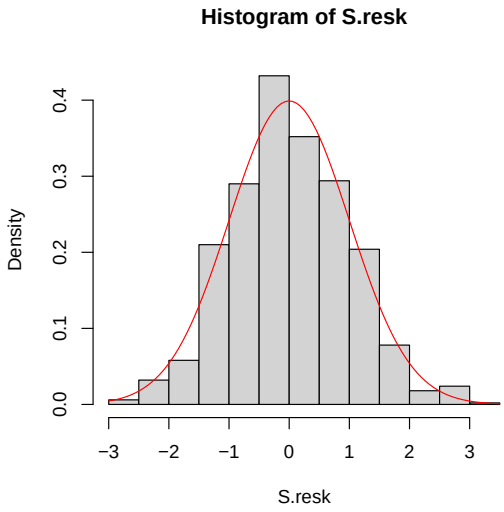


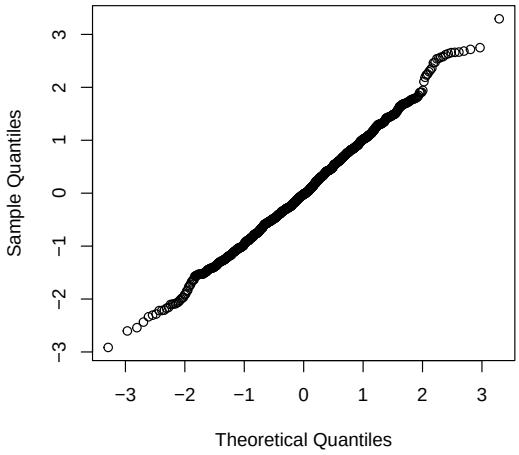

```r
#
# Man kann "nahezu beliebig" verteilte Summanden haben:
# Bsp:

fkt <- function(x) (4+3*x^2-0.1*x^4)/(1+sin(1+x^2)+abs(log(x^2)))-2
# (diese Fkt. ggf. variieren)

curve(fkt, xlim=c(-5,5))

X1 <- rnorm(10000)
X2 <- sapply(X1, fkt)

hist(X2, prob=TRUE, breaks=40)

# Sei X=fkt(Z), wo Z std-normal
# wir schätzen den Mittelwert
X2 <- sapply(rnorm(100000), fkt)
mu <- mean(X2)
sigma <- sd(X2)
```

```r
mu; sigma

## [1] 0.8762133
## [1] 2.486954

n <- 100 # variieren: n <- 500;  n <- 1000

versuche <- 1000
S.beob <- replicate(versuche, sum(sapply(rnorm(n), fkt)))
S.resk <- (S.beob-n*mu)/(sigma*sqrt(n))

hist(S.resk, prob=TRUE)
curve(dnorm, col="red", add=TRUE)

qqnorm(S.resk)
```





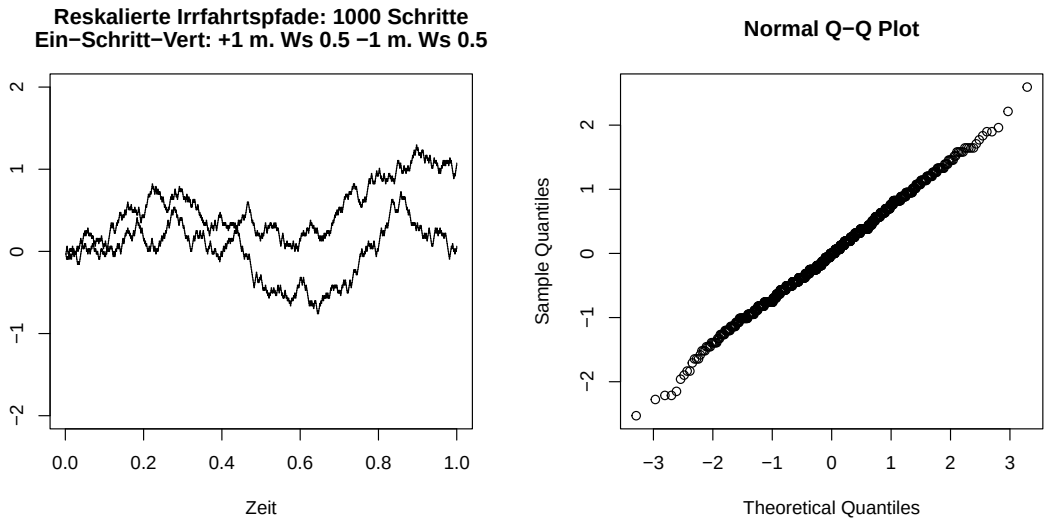


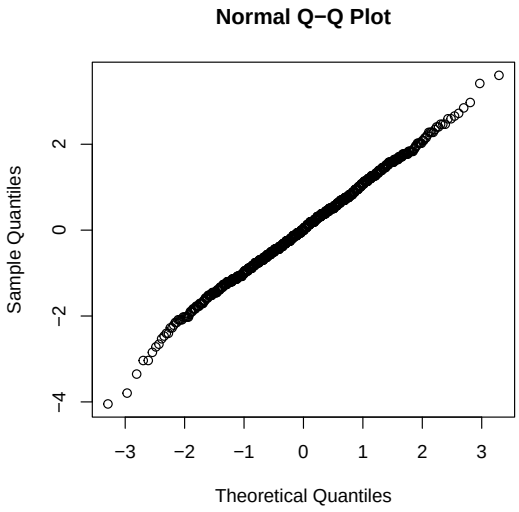


2 Von der Irrfahrt zur Brownschen Bewegung – zentraler Grenzwertsatz für zufällige Pfade

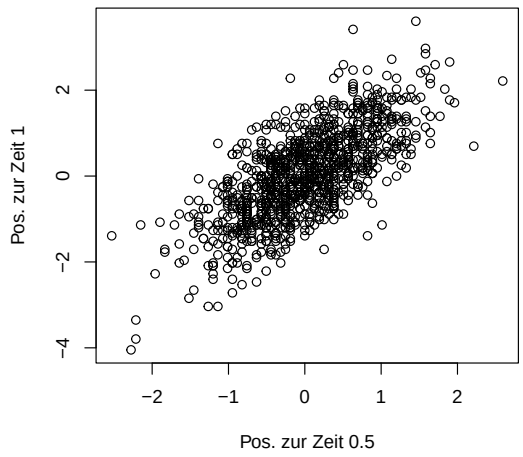
```
# (Verteilungs-)Konvergenz von Pfaden illustrieren
# (diverse ein-Schritt-Verteilungen)


# Bsp.: Summanden mit Werten in +/-1

N <- 1000 # variieren: N <- 100; N <- 500;  N <- 1000
p <- 0.5
mu <- p*1+(1-p)*(-1)
sigma <- sqrt(1-mu^2)

pfadsim <- function(n) (cumsum(2*rbinom(n, size=1, prob=p)-1)-(1:n)*mu)/(sigma*sqrt(n))

zeit <- (1:N)/N

pfad1 <- pfadsim(N)

plot(zeit, pfad1, ylim=c(-2,2), type="s", xlab="Zeit",ylab="",
     main=c(paste("Reskalierte Irrfahrtspfade:",N,"Schritte"),
            paste("Ein-Schritt-Vert: +1 m. Ws",p,"-1 m. Ws",1-p)))

# ggf. mehrere Pfade einzeichnen:
pfad2 <- pfadsim(N)
points(zeit, pfad2, type="s")

# Betrachten wir die gemeinsame Verteilung der Pfade zur
# (reskalierten) Zeit 0.5 und zur (reskalierten) Zeit 1:

wdh <- 1000
beob.gewIrrf <- matrix(0, nrow=wdh, ncol=2)
for (i in 1:wdh) {
  pfad <- pfadsim(N)
  beob.gewIrrf[i,]<-pfad[c(round(N/2),N)]
}

# Vert. zur (reskalierten) Zeit 0.5:
mean(beob.gewIrrf[,1]); var(beob.gewIrrf[,1])

## [1] 0.00170763
## [1] 0.5210421

qqnorm(beob.gewIrrf[,1])

# Vert. zur (reskalierten) Zeit 1:
mean(beob.gewIrrf[,2]); var(beob.gewIrrf[,2])

## [1] 0.03092708
## [1] 1.075339
```

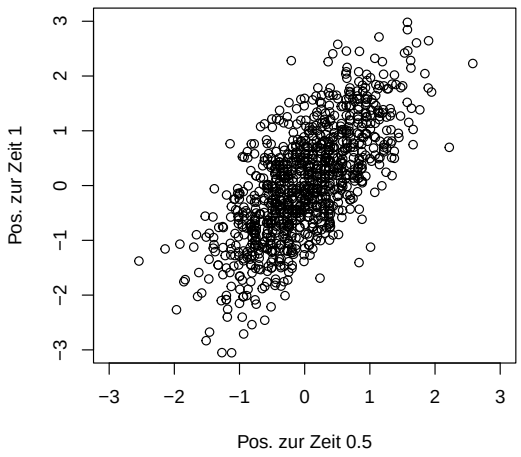
```
qqnorm(beob.gewIrrf[,2])
```



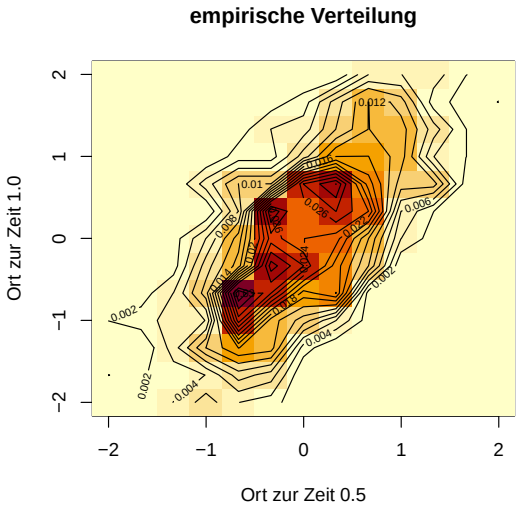



```
# gemeinsam:
plot(beob.gewIrrf[,1], beob.gewIrrf[,2], xlab="Pos. zur Zeit 0.5", ylab="Pos. zur Zeit 1")
# ggf. mit "Jitter":
jitterwert <- 0.02
plot(jitter(beob.gewIrrf[,1], amount=jitterwert),
     jitter(beob.gewIrrf[,2], amount=jitterwert),
     pty=21,
     xlab="Pos. zur Zeit 0.5", ylab="Pos. zur Zeit 1", xlim=c(-3,3), ylim=c(-3,3))

cov(beob.gewIrrf[,1], beob.gewIrrf[,2])

## [1] 0.5289601
```



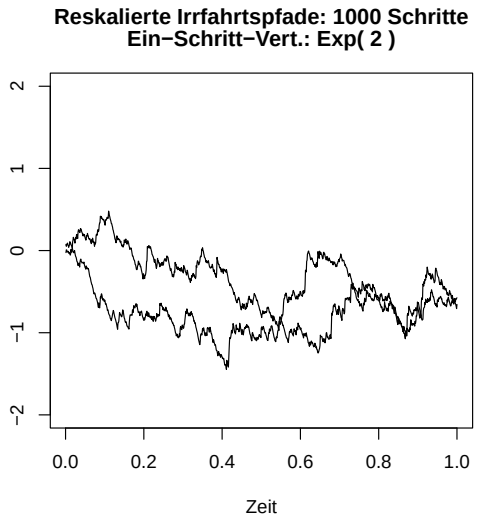



```r
# gemeinsam als image/contour
schrw <- 0.333 # schrw <- 0.1
x <- seq(from=-2, to=2, by=schrw)
y <- seq(from=-2, to=2, by=schrw)

pf2beob <- matrix(0, nrow=length(x), ncol=length(y))
for (i in 1:length(x)) {
  for (j in 1:length(y)) {
    pf2beob[i,j] <- sum( (beob.gewIrrf[,1]>=x[i]- schrw/2) & (beob.gewIrrf[,1]<x[i] + schrw/2)
                         & (beob.gewIrrf[,2]>=y[j] - schrw/2) & (beob.gewIrrf[,2] < y[j] + schrw/2))
    ## beachte: "&", nicht "&&"
  }
}

image(x,y,pf2beob/wdh, main="empirische Verteilung",
      xlab="Ort zur Zeit 0.5", ylab="Ort zur Zeit 1.0")
contour(x,y,pf2beob/wdh, add=TRUE, nlevels=20)
# Bem.: man muesste wdh (deutlich) erhöhen,
# um ein 'klareres' Bild zu erhalten, z.B.
wdh <- 1e5 ## (dann obigen Code nochmal ... was lange dauert ...)
```



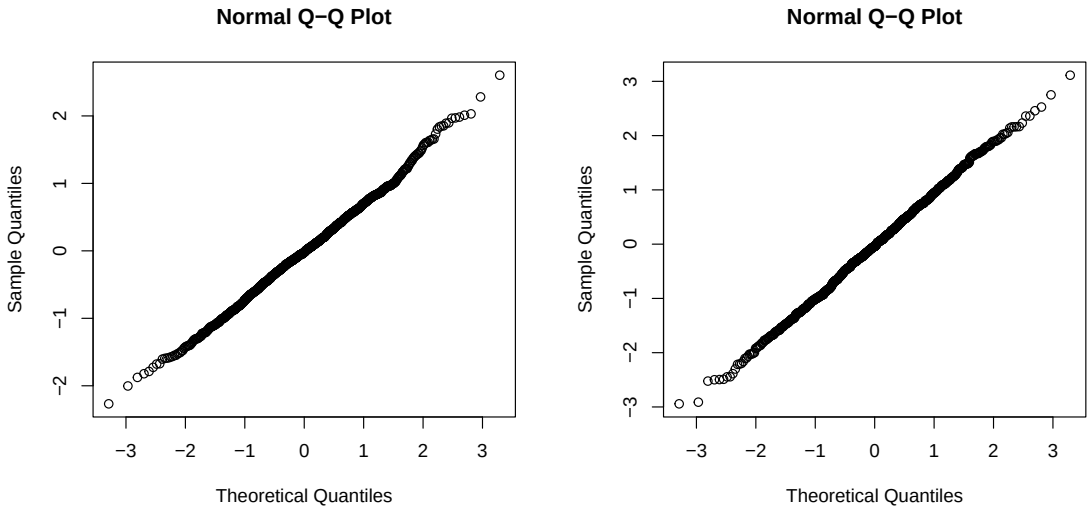

```
# Gemeinsame Verteilung:
# Welcher Anteil hat Pos. zur Zeit 0.5 <= x1 und
# Pos. zur Zeit 1.0 <= x2 ?
x1 <- 0.5; x2 <- -0.1
sum(beob.gewIrrf[,1] <= x1 & beob.gewIrrf[,2] <=x2)/wdh

## [1] 0.00438
```

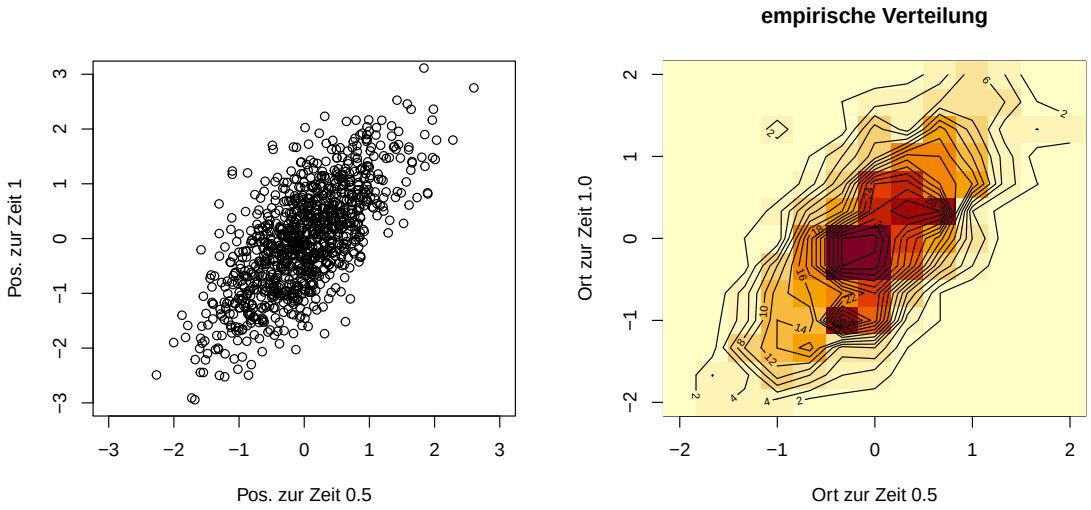
```
#  Bsp.: Exponentialverteilte Summanden:
N <- 1000 # variieren: N <- 500;  N <- 1000
lambda <- 2.0
mu <- 1/lambda
sigma <- 1/lambda

pfadsim <- function(n) (cumsum(rexp(n, rate=lambda))-(1:n)*mu)/(sigma*sqrt(n))

zeit <- (1:N)/N

pfad1 <- pfadsim(N)

plot(zeit, pfad1, ylim=c(-2,2), type="l", xlab="Zeit",ylab="",
     main=c(paste("Reskalierte Irrfahrtspfade:",N,"Schritte"),
            paste("Ein-Schritt-Vert.: Exp(",lambda,")")))

# ggf. mehrere Pfade einzeichnen:
pfad2 <- pfadsim(N)
points(zeit, pfad2, type="l")
```




```
# Betrachten wir die gemeinsame Verteilung der Pfade zur
# (reskalierten) Zeit 0.5 und zur (reskalierten) Zeit 1:

wdh <- 1000
beob.ExpIrrf <- matrix(0, nrow=wdh, ncol=2)
for (i in 1:wdh) {
  pfad <- pfadsim(N)
  beob.ExpIrrf[i,]<-pfad[c(round(N/2),N)]
}

# Vert. zur (reskalierten) Zeit 0.5:
mean(beob.ExpIrrf[,1]); var(beob.ExpIrrf[,1])

## [1] -0.008199208
## [1] 0.5146064

qqnorm(beob.ExpIrrf[,1])

# Vert. zur (reskalierten) Zeit 1:
mean(beob.ExpIrrf[,2]); var(beob.ExpIrrf[,2])

## [1] -0.02876585
## [1] 0.9297471

qqnorm(beob.ExpIrrf[,2])

# gemeinsam (als Scatterplot):
plot(beob.ExpIrrf[,1], beob.ExpIrrf[,2], xlab="Pos. zur Zeit 0.5",
     ylab="Pos. zur Zeit 1", xlim=c(-3,3),ylim=c(-3,3))

cov(beob.ExpIrrf[,1], beob.ExpIrrf[,2])

## [1] 0.482853
```

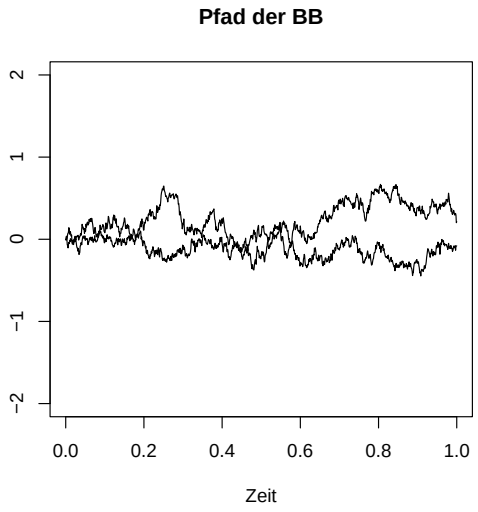
```r
# gemeinsam als image/contour :
schrw <- 0.333
x <- seq(from=-2, to=2, by=schrw)
y <- seq(from=-2, to=2, by=schrw)

pf2beob <- matrix(0, nrow=length(x), ncol=length(y))
for (i in 1:length(x)) {
  for (j in 1:length(y)) {
    pf2beob[i,j] <- sum( (beob.ExpIrrf[,1]>=x[i]- schrw/2) & (beob.ExpIrrf[,1]<x[i] + schrw/2)
                         & (beob.ExpIrrf[,2]>=y[j] - schrw/2) & (beob.ExpIrrf[,2] < y[j] + schrw/2))
    ## beachte: "&", nicht "&&"
  }
}

image(x,y,pf2beob, main="empirische Verteilung", xlab="Ort zur Zeit 0.5", ylab="Ort zur Zeit 1.0")
contour(x,y,pf2beob, add=TRUE, nlevels=20)


# Gemeinsame Verteilung:
# Welcher Anteil hat Pos. zur Zeit 0.5 <= x1 und
# Pos. zur Zeit 1.0 <= x2 ?
x1 <- 0.5; x2 <- -0.1
sum(beob.ExpIrrf[,1] <= x1 & beob.ExpIrrf[,2] <=x2)/N

## [1] 0.445
```

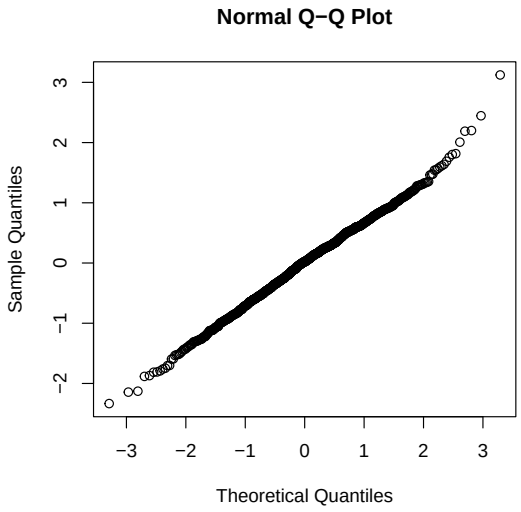


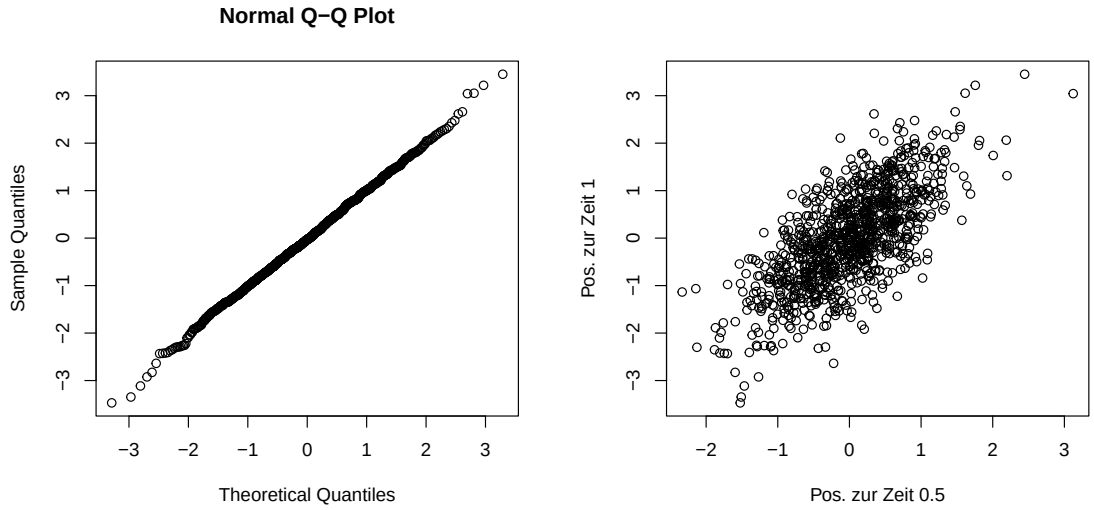



```
# ########################################################
#
# Brownsche Bewegung: Normalverteilte, "stationäre" (in Verteilung),
# unabhängige Zuwächse

# Simuliere längs einem Gitter von Zeitwerten mit Abstand delta:

delta <- 0.001 # variieren: delta <- 0.002; delta <- 0.001

N <- ceiling(1/delta)

zeitgitter <- (0:N)/N

# Einen Brownschen Pfad auf die offensichtliche Weise simulieren:
BBpfad <- seq(0, along=zeitgitter)
for (i in 1:N) {
  BBpfad[i+1] <- BBpfad[i]+rnorm(1, sd=sqrt(delta))
}

# und anschauen
plot(zeitgitter, BBpfad, type="l", xlab="Zeit",ylab="", ylim=c(-2,2),
     main="Pfad der BB")

# für ein festes Zeitintervall geht es schneller wie oben mittels
BBpfadsim <- function(n, delta) c(0,cumsum(rnorm(n, sd=sqrt(delta))))

BBpfad2 <- BBpfadsim(N, delta)
points(zeitgitter, BBpfad2, type="l")

#
# Betrachten wir wieder die gemeinsame Verteilung der Pfade zur
# Zeit 0.5 und zur Zeit 1:
```

```r
wdh <- 1000
beob.BB <- matrix(0, nrow=wdh, ncol=2)

for (i in 1:wdh) {
  zw1 <- rnorm(1, sd=sqrt(0.5)); zw2 <- rnorm(1, sd=sqrt(0.5))
  beob.BB[i,]<-c(zw1, zw1+zw2)
}

# Vert. zur (reskalierten) Zeit 0.5:
mean(beob.BB[,1]); var(beob.BB[,1])

## [1] -0.009082738
## [1] 0.492671

qqnorm(beob.BB[,1])

# Vert. zur (reskalierten) Zeit 1:
mean(beob.BB[,2]); var(beob.BB[,2])

## [1] -0.0131825
## [1] 1.030001

qqnorm(beob.BB[,2])

# gemeinsam:
plot(beob.BB[,1], beob.BB[,2], xlab="Pos. zur Zeit 0.5", ylab="Pos. zur Zeit 1")

# Gemeinsame Verteilung:
# Welcher Anteil hat Pos. zur Zeit 0.5 <= x1 und
# Pos. zur Zeit 1.0 <= x2 ?
x1 <- 0.5; x2 <- -0.1
sum(beob.BB[,1] <= x1 & beob.BB[,2] <=x2)/N

## [1] 0.443
```

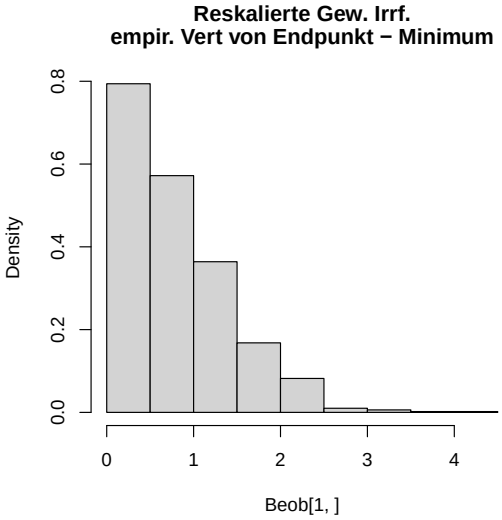


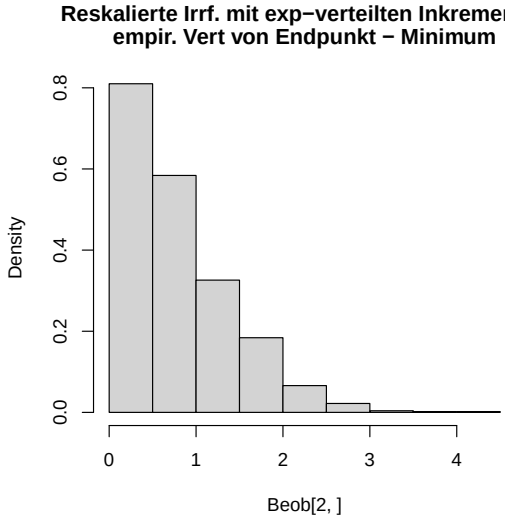


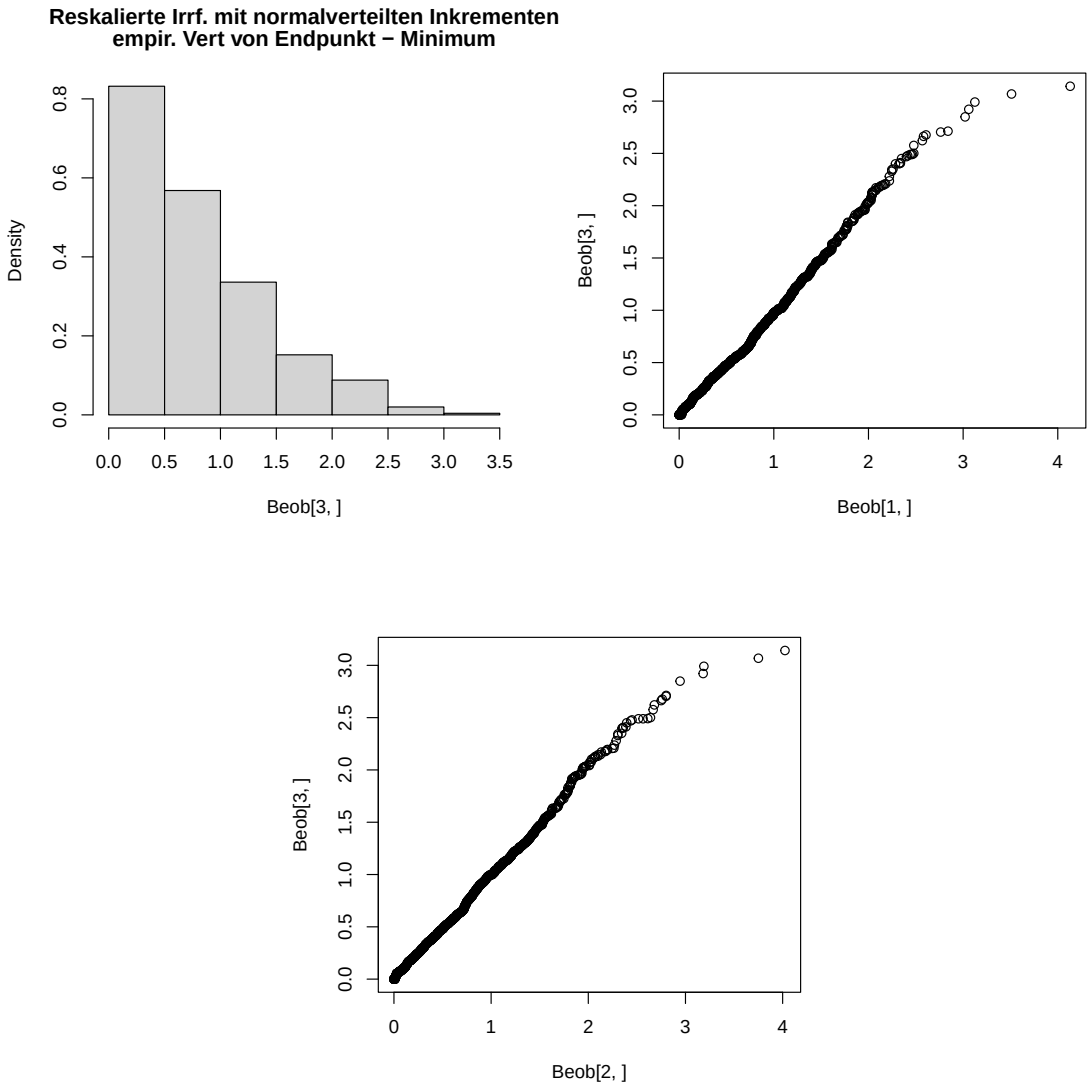



```
############################################################
#
# Betrachten wir zur Zeit 1, wie weit der Pfad nun oberhalb
# seines bisherigen Minimalwerts liegt
# (beachte: dies hat nicht die Eigenschaft,
# von nur endlich vielen Zeitpunkten abzuhängen)

# Wir betrachten wie oben drei Verteilungen von Inkrementen:

N <- 1000  # auch variieren, N <- 10000
delta <- 1/N

p <- 0.4
mu <- p*1+(1-p)*(-1)
# sigma=sqrt(1-mu^2)

pfadsim.gewIrrf <- function(n) c(0,(cumsum(2*rbinom(n, size=1, prob=p)-1)-
                                    (1:n)*mu)/(sqrt(1-mu^2)*sqrt(n)))

lambda <- 1.8 # ggfs. variieren, z.B. 2.0
# hier: mu= 1/lambda
# sigma= 1/lambda

pfadsim.ExpIrrf <- function(n) c(0,(cumsum(rexp(n, rate=lambda))-
                                    (1:n)*(1/lambda))/((1/lambda)*sqrt(n)))

# hier delta=1/n
pfadsim.BB <- function(n) c(0,cumsum(rnorm(n, sd=sqrt(1/n))))

#
wdh <- 1000
Beob <- matrix(0, nrow=3, ncol=wdh)
```

```r
for (i in 1:wdh) {
  pf <- pfadsim.gewIrrf(N); Beob[1,i] <- pf[N]-min(pf)
  pf <- pfadsim.ExpIrrf(N); Beob[2,i] <- pf[N]-min(pf)
  pf <- pfadsim.BB(N); Beob[3,i] <- pf[N]-min(pf)
}

#
hist(Beob[1,], prob=TRUE, main=c("Reskalierte Gew. Irrf.",
                                 "empir. Vert von Endpunkt - Minimum"))
hist(Beob[2,], prob=TRUE, main=c("Reskalierte Irrf. mit exp-verteilten Inkrementen",
                                 "empir. Vert von Endpunkt - Minimum"))
hist(Beob[3,], prob=TRUE, main=c("Reskalierte Irrf. mit normalverteilten Inkrementen",
                                 "empir. Vert von Endpunkt - Minimum"))

qqplot(Beob[1,],Beob[3,])
qqplot(Beob[2,],Beob[3,])
```






Reskalierte Irrf. mit normalverteilten Inkrementen
empir. Vert von Endpunkt − Minimum
Density
Beob[3,]
Beob[1,]
Beob[3,]
Beob[2,]