```
library(lattice)
library(latticeExtra)
# Für qda()
library(MASS)
# Für mehrdimensionale Verteilungen
library(mvtnorm)
source(file.path(pfadu, "ellipsefun.R"))
# Formant-Daten
v.df = read.table(file.path(pfadu, "v.df.txt"))


######################################################## TEIL 1: die Normalverteilung
# Siehe: http://www.phonetik.uni-muenchen.de/~jmh/lehre/sem/ws1213/Rspeech/gaussian.
pdf
# 10 Würfel werfen

sample(1:6, 10, replace=T)

# Den Mittelwert davon berechnen
mean(sample(1:6, 10, replace=T))

# Diesen Vorgang k = 50 Mal wiederholen
erg = NULL
for(k in 1:50){
vec = mean(sample(1:6, 10, replace=T))
erg = c(erg, vec)
}

# Histogramm davon
par(mfrow=c(1,2))
hist(erg)
hist(erg, freq=F)

# Die Normalverteilung davon
mu = mean(1:6)
# Die Standardabweichung der Mittelwerte, angenommen
# wir berechnen den Mittelwert von 10 Würfeln
SE = (sqrt((sum((1:6)^2)/6 - mu^2)))/sqrt(10)
curve(dnorm(x, mu, SE), add=T)

################################################### 2. Conditional probability
(bedingte Wahrscheinlichkeit)

# Was ist die Wahrscheinlichkeit, dass ich ein Mittelwert von 4.0 bekommen,
# gegeben dieses Modell (Ein Würfel wird 10 Mal geworfen, ich berechne den Mittelwert
davon)
dnorm(4, mu, SE)
abline(h = dnorm(4, mu, SE))

######### Sprache: Ein Dimensionales Modell
temp = v.df$Vpn == "S67"
m.df = v.df[temp,]; f.df = v.df[!temp,]
```

```
# Die F1-Werte von Sprecher 68 zum zeitlichen Mittelpunkt, /a/
temp = m.df$V == "a"
F1 = m.df$F1[temp]

hist(F1); hist(F1, freq=F)
# Die beste Einschätzungen von mu und SE dieser Verteilung
mu = mean(F1)
s = sd(F1)
curve(dnorm(x, mu, s), 400, 900, add=T)

# Conditional probability: p(600|a)
# Was ist die bedingte Wahrscheinlichkeit, dass 600 Hz vorkommt,
# gegeben dieser Verteilung von /a/

dnorm(600, mu, s)

# Bedingte Warscheinlichkeiten für alle 3 Klassen (p(600|E), p(600|a), p(600|I)
cond = NULL
x = 600
for(j in sort(unique(m.df$V))){
temp = m.df$V == j
mu = mean(m.df$F1[temp])
s = sd(m.df$F1[temp])
prob = dnorm(x, mu, s)
cond = c(cond, prob)
}

names(cond) = sort(unique(m.df$V))


##################################################### 3. Prior probability
# die Häufigkeit mit der eine Klasse
# vorkommt, *im Verhältnis zu anderen Klassen*
# Für Sprecher 67
prior = prop.table(table(m.df$V))

##################################################### 4. Posterior probability
# Gegeben diese Klassen, was
# ist die Wahrscheinlichkeit,
# dass ein Wert von 600 Hz /a/ (und nicht /I, E/)
# sein könnte: p(a | 600)?

# Bayes-Formel
# p(a | 600) = (p(600|a) * prior.a) / ((p(600|a) * prior.a) + (p(600|E) * prior.E) +
(p(600|I) * prior.I))
# Wir berechnen hier gleichzeitig: p(a|600), p(E|600), p(I|600)
post = (prior * cond)/ sum(prior * cond)

# das Gleiche mit der Funktion qda()
m.qda = qda(V ~ F1, data = m.df)
m.qda
predict(m.qda, data.frame(F1 = 600))
```

```
F1werte = seq(200, 1200, by = 10)
o = predict(m.qda, data.frame(F1 = F1werte))


################################################ 5. Selbst-Klassifizierung (closed
test)
p = predict(m.qda)
p = predict(m.qda, m.df)

## Verwechslungsmatrix: ursprüngliche x klassifizierte Etikettierungen
tab = table(m.df$V, p$class)
tab

################################################ 6. Klassifizierung mit anderen Daten
################################################ (open test)

# Open-test: Training und Test mit anderen Daten
f.pred = predict(m.qda, f.df)
tab = table(f.df$V, f.pred$class)

################################################ 7.  Hit-rate
diag(tab)/apply(tab, 1, sum)

# Total hit-rate
sum(diag(tab))/sum(tab)

################################################ 8.  Klassifikation in 2 oder
mehreren Dimensionen


# 8a Bedingte Wahrscheinlichkeit
temp = f.df$V == "a"
dat = with(f.df[temp,], cbind(F1, F2))
mu = apply(dat, 2, mean)
# Für Kovarianz, siehe: http://www.phonetik.uni-muenchen.de/~jmh/lehre/sem/ss12/
statistik/r1.pdf
kv = var(dat, dat)
# Bedingte Wahrscheinlichkeit von (600, 1400)
dmvnorm(c(600, 1400), mu, kv)



################## (Fakultativ: 2D Histogramm und Gaussglocken)
# 8a 2D Histogramm
temp = f.df$V == "a"

F1 = cut(f.df$F1[temp], 5)
# xtabs(F1)
F2 = cut(f.df$F2[temp], 5)
F1F2tab = xtabs( ~ F1 + F2)
F1F2.df <- as.data.frame.table(F1F2tab)

cloud(Freq ~ F1 * F2, data = F1F2.df,
```

```
      screen = list(z = -40, x = -40), zoom = 1.1,
      col.facet = "grey", xbase = 0.6, ybase = 0.6,
      par.settings = list(box.3d = list(col = "transparent")),
      aspect = c(1.5, 0.75), panel.aspect = 0.75,
      panel.3d.cloud = panel.3dbars)


# Die Gaussglocke für 2 Dimensionen
gr = expand.grid(F1 = do.breaks(c(200, 1300), 50), F2 = do.breaks(c(600, 2500), 50))
# Bedingte Wahrscheinlichkeit dieser Werte
dat.cond = dmvnorm(gr, mu, kv)
w.dat = data.frame(prob = dat.cond, F1 = gr[,1], F2 = gr[,2])
wireframe(prob ~ F1 * F2, data = w.dat, outer = TRUE, zlab = "Cond.prob",
screen = list(z = -30, x = -50),
scales = list(lwd = .5))

# Oder für alle 3 Glaussglocken gleichzeitig
mat = NULL
for(j in c("a", "E", "I")){
temp = f.df$V == j
dat = cbind(f.df$F1[temp], f.df$F2[temp])
mu = apply(dat, 2, mean)
kv = var(dat, dat)
dat.cond = dmvnorm(gr, mu, kv)
mat$labs = c(mat$labs, rep(j, nrow(gr)))
mat$cond = c(mat$cond, dat.cond)
}
p.df = data.frame(F1 = gr[,1], F2 = gr[,2], V = factor(mat$labs), den = mat$cond)

wireframe(den ~ F1 * F2, group = V, data = p.df, outer = TRUE, zlab = "",
screen = list(z = -30, x = -60),
scales = list(lwd = .5), auto.key=T)

##########################################################

# 8b Klassifikation in 2 (oder mehreren Dimensionen)
m.qda = qda(V ~ F1 + F2, data = f.df)
m.pred = predict(m.qda, f.df)
tab = table(f.df$V, m.pred$class)
# hit rate
diag(tab)/apply(tab, 1, sum)


#######################
########################################################## 9. Klassifikationen in hö
heren Dimensionen

# Die Daten
library(emu)
f5 = dcut(fr.dft, .5, prop=T)
table(fr.l, fr.sp)

# Energie-Summe in 20 mel-Filtern
```

```
m5 = mel(f5)
N = 20
int = seq(0, max(trackfreq(m5)), length=N+2)
left = int[1:N]
right = int[3:(N+2)]
cbind(left, right)
fsum = NULL
for(j in 1:N){
unten = left[j]
oben = right[j]
vals = fapply(m5[,unten:oben], sum, power=T)
fsum = cbind(fsum, vals)
}

# PCA durchführen getrennt pro Sprecher
pmat = matrix(0, nrow=nrow(fr.dft), ncol = 20)
for(j in unique(fr.sp)){
temp = fr.sp == j
pmat[temp,] = prcomp(fsum[temp,], scale=T)$x
}

p.df = data.frame(pmat, K = factor(fr.l), Vpn = factor(fr.sp))
# Klassifikation in z.B. 3 Dim
temp = p.df$Vpn == "68"
p3.qda = qda(K ~ X1 + X2 + X3, data = p.df[temp,])
p3.pred = predict(p3.qda, p.df[!temp,])
table(p.df[!temp,]$K, p3.pred$class)

res = NULL
for(j in 2:18){
xnam = paste0("X", 1:j)
fmla <- as.formula(paste("K ~ ", paste(xnam, collapse= "+")))
p.qda = qda(fmla, p.df[temp,])
p.pred = predict(p.qda, p.df[!temp,])
p.tab = table(p.df[!temp,]$K, p.pred$class)
res = c(res, sum(diag(p.tab))/sum(p.tab))
}
```