# Bayesianische Mischverteilungsmodelle

Volker Schmid

17. Juli 2017

# Bayesianische Mischverteilungsmodelle

# Bayesianische Mischverteilungsmodelle

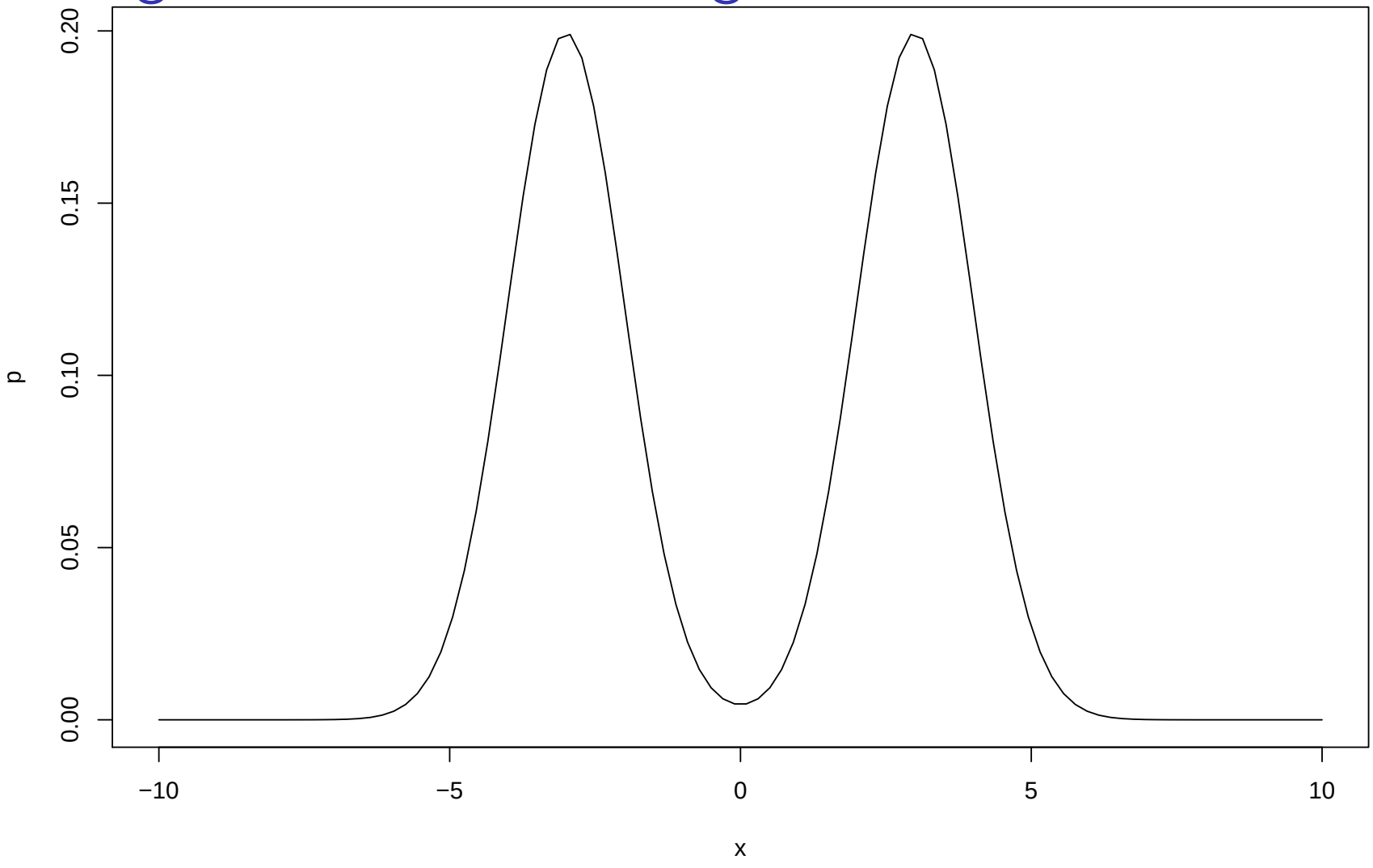
# Mischverteilungen

- (Normal-)Verteilungsannahme basiert in der Regel auf der Annahme, dass die Beobachtungen identisch verteilt sind, es also eine Verteilung der Grundgesamtheit gibt
- Alternative Annahme: Es gibt mehrere Gruppen in der Grundgesamheit, die alle die selbe Verteilungsklasse haben, aber unterschiedliche Parameter
- Einfachster Fall: Mischung von Normalverteilungen

# Mischung von Normalverteilungen I

```
x <- seq(-10, 10, length=100)
p <- 0.5*dnorm(x,-3,1) + 0.5*dnorm(x,3,1)
plot(x,p, type="l")
```

# Mischung von Normalverteilungen II

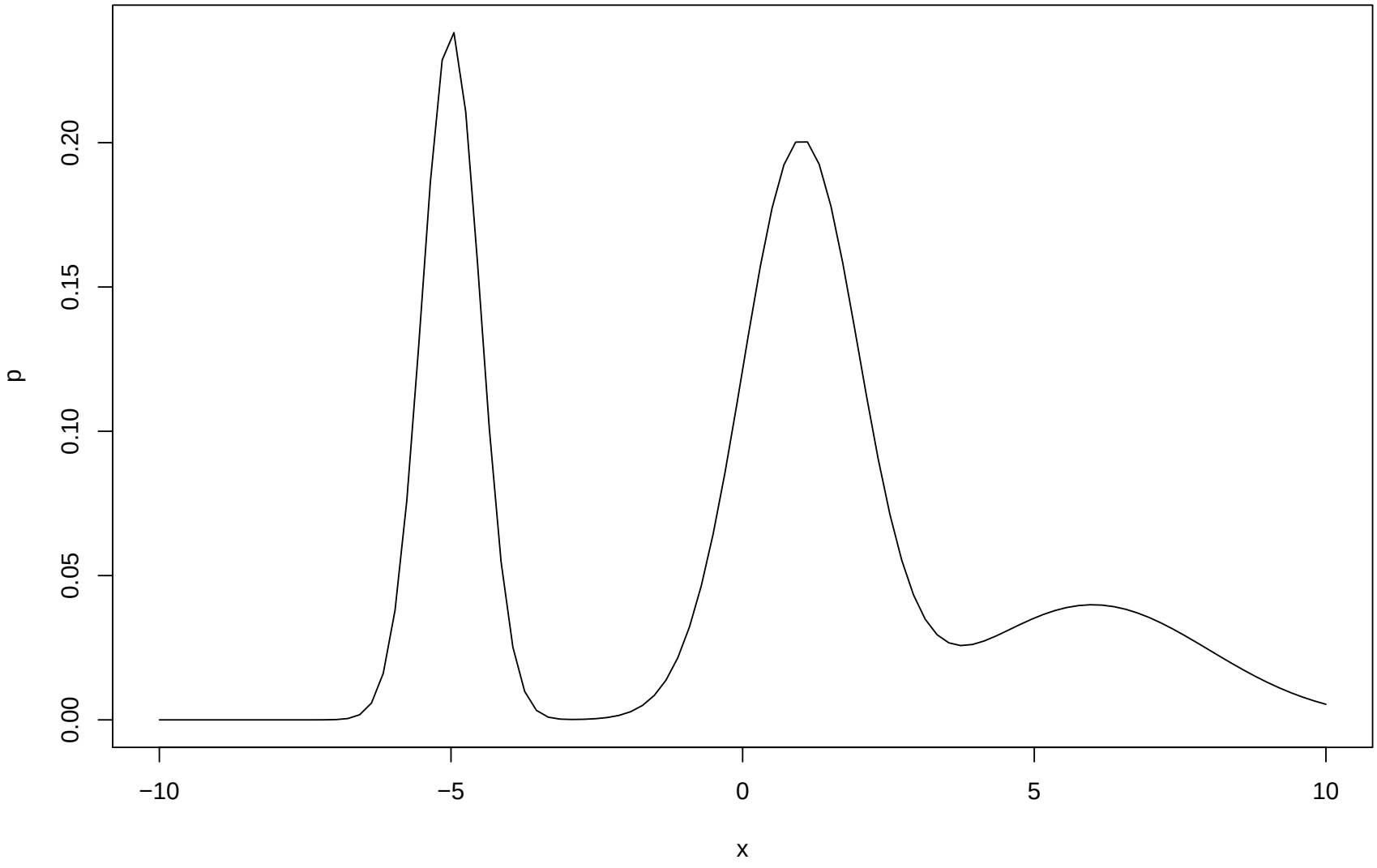



- Natürlich sind auch verschiedene Varianzen und verschiedene Gewichte möglich

# Mischung von Normalverteilungen III

```
x <- seq(-10, 10, length=100)
p <- 0.3*dnorm(x,-5,0.5) + 0.5*dnorm(x,1,1) + 0.2*dnorm(x,6
plot(x,p, type="l")
```

# Mischung von Normalverteilungen IV



# Fragen

- Welche Parameter und welche Gewichte haben die einzelnen Verteilungen?
- Wieviele Mischverteilungen brauchen wir?
- Zu welcher Verteilung gehört welche Beobachtung

# Latente Klassen

Idee der Modellierung: Führe latente Klassen $S_i$ ein

$$
\begin{aligned}
x_i | S_i &\sim N(\mu_{S_i}, \sigma^2_{S_i}) \\
\mu_j &\sim N(\mu_0, \sigma^2_0) \\
\sigma^2_j &\sim IG(a, b) \\
p(S_i = j) &= \pi_j \\
(\pi_1, \ldots, \pi_K) &\sim Diri(\alpha, \ldots, \alpha)
\end{aligned}
$$

Dirichlet-Verteilung:

$$
p(\pi_1, \ldots, \pi_K) \propto \prod \pi_i^{\alpha_i - 1}
$$

# Full conditionals

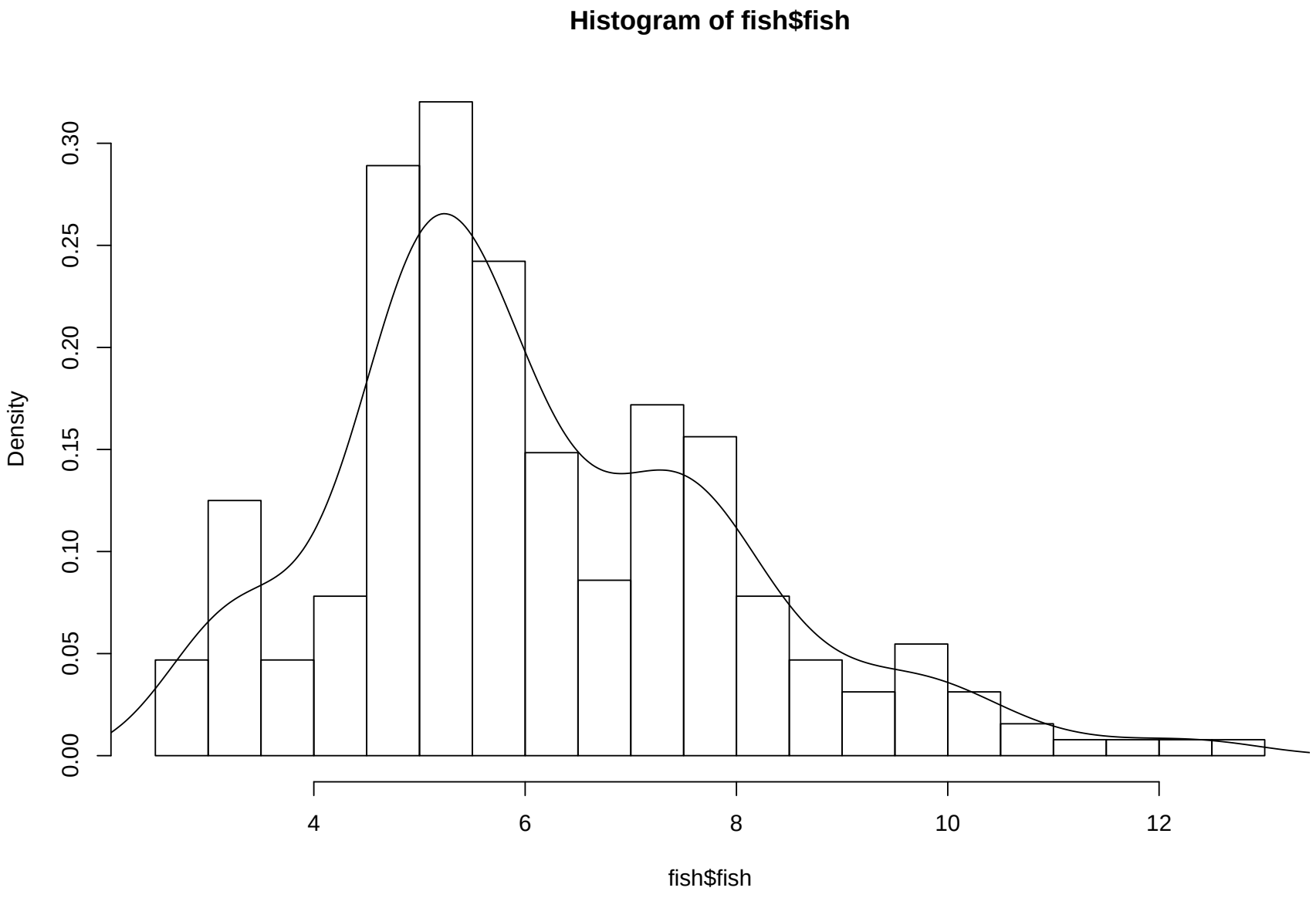
- Die full conditional von $\mu_j$ sind Normalverteilungen, wobei die $x_i$ mit $S_i = j$ eingehen
- Analog ist die full conditional von $\sigma_j^2$ eine IG-Verteilung
- $p(S_i = j)$ berechnet sich aus Dichte von $x_i$ mit $\mu_j$ und $\sigma_j^2$ (und Priori)
- Full Conditional von $\pi$ ist Dirichlet mit $(\alpha + n_1, \ldots, \alpha + n_k)$, wobei $n_j$ die aktuelle Anzahl der Beobachtungen in Klasse $j$ ist

# Beispiel I

Länge von 256 Fischen (aus D. M. Titterington, A. F. M. Smith and U.E. Makov (1985) Statistical Analysis of Finite Mixture Distributions. Wiley.)

```
data(fish, package="bayesmix")
hist(fish$fish, freq=FALSE, n=22)
lines(density(fish$fish))
```

# Beispiel II


Histogram of fish$fish
Density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
4
6
8
10
12
fish$fish

# bayesmix-Paket I

```
library(bayesmix)
model <- BMMmodel(fish, k = 4,
  initialValues = list(S0 = 2),
  priors = list(kind = "independence",
  parameter = "priorsFish",
  hierarchical = "tau"))
```

```
print(model)
```

```
## Data for nodes: b0, B0inv, nu0Half, g0Half, g0G0Half, k
## Initial values for nodes: eta, mu, tau, S0
##
## Model specification in BUGS language:
##
## var
```

# bayesmix-Paket II

```
##  b0,
##      B0inv,
##      nu0Half,
##      g0Half,
##      g0G0Half,
##      k,
##      N,
##      eta[4],
##  mu[4],
##  tau[4],
##  nu0S0Half,
##      S0,
##      e[4],
##  y[256],
##  S[256];
##
## model    {
```

# bayesmix-Paket III

```
##  for (i in 1:N) {
##      y[i] ~ dnorm(mu[S[i]],tau[S[i]]);
##      S[i] ~ dcat(eta[]);
##  }
##  for (j in 1:k) {
##      mu[j] ~ dnorm(b0,B0inv);
##      tau[j] ~ dgamma(nu0Half,nu0S0Half);
##  }
##  S0 ~ dgamma(g0Half,g0G0Half);
##  nu0S0Half <- nu0Half * S0;
##
##  eta[] ~ ddirch(e[]);
## }
```

# JAGS

- **JAGS** oder **Just another Gibbs sampler** ist eine Software zur MCMC-Simulation.
- Definition der Modelle entspricht der bei WinBUGS bzw. OpenBUGS
- Anbindung an R und andere Sprachen

# Fortsetzung Beispiel I

```
control <- JAGScontrol(variables = c("mu", "tau", "eta",
    "S"), burn.in = 1000, n.iter = 5000, seed = 10)
```

```
z <- JAGSrun(fish, model = model, control = control)
```

```
## Compiling model graph
##    Declaring variables
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 256
##    Unobserved stochastic nodes: 266
##    Total graph size: 1047
##
## Initializing model
```

# Fortsetzung Beispiel II

```
zSort <- Sort(z, by = "mu")
zSort
```

```
##
## Call:
## JAGSrun(y = fish, model = model, control = control)
##
## Markov Chain Monte Carlo (MCMC) output:
## Start = 1001
## End = 6000
## Thinning interval = 1
##
##  Empirical mean, standard deviation and 95% CI for eta
##            Mean      SD    2.5%  97.5%
## eta[1] 0.1289 0.07699 0.07061 0.3425
## eta[2] 0.5003 0.07249 0.34015 0.5962
## eta[3] 0.2702 0.05194 0.15888 0.3669
```

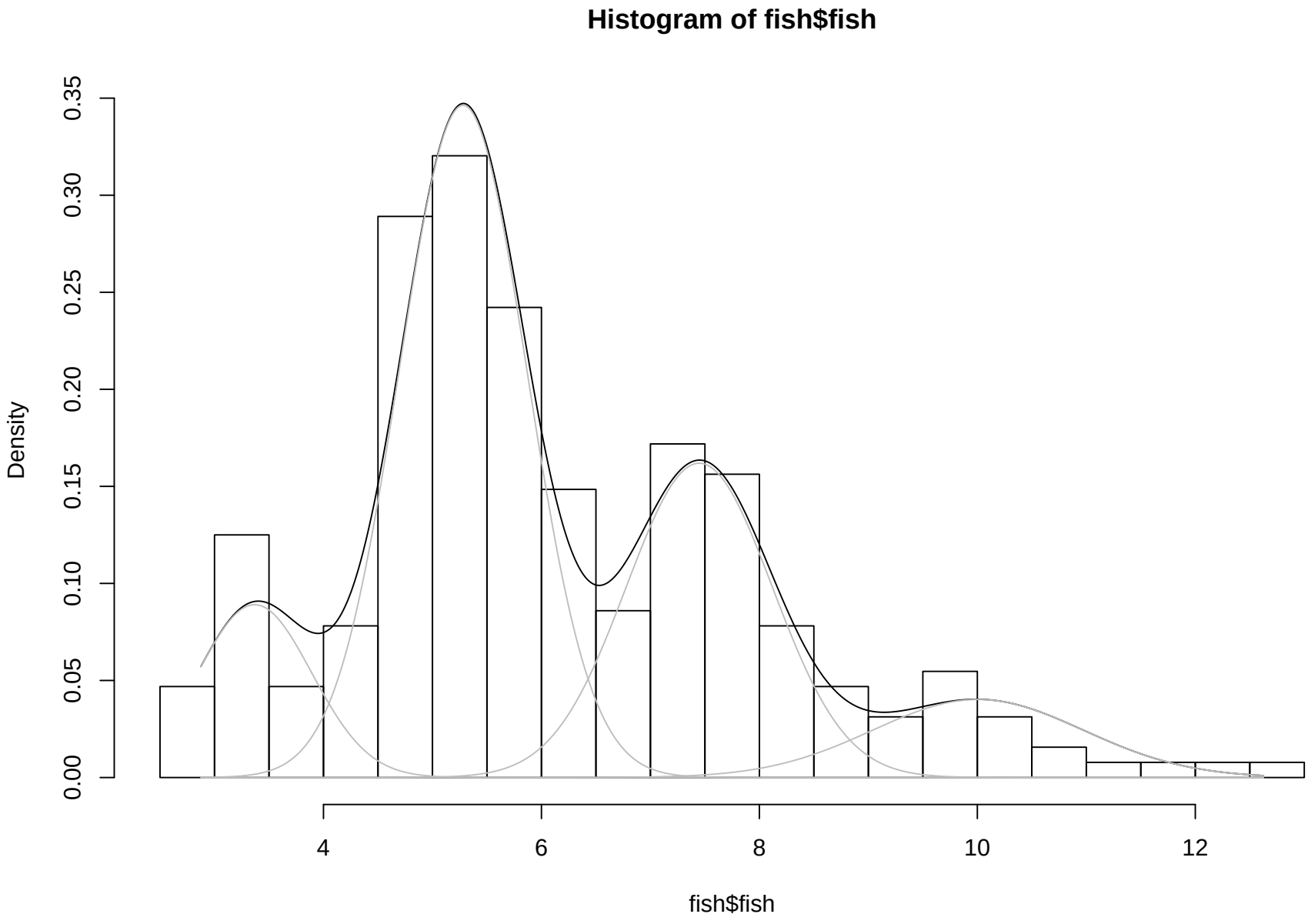
# Fortsetzung Beispiel III

```
## eta[4] 0.1007 0.03242 0.05115 0.1751
##
##  Empirical mean, standard deviation and 95% CI for mu
##         Mean     SD  2.5%  97.5%
## mu[1] 3.444 0.3656 3.128  5.005
## mu[2] 5.299 0.1868 5.129  5.481
## mu[3] 7.436 0.2296 7.090  7.720
## mu[4] 9.954 0.4480 8.819 10.658
##
##  Empirical mean, standard deviation and 95% CI for sigma
##              Mean     SD   2.5% 97.5%
## sigma2[1] 0.3477 0.3012 0.1485 1.405
## sigma2[2] 0.4055 0.3024 0.2403 1.377
## sigma2[3] 0.5327 0.4037 0.2389 1.400
## sigma2[4] 1.0431 0.4775 0.5389 2.298
```

# Medianmodell I

```
postmed<-apply(zSort$results, 2, median)
x<-seq(min(fish$fish), max(fish$fish), length=1000)
d1 <- postmed[257]*dnorm(x, postmed[261], sqrt(postmed[265]
d2 <- postmed[258]*dnorm(x, postmed[262], sqrt(postmed[266]
d3 <- postmed[259]*dnorm(x, postmed[263], sqrt(postmed[267]
d4 <- postmed[260]*dnorm(x, postmed[264], sqrt(postmed[268]
d <- d1 + d2 + d3 + d4
hist(fish$fish, freq=FALSE, n=22, ylim=c(0,max(d)))
lines(x, d)
lines(x, d1, col="grey")
lines(x, d2, col="grey")
lines(x, d3, col="grey")
lines(x, d4, col="grey")
```

# Medianmodell II



# Posteriori-Verteilung der Klassen I

```
S <- apply(zSort$results[,1:256],2,
     bioimagetools::table.n,4, percentage=TRUE)
barplot(S, col=c("red","green","blue","orange"))
```

# Posteriori-Verteilung der Klassen II