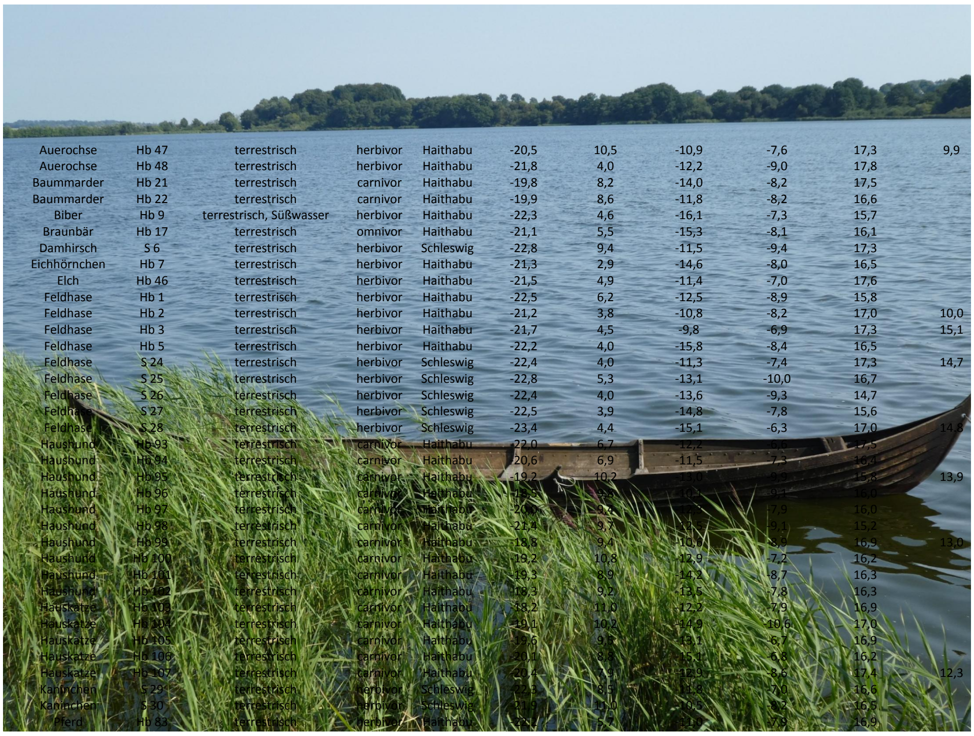
„Anwendung KDD-basierter Methoden zur Interpretation multi-dimensionaler Isotopen-Fingerabdrücke“



R-Skript zur Dissertation

eingereicht an der

Fakultät für Biologie

der Ludwig-Maximilians-Universität München

vorgelegt von

Andrea Barbara Göhring

München, 19. Februar 2019

Titelbild: Schlei bei Haithabu (© A. Göhring)

Inhaltsverzeichnis

1 Einführende Worte

Das vorliegende R-Skript stellt den Begleitband zur Dissertation zum Thema „Anwendung KDD-basierter Methoden zur Interpretation multi-dimensionaler Isotopen-Fingerabdrücke“ dar. Das R-Skript soll es ermöglichen, die in der Dissertation angewendeten Methoden, wie z. B. Clusteranalyse, auch auf andere (Isotopen-)Daten anzuwenden.

Die dargestellten R-Befehle sind (mit Ausnahme der Mischungsmodelle) mittels des Beispielsdatensatzes der Fische aus Haithabu und Schleswig („Datensatz_Fische.csv“; Tabelle A1) lauffähig. Grau hinterlegt finden sich jeweils die R-Befehle, sowie deren mittels R erzeugte Ausgabe.

Für die verwendeten Mischungsmodelle gibt es bereits publizierte Beschreibungen (vgl. Kapitel 10.1 bis 10.3). Hier werden nur die R-Befehle abgedruckt und kommentiert. Diese sind jedoch mit entsprechenden Daten für Konsumenten, Nahrungsquellen und Tophiestufenfaktoren ebenfalls lauffähig.

Das vorliegende R-Skript wurde mit dem R-Paket „knitr“ (Version 1.20; Xie, 2018) erstellt.

2 Einlesen des Beispieldatensatzes

Um den Beispieldatensatz „Datensatz_Fische.csv“ (vgl. Tabelle A1) einzulesen, muss die „Working Directory“ von „R“ auf den Speicherort dieser Datei gelegt werden.

```
# setwd(...)
Daten <- read.table("Datensatz_Fische.csv", header = TRUE, sep = ";")
head(Daten)

#   Spezies Probennr.                 Habitat Ernährung   Fundort
# 1 Brachse   35 B2Op  Süßwasser, Brackwasser   omnivor Schleswig
# 2 Brachse  36 B3Pop  Süßwasser, Brackwasser   omnivor Schleswig
# 3 Brachse  37 B4Pop  Süßwasser, Brackwasser   omnivor Schleswig
# 4 Brachse    38 B5C  Süßwasser, Brackwasser   omnivor Schleswig
# 5  Dorsch     1 D1V Brackwasser, Salzwasser   omnivor  Haithabu
# 6  Dorsch     3 D3V Brackwasser, Salzwasser   omnivor  Haithabu
#   d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# 1       -27.73         6.02        -0.64       -13.04
# 2       -25.07         6.17        -1.66       -11.68
# 3       -22.41         5.38        -1.60       -12.01
# 4       -27.37        10.72        -4.46       -15.07
# 5       -16.06        13.68        -1.82       -13.20
# 6       -16.21        15.89        -1.30        -3.76
```

3 Detektion und Entfernen multivariater Ausreißer

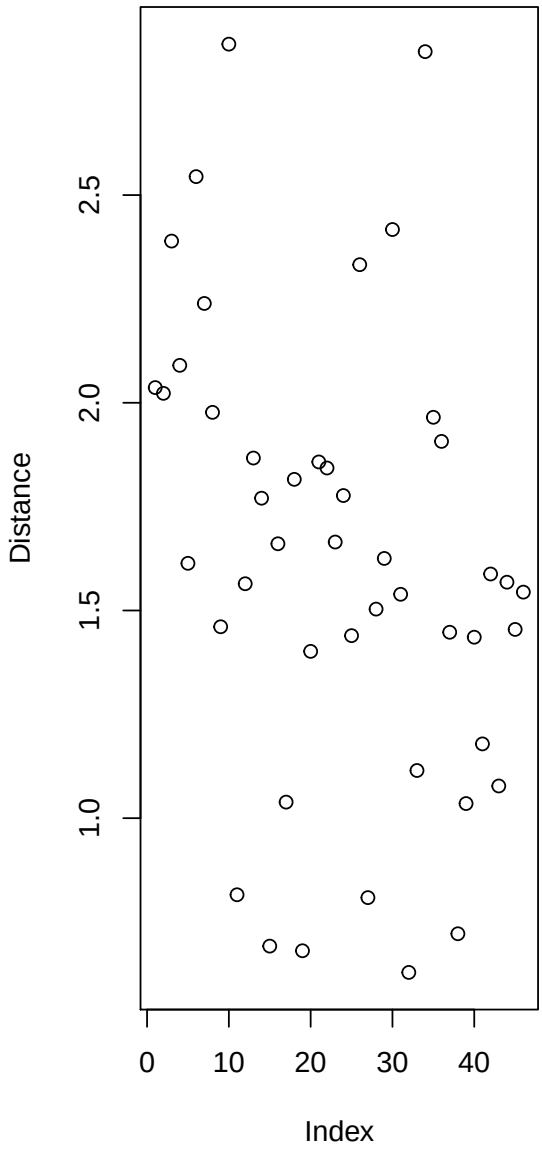
Die R-Pakete

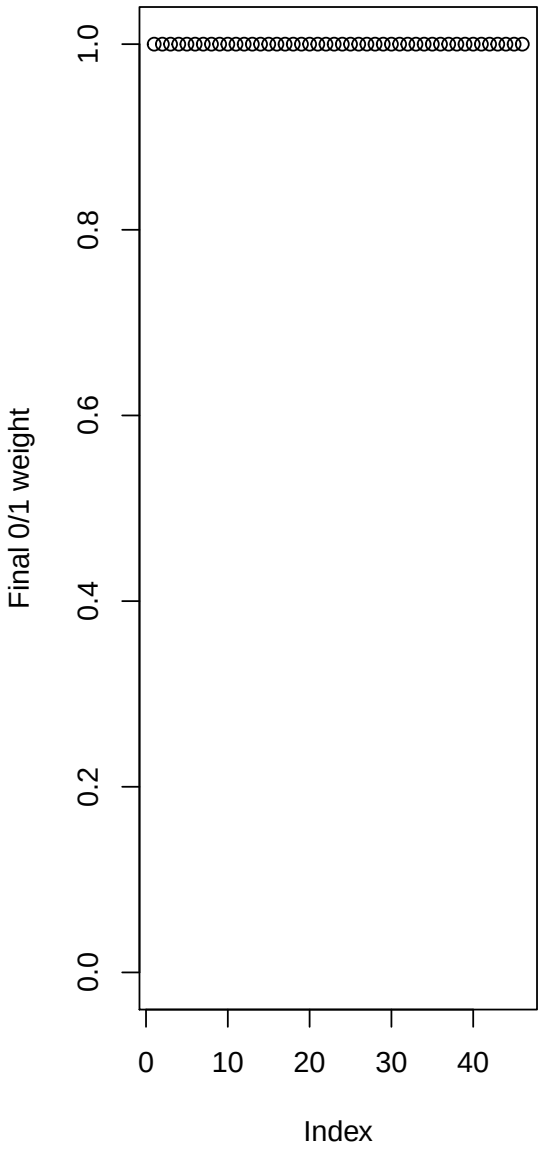
- „mvoutlier“ (Version 2.0.9; Filzmoser & Gschwandtner, 2017)
- „dplyr“ (Version 0.7.7; Wickham et al., 2018b)

müssen installiert sein.

```
# install.packages(c("mvoutlier", "dplyr"))
library(mvoutlier)
library(dplyr)
```

```
# Auswahl der Spalten im Datensatz, die Isotopenwerte enthalten
# hier: Spalte 6 - 9 (sh. head(Daten))
Outlier <- sign1(Daten[, 6:9], makeplot = TRUE, qcrit = 0.975)
```






```
# Ausreißer werden mit "0" markiert
Outlier$wfinal01

#  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
# [34] 1 1 1 1 1 1 1 1 1 1 1 1

# Der Beispieldatensatz enthält keine multivariaten Ausreißer
# Jedoch wird beispielhaft gezeigt, wie bei Vorhandensein von Ausreißern
# vorgegangen werden muss
```

```r
# Erzeugen einer neuen Spalte (Ausreisser) im Datensatz
Ausreisser <- Outlier$wfinal01
Daten <- cbind(Daten, Ausreisser)
# Auswahl derjenigen Individuen, die keine Ausreißer
# (Ausreisser == "1") sind
Daten <- filter(Daten, Ausreisser == "1")
head(Daten)

#   Spezies Probennr.                Habitat Ernährung   Fundort
# 1 Brachse   35 B2Op  Süßwasser, Brackwasser   omnivor Schleswig
# 2 Brachse  36 B3Pop  Süßwasser, Brackwasser   omnivor Schleswig
# 3 Brachse  37 B4Pop  Süßwasser, Brackwasser   omnivor Schleswig
# 4 Brachse    38 B5C  Süßwasser, Brackwasser   omnivor Schleswig
# 5  Dorsch     1 D1V Brackwasser, Salzwasser   omnivor  Haithabu
# 6  Dorsch     3 D3V Brackwasser, Salzwasser   omnivor  Haithabu
#   d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat Ausreisser
# 1       -27.73         6.02        -0.64       -13.04          1
# 2       -25.07         6.17        -1.66       -11.68          1
# 3       -22.41         5.38        -1.60       -12.01          1
# 4       -27.37        10.72        -4.46       -15.07          1
# 5       -16.06        13.68        -1.82       -13.20          1
# 6       -16.21        15.89        -1.30        -3.76          1
```

4 Clusteranalyse

4.1 k-means

1) Skalieren der Daten

Für die Clusteranalyse mittels k-means werden die Isotopendaten zunächst skaliert.

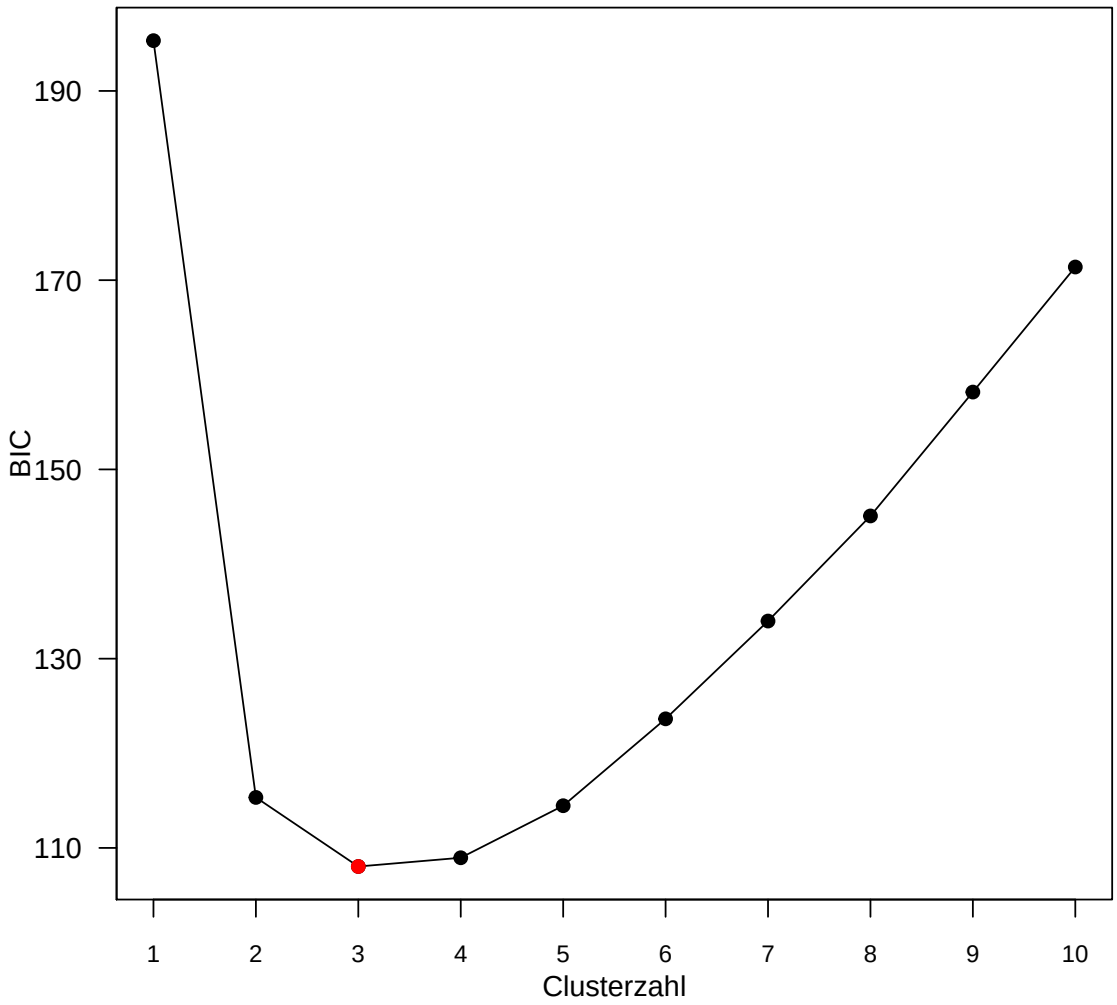
```
# Skalierung der Isotopendaten
Isotopen_skaliert <- scale(Daten[, 6:9])
```

2) Finden der optimalen Clusterzahl mittels BIC

Anschließend kann die optimale Clusterzahl z. B. mit Hilfe des BIC untersucht werden.

```
# Bestimmen des BIC für k-means mit 1 - 10 Clustern
logLik.kmeans <- function(object) structure(
  -object$tot.withinss/2,
  df = nrow(object$centers)*ncol(object$centers),
  nobs = length(object$cluster)
)

BIC <- c()
for(i in 1:10){
  kc <- kmeans(Isotopen_skaliert, i, nstart = 100)
  BIC[[i]] <- BIC(kc)
}
print(BIC)

#  [1] 195.3146 115.3361 108.0355 108.9598 114.4592 123.6418 133.9765
#  [8] 145.0846 158.1741 171.3912
```

```
# pdf(width = 6, height = 6, "kmeans_BIC_Plot.pdf",
# encoding = "MacRoman")
plot(1:10, BIC, type = "n", ann = FALSE, axes = FALSE)
lines(1:10, BIC, col = "black")
points(1:10, BIC, pch = 19, col = "black")
points(match(min(BIC), BIC), min(BIC), col = "red", pch = 19)
box()
axis(1 ,at = seq(1, 10, 1), cex.axis = 0.8)
axis(2, at = seq(10, 240, 20), las = 2)
mtext(side = 1, line = 2, "Clusterzahl")
mtext(side = 2, line = 2.3, "BIC")
```




```
# dev.off()
```

3) Clusteranalyse

Die Clusterzahl wird nun festgelegt, z. B. auf drei Cluster:

```
# Setze Seed

set.seed(1323)


# k-means Clusteranalyse mit drei Clustern:
```

```r
# kmeans(zu clusternde Daten, Clusterzahl)
kmeans_Ergebnis <- kmeans(Isotopen_skaliert, 3)
kmeans_Ergebnis

# K-means clustering with 3 clusters of sizes 18, 10, 18
#
# Cluster means:
#   d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# 1   -1.1304947   -0.7954053   -0.9638108   -0.6187481
# 2    0.6879384    0.5649727    0.8789997   -0.7778869
# 3    0.7483067    0.4815316    0.4754777    1.0509075
#
# Clustering vector:
#  [1] 1 1 1 1 2 3 3 3 2 3 3 3 2 2 2 3 3 1 3 3 1 1 1 3 1 1 1 1 3 3 3 2 2
# [34] 2 1 2 2 3 3 3 3 1 1 1 1 1
#
# Within cluster sum of squares by cluster:
# [1] 26.414874  9.837903 25.838994
#  (between_SS / total_SS =  65.5 %)
#
# Available components:
#
# [1] "cluster"      "centers"      "totss"        "withinss"
# [5] "tot.withinss" "betweenss"    "size"         "iter"
# [9] "ifault"

# Clusterzentren
kmeans_Clusterzentren <- kmeans_Ergebnis$centers
```

```
# Übertragen der Clusterzuweisungen in die Datentabelle
kmeans_3_Cluster <- kmeans_Ergebnis$cluster
Daten <- cbind(Daten, kmeans_3_Cluster)
```

4) Graphische Darstellung

Die Ergebnisse der Clusteranalyse können graphisch dargestellt werden.
Beispielhaft sei ein $\delta^{13}C_{Kollagen}$-$\delta^{15}N_{Kollagen}$-Graph und ein $\delta^{13}C_{Karbonat}$-$\delta^{18}O_{Karbonat}$-Graph dargestellt.

Die R-Pakete

- „ggplot2“ (Version 3.0.0; Wickham, 2016)
- „ggpubr“ (Version 0.1.8; Kassambara, 2018)

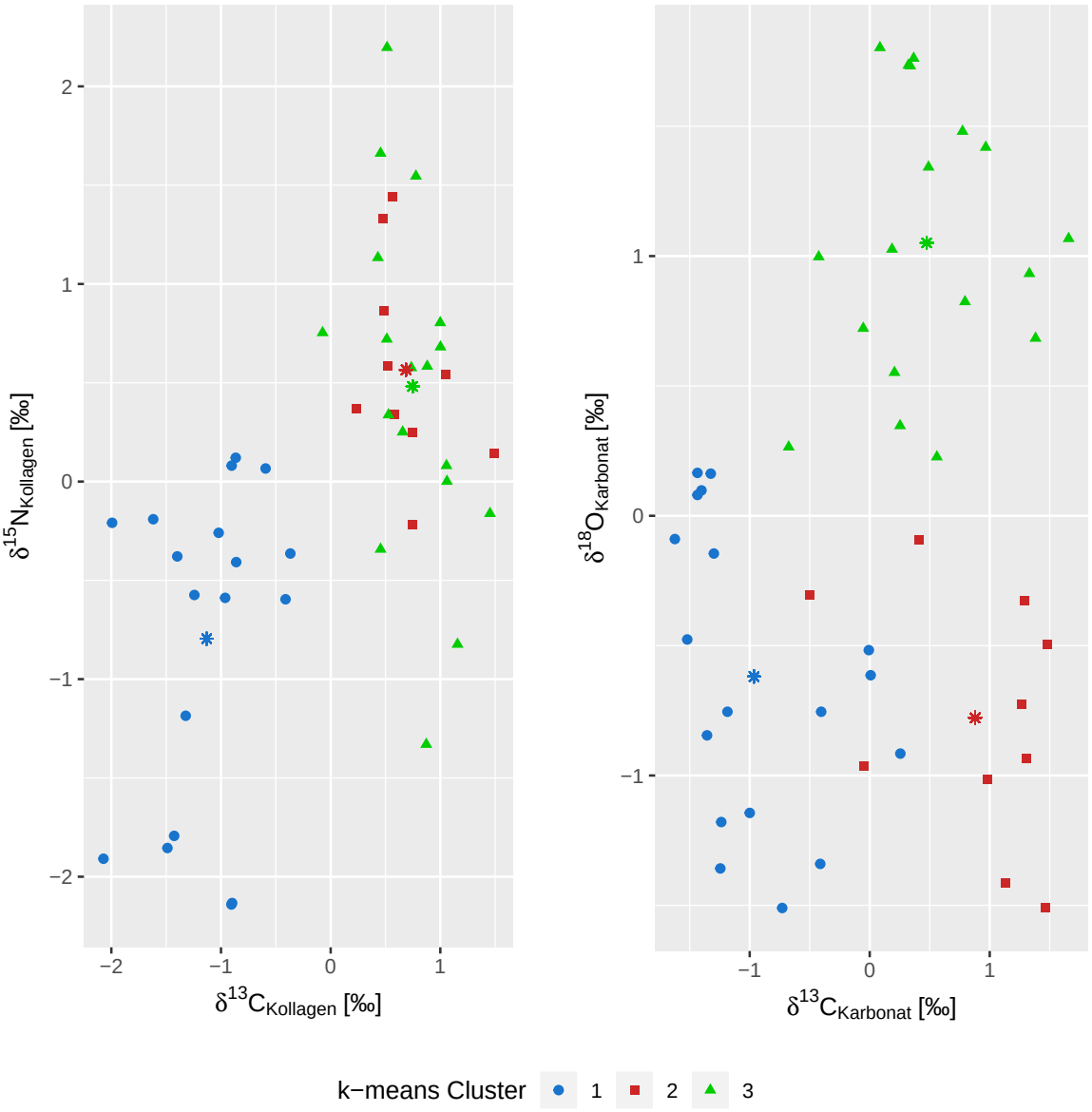
müssen installiert sein.

```
# install.packages(c("ggplot2", "ggpubr"))
library(ggplot2)
library(ggpubr)
```

```
# Auswahl der Farbkodierung
Farben <- c("dodgerblue3", "firebrick3", "green3")

# Auswahl der Symbole
Symbole <- c(19, 15, 17)

# Erstellen der Graphen
Clusterergebnis <- data.frame(Isotopen_skaliert[, 1:4],
                              Cluster = factor(kmeans_Ergebnis$cluster))
```

```
d13CKoll_d15NKoll <- ggplot(Clusterergebnis)+
  scale_color_manual(values = Farben, name = "k-means Cluster")+
  scale_shape_manual(values = Symbole, name = "k-means Cluster")+
  geom_point(aes(x = Clusterergebnis[, 1],
                 y = Clusterergebnis[, 2],
                 colour = Cluster,
                 shape = Cluster))+
  geom_point(aes(x = kmeans_Clusterzentren[1, 1],
                 y = kmeans_Clusterzentren[1, 2]),
             color = "dodgerblue3", shape = 8)+
  geom_point(aes(x = kmeans_Clusterzentren[2, 1],
                 y = kmeans_Clusterzentren[2, 2]),
             color = "firebrick3", shape = 8)+
  geom_point(aes(x = kmeans_Clusterzentren[3, 1],
                 y = kmeans_Clusterzentren[3, 2]),
             color = "green3", shape = 8)+
  xlab(expression(paste(delta^13*"C"[Kollagen]~"[\211]")))+
  ylab(expression(paste(delta^15*"N"[Kollagen]~"[\211]")))+
  theme(plot.margin = margin(0.5, 0.5, 0.5, 0.5, "cm"))


d13CKarb_d18OKarb <- ggplot(Clusterergebnis)+
  scale_color_manual(values = Farben, name = "k-means Cluster")+
  scale_shape_manual(values = Symbole, name = "k-means Cluster")+
  geom_point(aes(x = Clusterergebnis[, 3],
                 y = Clusterergebnis[, 4],
                 colour = Cluster,
                 shape = Cluster))+
  geom_point(aes(x = kmeans_Clusterzentren[1, 3],
```

```
                y = kmeans_Clusterzentren[1, 4]),
            color = "dodgerblue3", shape = 8)+
  geom_point(aes(x = kmeans_Clusterzentren[2, 3],
                y = kmeans_Clusterzentren[2, 4]),
            color = "firebrick3", shape = 8)+
  geom_point(aes(x = kmeans_Clusterzentren[3, 3],
                y = kmeans_Clusterzentren[3, 4]),
            color = "green3", shape = 8)+
  xlab(expression(paste(delta^13*"C"[Karbonat]~"[\211]")))+
  ylab(expression(paste(delta^18*"O"[Karbonat]~"[\211]")))+
  theme(plot.margin = margin(0.5, 0.5, 0.5, 0.5, "cm"))

# Zusammenführen der beiden einzelnen Graphen in eine Gesamtgraphik
# pdf(width = 12, height = 6, "kmeans_Plot.pdf",
# encoding = "MacRoman")
ggarrange(d13CKoll_d15NKoll, d13CKarb_d18OKarb,
          ncol = 2, nrow = 1,
          common.legend = TRUE, legend = "bottom")
```


δ15NKollagen [‰]
δ13CKollagen [‰]
δ18OKarbonat [‰]
δ13CKarbonat [‰]
k−means Cluster
1
2
3

```
# dev.off()
```

4.2 „Gaussian Mixture Model“ (GMM)

Das R-Paket

- „mclust“ (Version 5.4.1; Scrucca et al., 2016)

muss installiert sein.

```
# install.packages("mclust")
library(mclust)
```

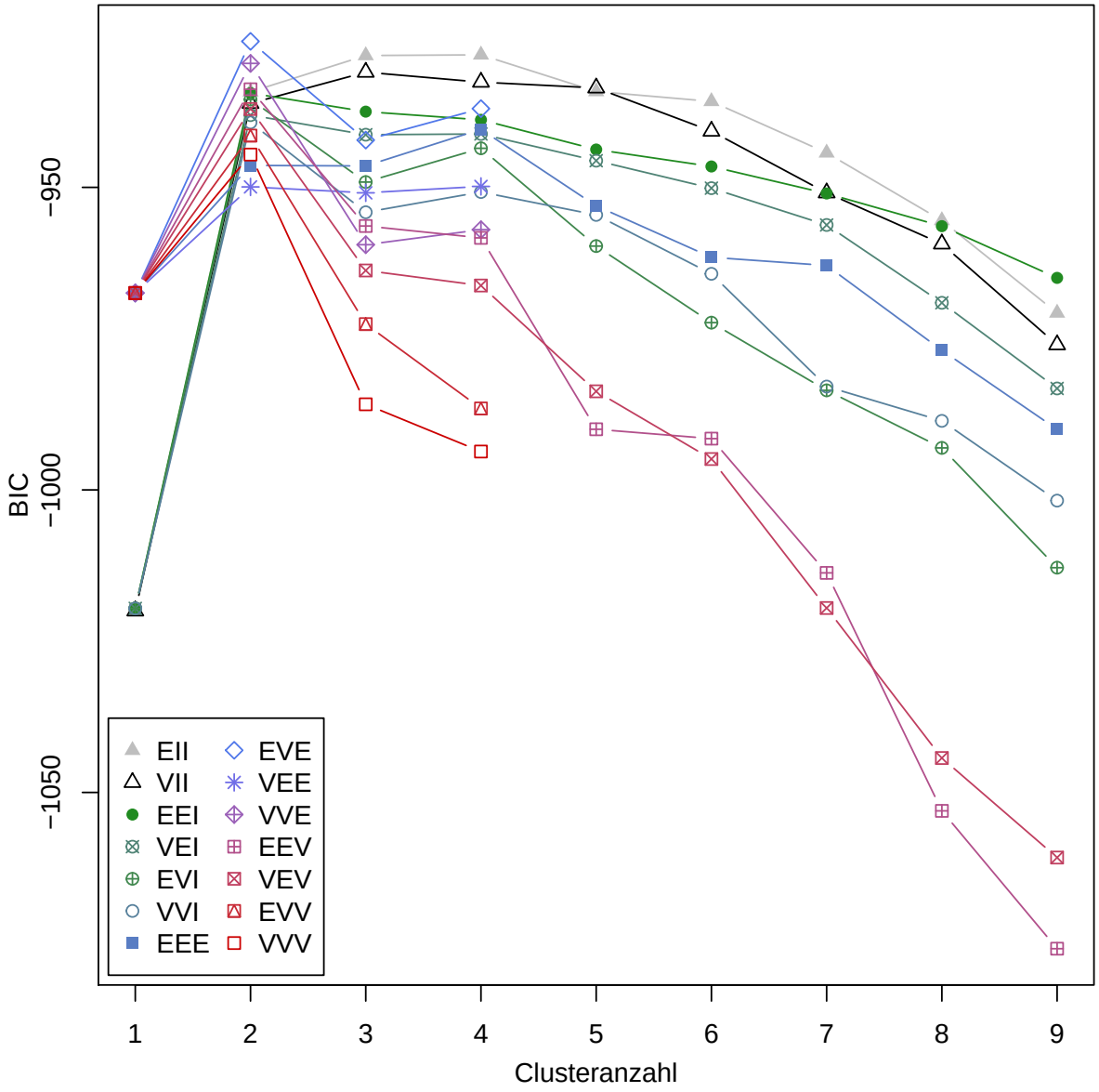
```
# Einlesen der originalen (nicht-transformierten) Daten
mclust.options(hcUSE = "VARS")

# Übersicht über die Farbkodierung der Cluster
mclust.options("classPlotColors")

#  [1] "dodgerblue2"    "red3"           "green3"
#  [4] "slateblue"      "darkorange"     "skyblue1"
#  [7] "violetred4"     "forestgreen"    "steelblue4"
# [10] "slategrey"      "brown"          "black"
# [13] "darkseagreen"   "darkgoldenrod3" "olivedrab"
# [16] "royalblue"      "tomato4"        "cyan2"
# [19] "springgreen2"
```

1) Finden des optimalen Clustermodelles mittels BIC

Hierbei gilt zu beachten, dass im „mclust“-Paket das BIC so definiert ist, dass ein möglichst hoher Wert optimal ist (vgl. Scrucca et al., 2016).

```
GMM_BIC <- mclustBIC(Daten[, 6:9])
GMM_BIC


# Bayesian Information Criterion (BIC):
#          EII        VII        EEI        VEI        EVI        VVI
# 1 -1020.0017 -1020.0017 -1019.5856 -1019.5856 -1019.5856 -1019.5856
# 2  -934.2243  -936.1192  -934.4271  -938.0628  -935.4788  -939.2755
# 3  -928.2114  -930.9668  -937.4858  -941.2978  -949.1240  -954.1142
# 4  -928.0991  -932.6159  -938.8373  -941.1813  -943.5425  -950.7692
# 5  -934.1585  -933.5647  -943.7463  -945.5910  -959.7050  -954.5538
# 6  -935.7948  -940.7120  -946.5475  -950.1340  -972.3566  -964.2881
# 7  -944.2998  -950.9053  -951.0099  -956.2175  -983.5479  -982.9160
# 8  -955.4748  -959.3481  -956.4131  -969.0800  -993.0579  -988.5917
# 9  -970.7602  -976.0112  -964.9602  -983.2341 -1012.8528 -1001.7716
#         EEE       EVE       VEE       VVE        EEV        VEV
# 1 -967.4610 -967.4610 -967.4610 -967.4610  -967.4610  -967.4610
# 2 -946.3749 -925.8684 -949.9262 -929.5119  -933.7912  -937.0543
# 3 -946.4602 -942.1890 -950.9040 -959.4777  -956.3856  -963.7354
# 4 -940.4060 -936.9863 -949.8452 -956.9938  -958.3482  -966.2228
# 5 -953.0609        NA        NA        NA  -989.9776  -983.7066
# 6 -961.5614        NA        NA        NA  -991.5231  -994.8997
# 7 -962.9094        NA        NA        NA -1013.7146 -1019.5182
# 8 -976.9280        NA        NA        NA -1053.0440 -1044.3080
# 9 -989.9938        NA        NA        NA -1075.8237 -1060.7615
#         EVV       VVV
# 1 -967.4610 -967.4610
# 2 -941.4376 -944.5805
# 3 -972.5958 -985.8519
# 4 -986.5297 -993.6580
```

```
# 5         NA         NA
# 6         NA         NA
# 7         NA         NA
# 8         NA         NA
# 9         NA         NA
#
# Top 3 models based on the BIC criterion:
#     EVE,2     EII,4     EII,3
# -925.8684 -928.0991 -928.2114

GMM <- Mclust(Daten[, 6:9])


# pdf(width = 6, height = 6, "GMM_BIC_Plot.pdf",
# encoding = "MacRoman")
par(mar = c(3.5, 3, 1, 1))
plot(GMM, what = "BIC", legendArgs = list(x = "bottomleft",
                                          ncol = 2, cex = 1),
     xlab = "",ylab = "")
mtext(side = 1,line = 2.2, "Clusteranzahl")
mtext(side = 2,line = 2, "BIC")
```




```
# dev.off()
```

2) Clusteranalyse

```
# GMM-Clusteranalyse mit optimaler Clusterzahl, ohne Modell-Vorgabe

GMM_Cluster_optimal <- Mclust(Daten[, 6:9])

summary(GMM_Cluster_optimal)


# ----------------------------------------------------
```

```
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust EVE (ellipsoidal, equal volume and orientation) model with 2
# components:
#
#  log.likelihood  n df       BIC       ICL
#       -420.8191 46 22 -925.8684 -925.8813
#
# Clustering table:
#  1  2
# 18 28

# Clusterzuordnungen
GMM_Cluster_optimal$classification

#  [1] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 1 1 2 1 1 1 1 2 2 2 2 2
# [34] 2 1 2 2 2 2 2 2 1 1 1 1 1

# Zuordnungswahrscheinlichkeiten
GMM_Cluster_optimal$z

#              [,1]         [,2]
#  [1,] 1.000000e+00 1.484513e-40
#  [2,] 1.000000e+00 1.152258e-26
#  [3,] 1.000000e+00 1.796023e-17
#  [4,] 1.000000e+00 3.122025e-23
#  [5,] 4.268564e-06 9.999957e-01
#  [6,] 1.047054e-16 1.000000e+00
#  [7,] 4.514963e-19 1.000000e+00
#  [8,] 1.510164e-13 1.000000e+00
```

```
#  [9,] 1.660055e-11 1.000000e+00
# [10,] 9.766734e-21 1.000000e+00
# [11,] 1.328412e-13 1.000000e+00
# [12,] 7.399168e-20 1.000000e+00
# [13,] 1.495279e-19 1.000000e+00
# [14,] 8.897608e-16 1.000000e+00
# [15,] 3.379608e-03 9.966204e-01
# [16,] 5.747756e-10 1.000000e+00
# [17,] 5.399609e-07 9.999995e-01
# [18,] 1.000000e+00 7.727219e-13
# [19,] 7.936535e-10 1.000000e+00
# [20,] 2.785226e-03 9.972148e-01
# [21,] 1.000000e+00 5.685240e-16
# [22,] 1.000000e+00 4.433959e-14
# [23,] 9.999990e-01 9.788894e-07
# [24,] 7.552446e-09 1.000000e+00
# [25,] 9.999999e-01 1.004848e-07
# [26,] 1.000000e+00 5.298758e-13
# [27,] 9.999265e-01 7.345255e-05
# [28,] 1.000000e+00 1.038074e-16
# [29,] 5.900141e-18 1.000000e+00
# [30,] 7.444150e-06 9.999926e-01
# [31,] 9.036872e-17 1.000000e+00
# [32,] 4.310574e-10 1.000000e+00
# [33,] 3.055047e-16 1.000000e+00
# [34,] 9.270898e-18 1.000000e+00
# [35,] 1.000000e+00 1.292451e-22
# [36,] 8.349917e-11 1.000000e+00
```

```
# [37,] 1.762043e-16 1.000000e+00
# [38,] 2.245318e-20 1.000000e+00
# [39,] 2.179651e-25 1.000000e+00
# [40,] 5.924153e-15 1.000000e+00
# [41,] 1.032921e-12 1.000000e+00
# [42,] 1.000000e+00 1.366779e-09
# [43,] 9.998342e-01 1.657554e-04
# [44,] 9.999920e-01 8.000303e-06
# [45,] 1.000000e+00 8.003560e-10
# [46,] 1.000000e+00 2.809256e-08

# Clusterparameter
GMM_Cluster_optimal$parameter

# $pro
# [1] 0.3914362 0.6085638
#
# $mean
#                     [,1]        [,2]
# d13CKollagen -23.436419 -14.9704675
# d15NKollagen   9.099615  12.7106729
# d13CKarbonat  -5.370584   0.7734926
# d18OKarbonat -12.025995  -8.5557515
#
# $variance
# $variance$modelName
# [1] "EVE"
#
# $variance$d
# [1] 4
```

```
#
# $variance$G
# [1] 2
#
# $variance$sigma
# , , 1
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen    6.3798882    0.8481878   -1.0548219   -0.5540655
# d15NKollagen    0.8481878    3.6147414   -1.8494357   -0.3474466
# d13CKarbonat   -1.0548219   -1.8494357    3.5486737   -0.7750772
# d18OKarbonat   -0.5540655   -0.3474466   -0.7750772    5.2676997
#
# , , 2
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen    2.4052396  -1.65502352   1.82698293   -0.2919339
# d15NKollagen   -1.6550235   4.73261565  -0.02821038    1.2721993
# d13CKarbonat    1.8269829  -0.02821038   5.70923761   -2.9303527
# d18OKarbonat   -0.2919339   1.27219927  -2.93035272   10.8634425
#
#
# $variance$scale
# [1] 4.063491
#
# $variance$shape
#           [,1]     [,2]
# [1,] 1.8139561 0.199586
```

```
# [2,] 0.3789352 1.220431
# [3,] 1.0473950 1.331346
# [4,] 1.3889857 3.083653
#
# $variance$orientation
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen    0.8237167   0.06826161    0.5479821   -0.1286340
# d15NKollagen    0.3897585   0.65238955   -0.6265309    0.1730181
# d13CKarbonat   -0.3878251   0.72172372    0.3956704   -0.4149114
# d18OKarbonat   -0.1384585   0.22100473    0.3880974    0.8839494
#
#
# $Vinv
# NULL

# Übertragen der Clusterzuweisungen in die Datentabelle
GMM_optimal <- GMM_Cluster_optimal$classification
Daten <- cbind(Daten, GMM_optimal)


# GMM-Clusteranalyse mit optimaler Clusterzahl, Modell: EII
GMM_Cluster_optimal_EII <- Mclust(Daten[, 6:9], modelNames = "EII")
summary(GMM_Cluster_optimal_EII)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust EII (spherical, equal volume) model with 4 components:
#
#  log.likelihood  n df       BIC     ICL
```

```
#       -425.7632 46 20 -928.0991 -932.35
#
# Clustering table:
#  1  2  3  4
#  7 13 18  8

# Clusterzuordnungen
GMM_Cluster_optimal_EII$classification

#  [1] 1 1 1 1 4 3 3 3 4 3 3 3 4 4 2 3 3 2 3 2 2 2 2 3 2 1 2 1 3 3 3 3 4
# [34] 4 1 4 4 3 3 3 3 2 2 2 2 2

# Zuordnungswahrscheinlichkeiten
GMM_Cluster_optimal_EII$z

#               [,1]         [,2]         [,3]         [,4]
#  [1,] 9.999751e-01 2.489743e-05 2.200102e-16 1.970435e-14
#  [2,] 9.979790e-01 2.021037e-03 6.693109e-12 2.001355e-11
#  [3,] 9.930940e-01 6.906007e-03 3.645547e-09 1.927014e-08
#  [4,] 9.326369e-01 6.736307e-02 5.020734e-16 3.038763e-13
#  [5,] 6.670352e-08 9.795831e-04 1.244113e-02 9.865792e-01
#  [6,] 2.461332e-16 9.639008e-09 9.999990e-01 1.009808e-06
#  [7,] 1.788422e-18 5.960242e-11 9.999975e-01 2.537000e-06
#  [8,] 7.096602e-15 2.609030e-09 9.999978e-01 2.213233e-06
#  [9,] 2.540831e-10 2.144116e-08 7.555965e-04 9.992444e-01
# [10,] 4.797492e-18 4.618798e-10 9.999968e-01 3.209515e-06
# [11,] 2.921853e-12 2.146005e-08 9.967014e-01 3.298583e-03
# [12,] 2.911358e-18 3.529696e-14 9.950645e-01 4.935455e-03
# [13,] 1.195086e-13 4.237694e-10 2.823570e-02 9.717643e-01
# [14,] 1.967863e-16 1.328373e-13 6.658588e-04 9.993341e-01
# [15,] 2.915225e-05 7.145015e-01 2.230731e-01 6.239625e-02
```

```
# [16,] 1.832139e-11 3.043658e-08 9.290936e-01 7.090638e-02
# [17,] 1.822917e-08 5.688654e-05 9.957791e-01 4.163980e-03
# [18,] 3.868702e-03 9.961313e-01 1.342356e-10 1.434832e-13
# [19,] 2.214175e-12 1.311947e-07 9.813946e-01 1.860524e-02
# [20,] 5.925634e-06 8.937551e-01 1.057855e-01 4.534838e-04
# [21,] 1.585069e-02 9.841493e-01 1.163816e-13 8.668675e-15
# [22,] 4.781858e-03 9.952181e-01 2.513192e-11 5.054670e-14
# [23,] 1.420902e-04 9.998579e-01 4.712717e-08 7.556436e-11
# [24,] 1.234377e-11 9.503778e-05 9.998466e-01 5.839656e-05
# [25,] 2.294614e-03 9.977054e-01 9.178947e-10 1.018803e-09
# [26,] 6.002058e-01 3.997942e-01 1.520031e-12 1.269416e-12
# [27,] 4.301418e-02 9.568411e-01 1.946592e-05 1.252409e-04
# [28,] 7.570989e-01 2.429011e-01 2.321158e-13 4.548524e-12
# [29,] 3.662055e-17 2.036760e-12 9.999180e-01 8.204711e-05
# [30,] 4.974428e-09 1.473384e-05 9.994492e-01 5.360250e-04
# [31,] 5.640196e-15 5.805010e-10 9.999595e-01 4.052298e-05
# [32,] 2.278912e-09 1.912998e-05 6.022663e-01 3.977145e-01
# [33,] 1.160130e-12 7.185083e-11 7.505878e-03 9.924941e-01
# [34,] 3.337252e-13 2.407696e-11 6.479901e-06 9.999935e-01
# [35,] 9.976698e-01 2.330227e-03 1.062597e-14 3.029500e-12
# [36,] 1.799937e-10 1.243186e-09 4.010188e-05 9.999599e-01
# [37,] 1.654603e-14 1.164526e-11 2.632318e-03 9.973677e-01
# [38,] 1.302793e-15 7.464618e-12 9.635825e-01 3.641755e-02
# [39,] 1.708692e-18 2.652995e-14 9.942031e-01 5.796899e-03
# [40,] 4.151804e-15 1.431769e-09 9.999521e-01 4.788794e-05
# [41,] 1.996128e-15 4.371089e-09 9.998112e-01 1.887887e-04
# [42,] 1.116974e-03 9.988830e-01 1.431239e-10 5.177560e-13
# [43,] 1.800383e-04 9.998191e-01 8.512774e-07 9.959584e-09
```

```
# [44,] 2.015935e-03 9.979840e-01 6.295606e-10 3.432497e-08
# [45,] 2.208873e-02 9.779113e-01 1.211233e-11 1.963783e-10
# [46,] 6.303860e-04 9.993696e-01 7.945865e-09 1.449502e-11

# Clusterparameter
GMM_Cluster_optimal_EII$parameter

# $pro
# [1] 0.1385800 0.2877094 0.3877864 0.1859242
#
# $mean
#                    [,1]      [,2]       [,3]       [,4]
# d13CKollagen -24.978773 -22.03463 -14.762771 -14.836255
# d15NKollagen   6.923200  10.60949  12.617039  12.868637
# d13CKarbonat  -3.149788  -6.26835   0.377426   2.485306
# d18OKarbonat -13.250757 -11.17577  -6.380347 -12.845314
#
# $variance
# $variance$modelName
# [1] "EII"
#
# $variance$d
# [1] 4
#
# $variance$G
# [1] 4
#
# $variance$sigma
# , , 1
#
```

```
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     3.269704     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     3.269704     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     3.269704     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     3.269704
#
# , , 2
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     3.269704     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     3.269704     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     3.269704     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     3.269704
#
# , , 3
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     3.269704     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     3.269704     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     3.269704     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     3.269704
#
# , , 4
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     3.269704     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     3.269704     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     3.269704     0.000000
```

```
# d18OKarbonat     0.000000     0.000000     0.000000     3.269704
#
#
# $variance$Sigma
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     3.269704     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     3.269704     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     3.269704     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     3.269704
#
# $variance$sigmasq
# [1] 3.269704
#
# $variance$scale
# [1] 3.269704

# Übertragen der Clusterzuweisungen in die Datentabelle
GMM_optimal_EII <- GMM_Cluster_optimal_EII$classification
Daten <- cbind(Daten, GMM_optimal_EII)


# GMM-Clusteranalyse mit vorgegebener Clusterzahl (G = 3) und
# Modell "EII"
GMM_Cluster_3_EII <- Mclust(Daten[, 6:9], G = 3, modelNames = "EII")
summary(GMM_Cluster_3_EII)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust EII (spherical, equal volume) model with 3 components:
```

```
#
#  log.likelihood  n df       BIC       ICL
#       -435.3909 46 15 -928.2114 -930.8051
#
# Clustering table:
#  1  2  3
# 18 20  8

# Clusterzuordnungen
GMM_Cluster_3_EII$classification

#  [1] 1 1 1 1 3 2 2 2 3 2 2 2 3 3 2 2 2 1 2 2 1 1 1 2 1 1 1 1 2 2 2 2 3
# [34] 3 1 3 3 2 2 2 2 1 1 1 1 1

# Zuordnungswahrscheinlichkeiten
GMM_Cluster_3_EII$z

#              [,1]         [,2]         [,3]
#  [1,] 1.000000e+00 5.125862e-11 4.414901e-10
#  [2,] 1.000000e+00 1.963966e-08 1.382931e-08
#  [3,] 9.999964e-01 1.580686e-06 2.053929e-06
#  [4,] 1.000000e+00 2.652741e-12 6.308368e-11
#  [5,] 6.206387e-04 7.325700e-02 9.261224e-01
#  [6,] 4.277687e-09 9.999861e-01 1.393378e-05
#  [7,] 8.012229e-11 9.999658e-01 3.416348e-05
#  [8,] 6.761650e-09 9.999687e-01 3.128988e-05
#  [9,] 4.718841e-07 6.065758e-03 9.939338e-01
# [10,] 2.801276e-10 9.999638e-01 3.617409e-05
# [11,] 9.695972e-08 9.911872e-01 8.812670e-03
# [12,] 2.299298e-12 9.807913e-01 1.920869e-02
# [13,] 5.634551e-09 8.282648e-02 9.171735e-01
```

```
# [14,] 1.334836e-11 3.752780e-03 9.962472e-01
# [15,] 1.228896e-01 7.674036e-01 1.097068e-01
# [16,] 1.851461e-07 9.013303e-01 9.866947e-02
# [17,] 6.395317e-05 9.914119e-01 8.524098e-03
# [18,] 9.999999e-01 5.201538e-08 5.406395e-11
# [19,] 1.991394e-07 9.690476e-01 3.095219e-02
# [20,] 1.776066e-01 8.181763e-01 4.217098e-03
# [21,] 1.000000e+00 1.495823e-10 3.278765e-12
# [22,] 1.000000e+00 1.334354e-08 2.120340e-11
# [23,] 9.999847e-01 1.530437e-05 2.424852e-08
# [24,] 8.980077e-06 9.997407e-01 2.502761e-04
# [25,] 9.999996e-01 3.485690e-07 8.038027e-08
# [26,] 1.000000e+00 5.591747e-10 1.081929e-10
# [27,] 9.991136e-01 3.723971e-04 5.139963e-04
# [28,] 1.000000e+00 1.695457e-10 3.451512e-10
# [29,] 3.947732e-11 9.993171e-01 6.828794e-04
# [30,] 2.469683e-05 9.980333e-01 1.941972e-03
# [31,] 2.791063e-09 9.996891e-01 3.108695e-04
# [32,] 1.762522e-05 7.074188e-01 2.925636e-01
# [33,] 4.529573e-09 2.668285e-02 9.733171e-01
# [34,] 2.090371e-09 1.505934e-04 9.998494e-01
# [35,] 1.000000e+00 1.314310e-10 2.515238e-09
# [36,] 9.982325e-08 5.844254e-04 9.994155e-01
# [37,] 5.102236e-10 1.231973e-02 9.876803e-01
# [38,] 1.761801e-10 9.292710e-01 7.072899e-02
# [39,] 1.638146e-12 9.793579e-01 2.064210e-02
# [40,] 3.887115e-09 9.996603e-01 3.397215e-04
# [41,] 5.014949e-09 9.990856e-01 9.144179e-04
```

```
# [42,] 9.999999e-01 8.358511e-08 2.199479e-10
# [43,] 9.998540e-01 1.448062e-04 1.221084e-06
# [44,] 9.999983e-01 3.023161e-07 1.406896e-06
# [45,] 1.000000e+00 6.127849e-09 1.030889e-08
# [46,] 9.999977e-01 2.330205e-06 4.055365e-09

# Clusterparameter
GMM_Cluster_3_EII$parameter

# $pro
# [1] 0.3978299 0.4181953 0.1839748
#
# $mean
#                    [,1]         [,2]      [,3]
# d13CKollagen -23.349702 -14.99451326 -14.80911
# d15NKollagen   9.161648  12.65560841  12.82719
# d13CKarbonat  -5.348149   0.08836789   2.49587
# d18OKarbonat -11.990234  -6.66643539 -12.80711
#
# $variance
# $variance$modelName
# [1] "EII"
#
# $variance$d
# [1] 4
#
# $variance$G
# [1] 3
#
# $variance$sigma
```

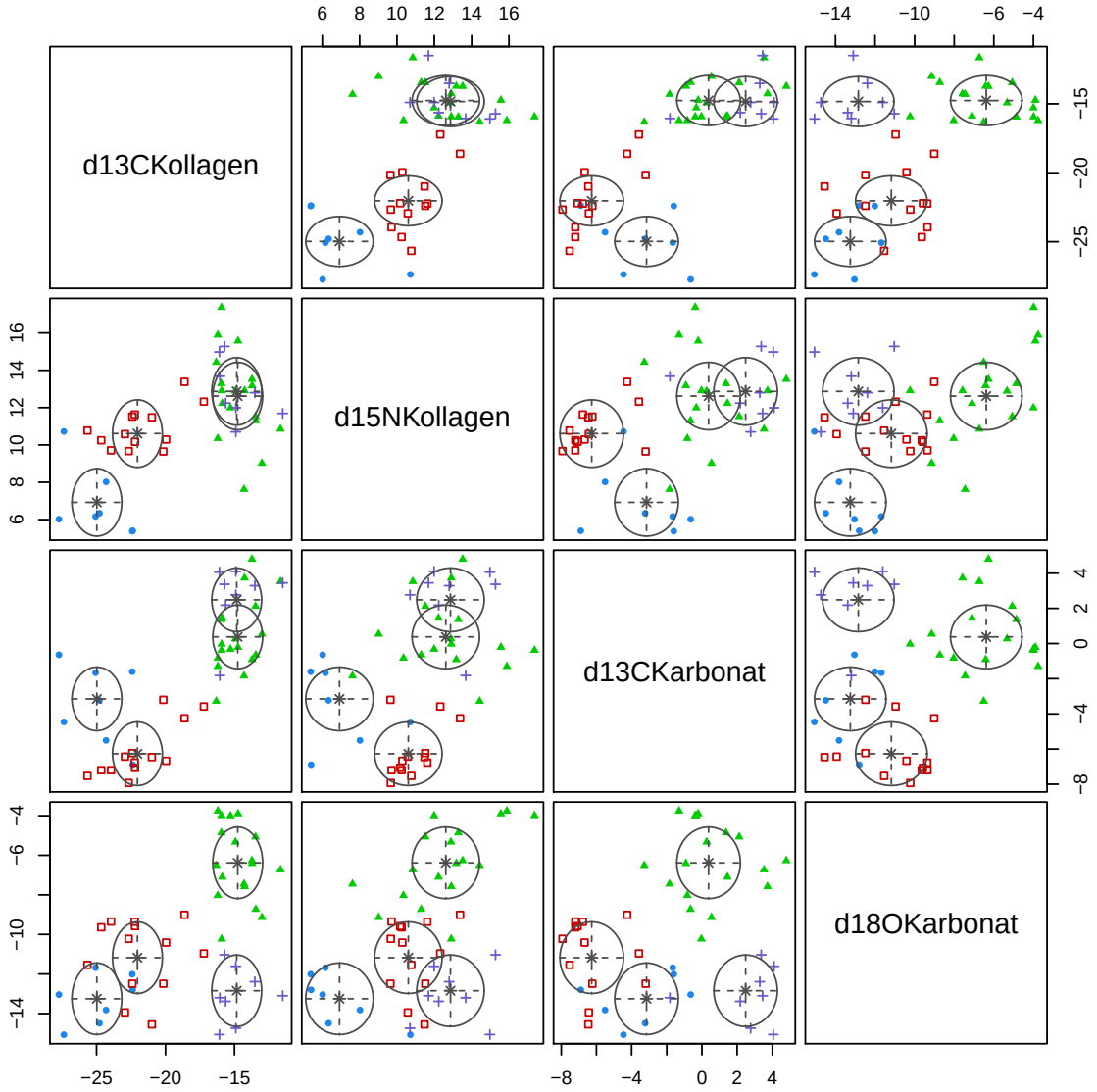
```
# , , 1
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     4.108899     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     4.108899     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     4.108899     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     4.108899
#
# , , 2
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     4.108899     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     4.108899     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     4.108899     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     4.108899
#
# , , 3
#
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     4.108899     0.000000     0.000000     0.000000
# d15NKollagen     0.000000     4.108899     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     4.108899     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     4.108899
#
#
# $variance$Sigma
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     4.108899     0.000000     0.000000     0.000000
```

```
# d15NKollagen     0.000000     4.108899     0.000000     0.000000
# d13CKarbonat     0.000000     0.000000     4.108899     0.000000
# d18OKarbonat     0.000000     0.000000     0.000000     4.108899
#
# $variance$sigmasq
# [1] 4.108899
#
# $variance$scale
# [1] 4.108899
```

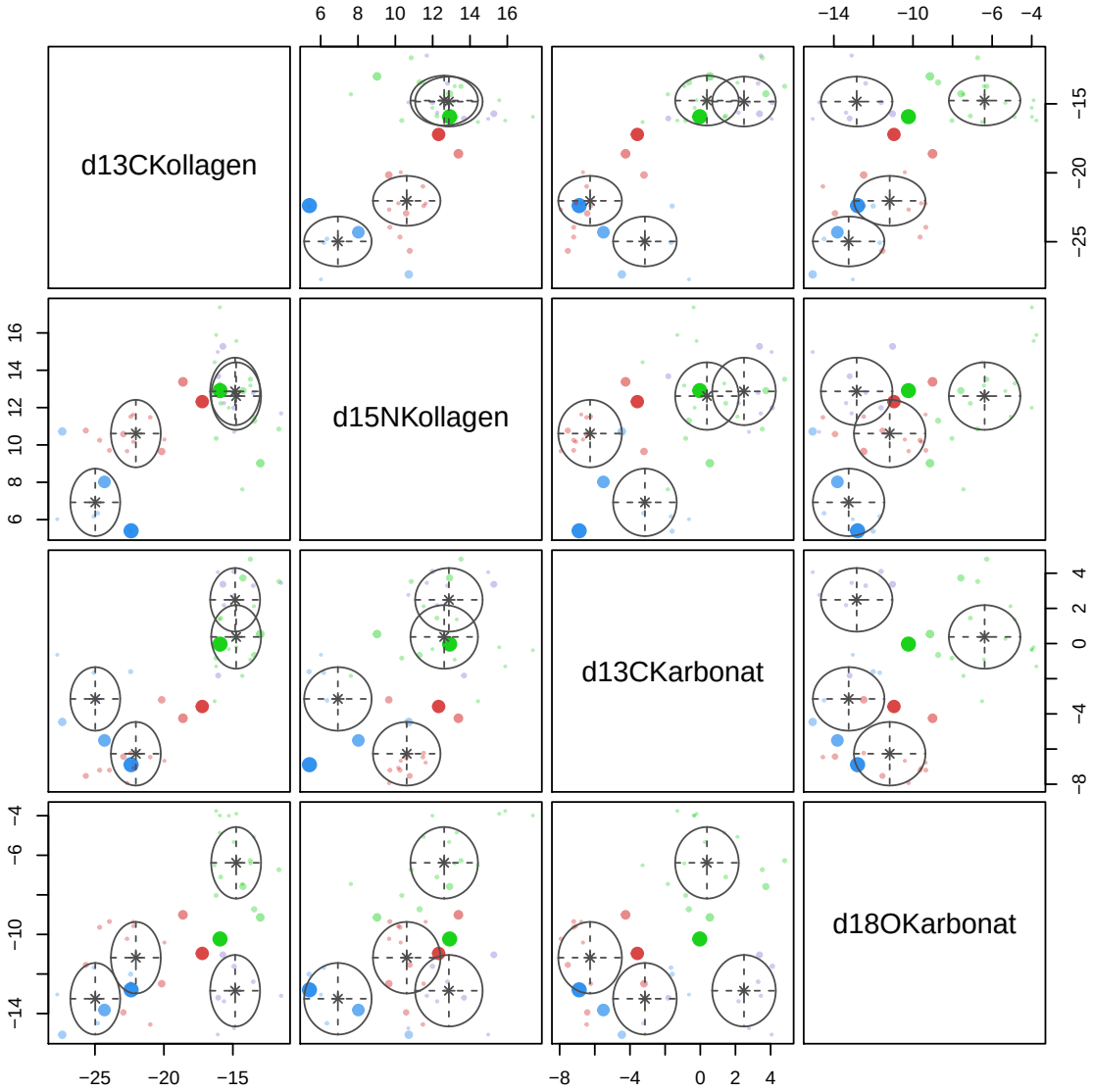
```
# Übertragen der Clusterzuweisungen in die Datentabelle
GMM_3Cluster_EII <- GMM_Cluster_3_EII$classification
Daten <- cbind(Daten, GMM_3Cluster_EII)
```

3) Graphische Darstellung

```
# Graphische Darstellung des optimalen Clusters ("EII"-Modell)
# pdf(width = 12, height = 12, "GMM_Classification_Plot.pdf",
# encoding = "MacRoman")
plot(GMM_Cluster_optimal_EII, what = "classification", CEX = 0.8)
```




```
# dev.off()


# pdf(width = 12, height = 12, "GMM_Uncertainty_Plot.pdf",
# encoding = "MacRoman")
plot(GMM_Cluster_optimal_EII, what = "uncertainty", CEX = 0.8)
```


d13CKollagen
d15NKollagen
d13CKarbonat
d18OKarbonat
6 8 10 12 14 16
−14 −10 −6 −4
−15 −20 −25
6 8 10 12 14 16
4 2 0 −4 −8
−4 −6 −10 −14
−25 −20 −15
−8 −4 0 2 4

```
# dev.off()
```

5 Diskriminanzanalyse

1) Erzeugen eines Matrixeintrags mit Clusterzuweisung

vgl. Kapitel 4.2

2) Trainings- und Testdatensatz

```
TrainTest <- matrix(NA, ncol = 2, nrow = 200)
colnames(TrainTest) <- c("Trainingsfehler", "Testfehler")
for (i in 1:200){
  n <- floor(2/3 * nrow(Daten))
  Training_Individuen <- sample(seq_len(nrow(Daten)), size = n)
  # Trainingsdaten: Spalten mit Isotopendaten
  Training <- Daten[Training_Individuen, 6:9]
  # Testdaten: Spalten mit Isotopendaten
  Test <- Daten[-Training_Individuen, 6:9]
  # Trainingsdaten: Spalten mit Klassenzuweisung
  # (hier: GMM-Clusterergebnis)
  Training_Label <- Daten[Training_Individuen, ]$GMM_3Cluster_EII
  # Testdaten: Spalten mit Klassenzuweisung
  # (hier: GMM-Clusterergebnis)
  Test_Label <- Daten[-Training_Individuen, ]$GMM_3Cluster_EII
  tryCatch({
    MclustDA <- MclustDA(Training, Training_Label, modelNames = "EII",
                         G = 3)
    summary <- summary(MclustDA, newdata = Test, newclass = Test_Label)
    summary$err
    summary$err.newdata
    TrainTest[i, "Trainingsfehler"] <- summary$err
```

```
    TrainTest[i, "Testfehler"] <- summary$err.newdata
  }, error = function(e){})
}

# In Fällen, in denen der Trainingsdatensatz nicht alle
# verfügbaren Klassen (hier: 3) beinhaltet, wird ein "NA"
# erzeugt
# Entfernen der "NA"-Einträge
TrainTest_ohneNA <- na.omit(TrainTest)

# Mittlerer Trainingsfehler der ersten 100 Einträge
mean(TrainTest_ohneNA[1:100, 1])

# [1] 0

# Mittlerer Testfehler der ersten 100 Einträge
mean(TrainTest_ohneNA[1:100, 2])

# [1] 0.0875
```

6 Hauptkomponentenanalyse (PCA)

1) Berechnung der Hautkomponenten

```
pca <- prcomp(Daten[, 6:9], scale = FALSE)
summary(pca)

# Importance of components:
#                           PC1    PC2     PC3    PC4
# Standard deviation     6.0888 3.2273 2.13180 1.7619
# Proportion of Variance 0.6724 0.1889 0.08242 0.0563
# Cumulative Proportion  0.6724 0.8613 0.94370 1.0000

pca

# Standard deviations (1, .., p=4):
# [1] 6.088832 3.227315 2.131796 1.761901
#
# Rotation (n x k) = (4 x 4):
#                    PC1         PC2         PC3        PC4
# d13CKollagen 0.7207277 -0.06752277  0.03626097  0.6889684
# d15NKollagen 0.2953285  0.26595493  0.85701605 -0.3279826
# d13CKarbonat 0.5346560 -0.54266014 -0.24530028 -0.5995754
# d18OKarbonat 0.3278348  0.79386940 -0.45170388 -0.2413697
```

2) Graphische Darstellung

Die R-Pakete

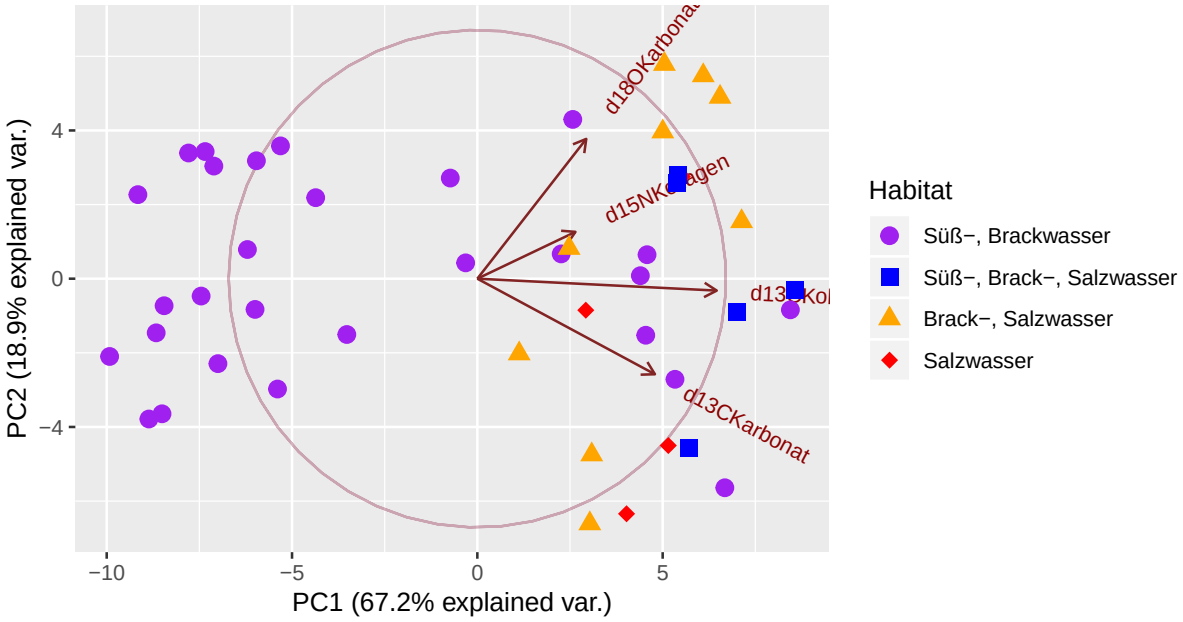
- „devtools“ (Version 2.0.1; Wickham et al., 2018a)
- „ggbiplot“ (Version 0.55; Vu, 2011)

müssen installiert sein.

```
# install.packages("devtools")
library(devtools)


# install_github("vqv/ggbiplot")
library(ggbiplot)
```

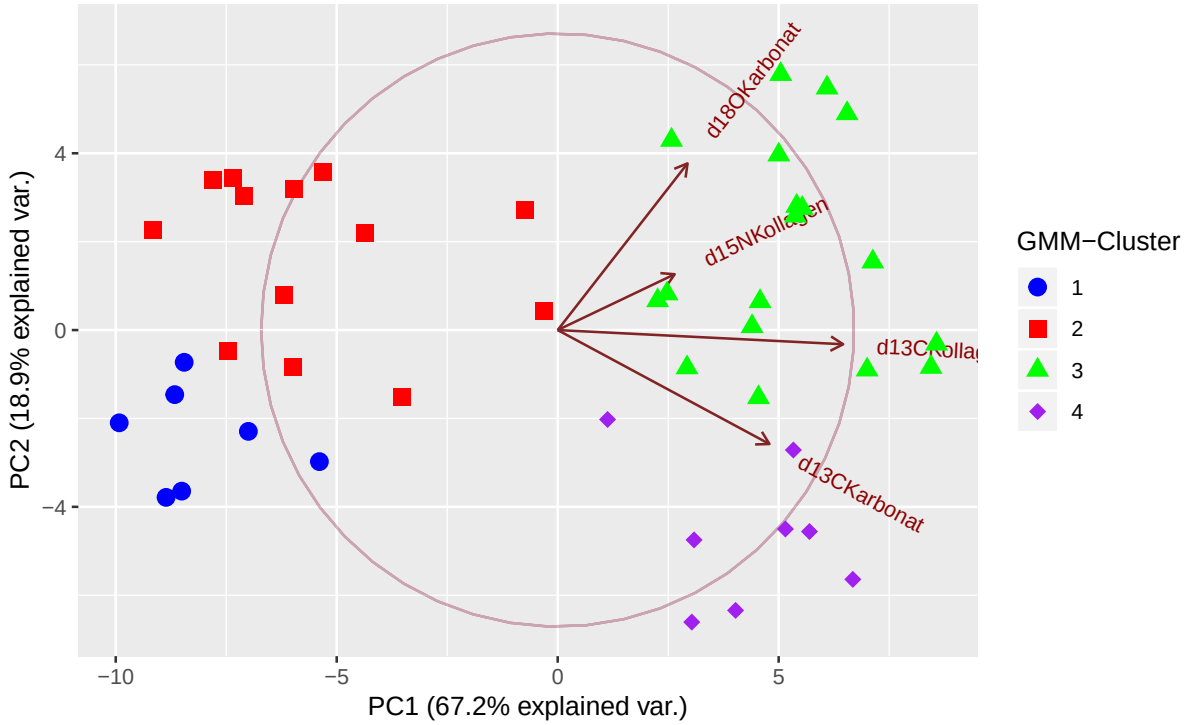
```
# Habitat
# pdf(width = 6, height = 6, "PCA_Fische_Habitat_PC1_PC2.pdf",
# encoding = "MacRoman")
ggbiplot(pca, choices = c(1, 2), obs.scale = 1, var.scale = 1,
         groups = Daten[, 3], ellipse = FALSE, circle = TRUE) +
  scale_color_manual(values = c("orange", "red", "purple", "blue"),
                     name = "Habitat",
                     breaks = c("Süßwasser, Brackwasser",
                                "Süßwasser, Brackwasser, Salzwasser",
                                "Brackwasser, Salzwasser",
                                "Salzwasser"),
                     labels = c("Süß-, Brackwasser",
                                "Süß-, Brack-, Salzwasser",
                                "Brack-, Salzwasser",
                                "Salzwasser"))+
  scale_shape_manual(values = c(17, 18, 19, 15),
                     name = "Habitat",
                     breaks = c("Süßwasser, Brackwasser",
                                "Süßwasser, Brackwasser, Salzwasser",
                                "Brackwasser, Salzwasser",
                                "Salzwasser"),
```

```
                    labels = c("Süß-, Brackwasser",
                               "Süß-, Brack-, Salzwasser",
                               "Brack-, Salzwasser",
                               "Salzwasser"))+
  geom_point(aes(colour = Daten[, 3], shape = Daten[, 3]),
                  size = 3)
```




```
# dev.off()
```

```
# Illustration der GMM-Cluster (sh. Kapitel 4.2)
# Umkodieren der GMM-Ergebnisse als Variable vom Typ "character"
GMM_optimal_EII_character <- as.character(Daten$GMM_optimal_EII)

# pdf(width=6,height=6,"PCA_Fische_GMM_PC1_PC2.pdf",encoding="MacRoman")
ggbiplot(pca, choices = c(1, 2), obs.scale = 1, var.scale = 1,
         groups = GMM_optimal_EII_character, ellipse = FALSE,
         circle = TRUE) +
  scale_color_manual(values = c("blue", "red", "green", "purple"),
                     name = "GMM-Cluster",
                     breaks = c("1", "2", "3", "4"),
                     labels = c("1", "2", "3", "4"))+
  scale_shape_manual(values = c(19, 15, 17, 18),
                     name = "GMM-Cluster",
                     breaks = c("1", "2", "3", "4"),
                     labels = c("1", "2", "3", "4"))+
  geom_point(aes(colour = GMM_optimal_EII_character,
                 shape = GMM_optimal_EII_character),
                 size = 3)
```


d18OKarbonat
d15NKollagen
d13CKollag
d13CKarbonat
PC2 (18.9% explained var.)
PC1 (67.2% explained var.)
4
0
−4
−10
−5
0
5
GMM−Cluster
1
2
3
4

```
# dev.off()
```

7 „Support Vector Machine“ (SVM)

Das R-Paket

- „kernlab“ (Version 0.9-27; Karatzoglou et al., 2004)

muss installiert sein.

```
# install.packages("kernlab")
library(kernlab)
```

1) Erzeugen des Trainings- und Testdatensatzes

```
set.seed(183)
Training_Individuen_2 <- sample(seq_len(nrow(Daten)),
                                size = 2/3*nrow(Daten))
Training_2 <- Daten[Training_Individuen_2, c(6:9, 3)]
Test_2 <- Daten[-Training_Individuen_2, c(6:9, 3)]
```

2) Erstellen der SVM

```
# Wahl des Kerns: vanilladot
# Trainingsdatensatz
vanilladot <- ksvm(Training_2[, 5]~.,data = Training_2[, 1:4],
                   type = "C-svc",kernel = "vanilladot",
                   C = 1, prob.model = TRUE)

#  Setting default kernel parameters

vanilladot
```

```
# Support Vector Machine object of class "ksvm"
#
# SV type: C-svc  (classification)
#  parameter : cost C = 1
#
# Linear (vanilla) kernel function.
#
# Number of Support Vectors : 23
#
# Objective Function Value : -5.291 -9.6064 -7.2209 -5.351 -3.4656 -7.5268
# Training error : 0.333333
# Probability model included.

# Klassifikation des Trainingsdatensatzes mit Hilfe der SVM
fitted(vanilladot)

#  [1] Süßwasser, Brackwasser  Brackwasser, Salzwasser
#  [3] Süßwasser, Brackwasser  Süßwasser, Brackwasser
#  [5] Brackwasser, Salzwasser Salzwasser
#  [7] Brackwasser, Salzwasser Süßwasser, Brackwasser
#  [9] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [11] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [13] Salzwasser              Süßwasser, Brackwasser
# [15] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [17] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [19] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [21] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [23] Süßwasser, Brackwasser  Brackwasser, Salzwasser
# [25] Süßwasser, Brackwasser  Brackwasser, Salzwasser
# [27] Süßwasser, Brackwasser  Süßwasser, Brackwasser
```

```
# [29] Salzwasser              Süßwasser, Brackwasser
# 4 Levels: Brackwasser, Salzwasser ... Süßwasser, Brackwasser, Salzwasser

# Testdatensatz
predict <- predict(vanilladot, Test_2[, 1:4], type="probabilities")
predict

#       Brackwasser, Salzwasser Salzwasser Süßwasser, Brackwasser
#  [1,]              0.09868128  0.1901700              0.5612434
#  [2,]              0.09981582  0.1459870              0.6484030
#  [3,]              0.21860881  0.1654738              0.4002956
#  [4,]              0.19923239  0.1660192              0.3840837
#  [5,]              0.24534142  0.1566435              0.3624619
#  [6,]              0.12066318  0.1395107              0.5993036
#  [7,]              0.16790477  0.1516777              0.4902001
#  [8,]              0.30419815  0.1402498              0.2706214
#  [9,]              0.27082432  0.1393173              0.3016128
# [10,]              0.16951672  0.1717726              0.3944774
# [11,]              0.30327544  0.1346298              0.2795759
# [12,]              0.07347992  0.1705856              0.6465542
# [13,]              0.33593179  0.1363229              0.2607378
# [14,]              0.11710100  0.1481745              0.5720113
# [15,]              0.14557898  0.1588959              0.5098707
# [16,]              0.12117258  0.1539392              0.5903868
#       Süßwasser, Brackwasser, Salzwasser
#  [1,]                          0.1499053
#  [2,]                          0.1057941
#  [3,]                          0.2156218
#  [4,]                          0.2506647
#  [5,]                          0.2355532
```

```
#  [6,]                          0.1405225
#  [7,]                          0.1902174
#  [8,]                          0.2849306
#  [9,]                          0.2882456
# [10,]                          0.2642333
# [11,]                          0.2825188
# [12,]                          0.1093802
# [13,]                          0.2670075
# [14,]                          0.1627132
# [15,]                          0.1856545
# [16,]                          0.1345014

# Wahl des Kerns: polydot (quadratisch)
# Trainingsdatensatz
polydot <- ksvm(Training_2[, 5]~., data = Training_2[, 1:4],
                    type="C-svc", kernel = "polydot"(degree = 2),
                    C = 1, prob.model = TRUE)
polydot

# Support Vector Machine object of class "ksvm"
#
# SV type: C-svc  (classification)
#  parameter : cost C = 1
#
# Polynomial kernel function.
#  Hyperparameters : degree =  2  scale =  1  offset =  1
#
# Number of Support Vectors : 20
#
# Objective Function Value : -1.9128 -1.1074 -1.95 -2.3777 -1.7616 -4.0998
```

```
# Training error : 0.033333
# Probability model included.

# Klassifikation des Trainingsdatensatzes mit Hilfe der SVM
fitted(polydot)

#  [1] Süßwasser, Brackwasser
#  [2] Brackwasser, Salzwasser
#  [3] Süßwasser, Brackwasser
#  [4] Brackwasser, Salzwasser
#  [5] Brackwasser, Salzwasser
#  [6] Brackwasser, Salzwasser
#  [7] Brackwasser, Salzwasser
#  [8] Salzwasser
#  [9] Süßwasser, Brackwasser
# [10] Süßwasser, Brackwasser
# [11] Süßwasser, Brackwasser
# [12] Süßwasser, Brackwasser, Salzwasser
# [13] Salzwasser
# [14] Brackwasser, Salzwasser
# [15] Süßwasser, Brackwasser
# [16] Süßwasser, Brackwasser
# [17] Süßwasser, Brackwasser
# [18] Süßwasser, Brackwasser
# [19] Süßwasser, Brackwasser
# [20] Salzwasser
# [21] Süßwasser, Brackwasser, Salzwasser
# [22] Süßwasser, Brackwasser
# [23] Süßwasser, Brackwasser
# [24] Brackwasser, Salzwasser
```

```
# [25] Süßwasser, Brackwasser
# [26] Süßwasser, Brackwasser
# [27] Süßwasser, Brackwasser, Salzwasser
# [28] Süßwasser, Brackwasser
# [29] Süßwasser, Brackwasser
# [30] Süßwasser, Brackwasser
# 4 Levels: Brackwasser, Salzwasser ... Süßwasser, Brackwasser, Salzwasser

# Testdatensatz
predict <- predict(polydot, Test_2[, 1:4], type = "probabilities")
predict

#       Brackwasser, Salzwasser Salzwasser Süßwasser, Brackwasser
#  [1,]              0.21444619 0.18178994              0.2889860
#  [2,]              0.13752244 0.16748265              0.3953212
#  [3,]              0.29367220 0.17964740              0.3015591
#  [4,]              0.23936941 0.17903599              0.3447806
#  [5,]              0.31139991 0.17937464              0.2867902
#  [6,]              0.03995794 0.12191429              0.6238380
#  [7,]              0.14108248 0.18659062              0.4206650
#  [8,]              0.45093710 0.15231282              0.1918770
#  [9,]              0.13683064 0.15136727              0.4806653
# [10,]              0.22645815 0.17925838              0.3202907
# [11,]              0.27295037 0.16040804              0.3137448
# [12,]              0.19567816 0.17396526              0.3045028
# [13,]              0.03487571 0.09129298              0.6957582
# [14,]              0.07384516 0.15947215              0.5318751
# [15,]              0.15082213 0.18844733              0.4064729
# [16,]              0.14703329 0.17859364              0.3890766
#       Süßwasser, Brackwasser, Salzwasser
```

```
#  [1,]                          0.3147779
#  [2,]                          0.2996737
#  [3,]                          0.2251213
#  [4,]                          0.2368140
#  [5,]                          0.2224352
#  [6,]                          0.2142897
#  [7,]                          0.2516619
#  [8,]                          0.2048730
#  [9,]                          0.2311368
# [10,]                          0.2739927
# [11,]                          0.2528968
# [12,]                          0.3258538
# [13,]                          0.1780731
# [14,]                          0.2348076
# [15,]                          0.2542576
# [16,]                          0.2852964

# Wahl des Kerns: rbfdot
# Trainingsdatensatz
rbfdot<-ksvm(Training_2[, 5]~., data = Training_2[, 1:4],
                 type = "C-svc", kernel = "rbfdot",
                 C = 1, prob.model = TRUE)
rbfdot

# Support Vector Machine object of class "ksvm"
#
# SV type: C-svc  (classification)
#  parameter : cost C = 1
#
# Gaussian Radial Basis kernel function.
```

```
#  Hyperparameter : sigma =  0.267008952646232
#
# Number of Support Vectors : 24
#
# Objective Function Value : -5.2904 -8.39 -6.4943 -5.5285 -4.7595 -7.3926
# Training error : 0.3
# Probability model included.

# Klassifikation des Trainingsdatensatzes mit Hilfe der SVM
fitted(rbfdot)

#  [1] Süßwasser, Brackwasser  Brackwasser, Salzwasser
#  [3] Süßwasser, Brackwasser  Brackwasser, Salzwasser
#  [5] Süßwasser, Brackwasser  Brackwasser, Salzwasser
#  [7] Brackwasser, Salzwasser Süßwasser, Brackwasser
#  [9] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [11] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [13] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [15] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [17] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [19] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [21] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [23] Süßwasser, Brackwasser  Brackwasser, Salzwasser
# [25] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [27] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [29] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# 4 Levels: Brackwasser, Salzwasser ... Süßwasser, Brackwasser, Salzwasser

# Testdatensatz
predict <- predict(rbfdot, Test_2[, 1:4], type="probabilities")
predict
```

```
#       Brackwasser, Salzwasser Salzwasser Süßwasser, Brackwasser
#  [1,]              0.1942135  0.1839954             0.3475950
#  [2,]              0.1978125  0.1833141             0.3435042
#  [3,]              0.2638478  0.1669982             0.3205859
#  [4,]              0.1603589  0.1761390             0.4182485
#  [5,]              0.3161627  0.1570909             0.2876131
#  [6,]              0.1871056  0.1855229             0.3499109
#  [7,]              0.2040063  0.1811237             0.3433546
#  [8,]              0.4668980  0.1490813             0.1332477
#  [9,]              0.2148486  0.1566733             0.3962541
# [10,]              0.1837123  0.1794415             0.3823933
# [11,]              0.3842708  0.1572714             0.1950653
# [12,]              0.1844050  0.1862743             0.3500954
# [13,]              0.1609875  0.1590457             0.4526871
# [14,]              0.1839828  0.1860447             0.3520514
# [15,]              0.1866558  0.1854690             0.3504011
# [16,]              0.1844033  0.1874096             0.3440817
#       Süßwasser, Brackwasser, Salzwasser
#  [1,]                          0.2741960
#  [2,]                          0.2753692
#  [3,]                          0.2485681
#  [4,]                          0.2452535
#  [5,]                          0.2391333
#  [6,]                          0.2774607
#  [7,]                          0.2715154
#  [8,]                          0.2507730
#  [9,]                          0.2322240
# [10,]                          0.2544529
```

```
# [11,]                          0.2633925
# [12,]                          0.2792253
# [13,]                          0.2272797
# [14,]                          0.2779210
# [15,]                          0.2774740
# [16,]                          0.2841054
```

3) Graphische Darstellung

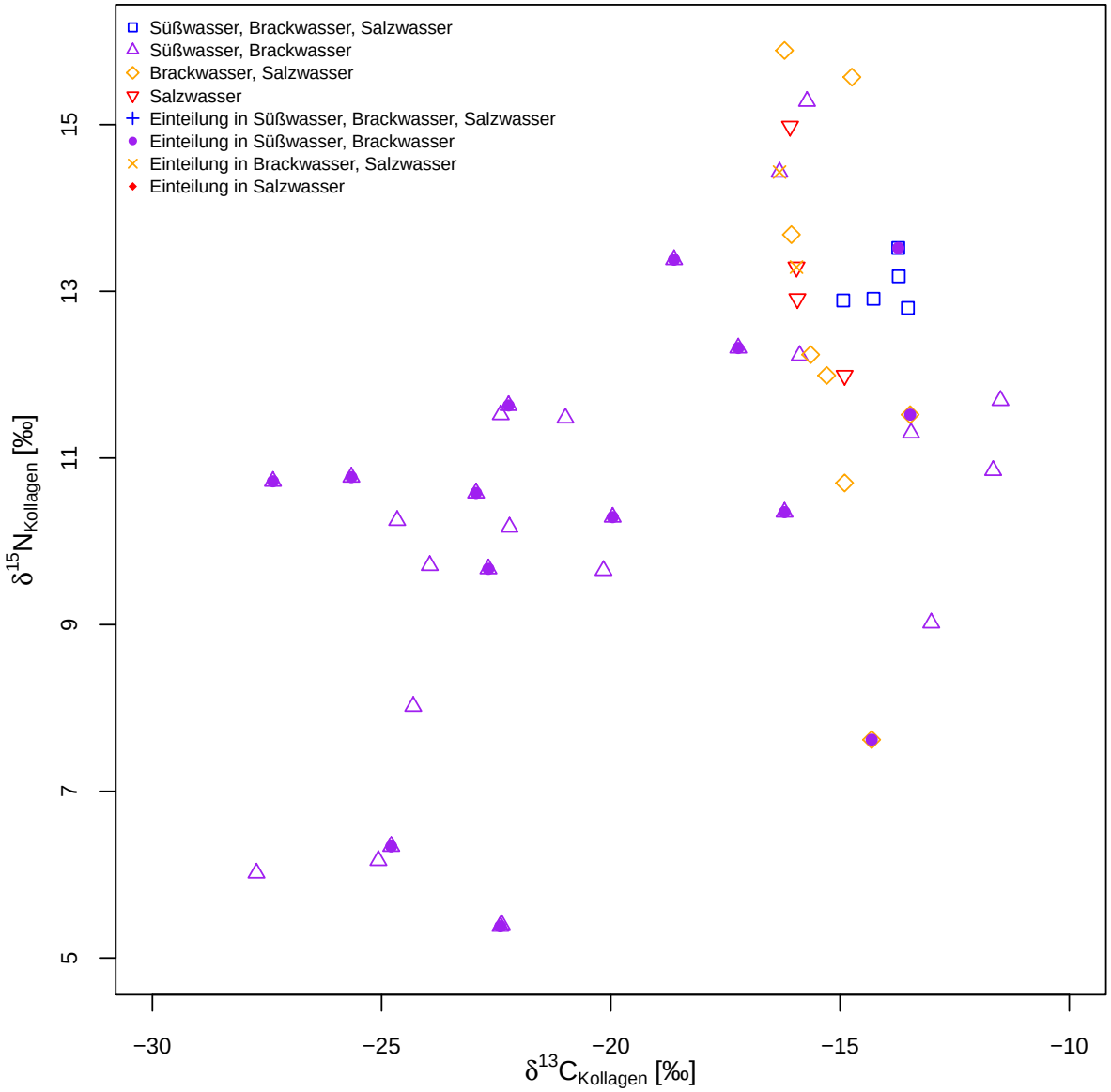
```
# Graphische Darstellung am Beispiel "rbfdot"
rbfdot<-ksvm(Training_2[, 5]~., data = Training_2[, 1:4],
                 type = "C-svc", kernel = "rbfdot",
                 C = 1, prob.model = TRUE)
predict <- predict(rbfdot, Test_2[, 1:4])
predict

#  [1] Süßwasser, Brackwasser  Süßwasser, Brackwasser
#  [3] Süßwasser, Brackwasser  Süßwasser, Brackwasser
#  [5] Süßwasser, Brackwasser  Süßwasser, Brackwasser
#  [7] Süßwasser, Brackwasser  Brackwasser, Salzwasser
#  [9] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [11] Brackwasser, Salzwasser Süßwasser, Brackwasser
# [13] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# [15] Süßwasser, Brackwasser  Süßwasser, Brackwasser
# 4 Levels: Brackwasser, Salzwasser ... Süßwasser, Brackwasser, Salzwasser

# tatsächliche Klassenzuordnung des Trainingsdatensatzes
train_SuessBrackSalz <- subset(Training_2, subset = Training_2$Habitat==
                               "Süßwasser, Brackwasser, Salzwasser")
train_SuessBrack <- subset(Training_2, subset = Training_2$Habitat==
```

```
                              "Süßwasser, Brackwasser")
train_BrackSalz <- subset(Training_2, subset = Training_2$Habitat==
                              "Brackwasser, Salzwasser")
train_Salz <- subset(Training_2, subset = Training_2$Habitat==
                         "Salzwasser")
# tatsächliche Klassenzuordnung des Testdatensatzes
test_SuessBrackSalz <- subset(Test_2, subset = Test_2$Habitat==
                                  "Süßwasser, Brackwasser, Salzwasser")
test_SuessBrack <- subset(Test_2, subset = Test_2$Habitat==
                              "Süßwasser, Brackwasser")
test_BrackSalz <- subset(Test_2, subset = Test_2$Habitat==
                            "Brackwasser, Salzwasser")
test_Salz <- subset(Test_2, subset = Test_2$Habitat==
                        "Salzwasser")
```

```
# Gemäß SVM vorhergesagte Klassenzuordnung des Trainingsdatensatzes
predict_SuessBrackSalz <- subset(Test_2,
                                 subset = predict=="Süßwasser,
                                  Brackwasser, Salzwasser")
predict_SuessBrack <- subset(Test_2,
                             subset = predict=="Süßwasser, Brackwasser")
predict_BrackSalz <- subset(Test_2,
                            subset = predict=="Brackwasser, Salzwasser")
predict_Salz <- subset(Test_2,
                       subset = predict=="Salzwasser")

# pdf(width = 6, height = 6, "SVM_rbfdot_d13CKolld15NKoll.pdf",
# encoding="MacRoman")
par(mar = c(3.5, 3.5, 1, 1))
plot(Test_2[, 1], Test_2[, 2], type = "n",
     xlim = c(-30,-10), ylim = c(5, 16),
     ann = FALSE, axes = FALSE)
points(test_SuessBrackSalz[, 1], test_SuessBrackSalz[, 2],
       pch = 0, col = "blue")
points(test_SuessBrack[, 1], test_SuessBrack[, 2],
       pch = 2, col = "purple")
points(test_BrackSalz[, 1], test_BrackSalz[, 2],
       pch = 5, col = "orange")
points(test_Salz[, 1], test_Salz[, 2],
       pch = 6, col = "red")
points(train_SuessBrackSalz[, 1], train_SuessBrackSalz[, 2],
       pch = 0, col = "blue")
points(train_SuessBrack[, 1], train_SuessBrack[, 2],
```

```
       pch = 2, col = "purple")
points(train_BrackSalz[, 1], train_BrackSalz[, 2],
       pch = 5, col = "orange")
points(train_Salz[, 1], train_Salz[, 2],
       pch = 6, col = "red")
points(predict_SuessBrackSalz[, 1], predict_SuessBrackSalz[, 2],
       pch = 3, col = "blue")
points(predict_SuessBrack[, 1], predict_SuessBrack[, 2],
       pch = 16, col = "purple")
points(predict_BrackSalz[, 1], predict_BrackSalz[, 2],
       pch = 4, col = "orange")
points(predict_Salz[, 1], predict_Salz[, 2],
       pch = 18, col = "red")
box()
axis(1, at = seq(-30, -10, 5), cex.axis = 0.8)
axis(2, at = seq(5, 15, 2), cex.axis = 0.8)
mtext(side = 1, line = 2.1,
      expression(paste(delta^13*"C"[Kollagen]~"[\211]")))
mtext(side = 2, line = 2.1,
      expression(paste(delta^15*"N"[Kollagen]~"[\211]")))
legend("topleft",
       legend = c("Süßwasser, Brackwasser, Salzwasser",
                  "Süßwasser, Brackwasser",
                  "Brackwasser, Salzwasser",
                  "Salzwasser",
                  "Einteilung in Süßwasser, Brackwasser, Salzwasser",
                  "Einteilung in Süßwasser, Brackwasser",
                  "Einteilung in Brackwasser, Salzwasser",
```

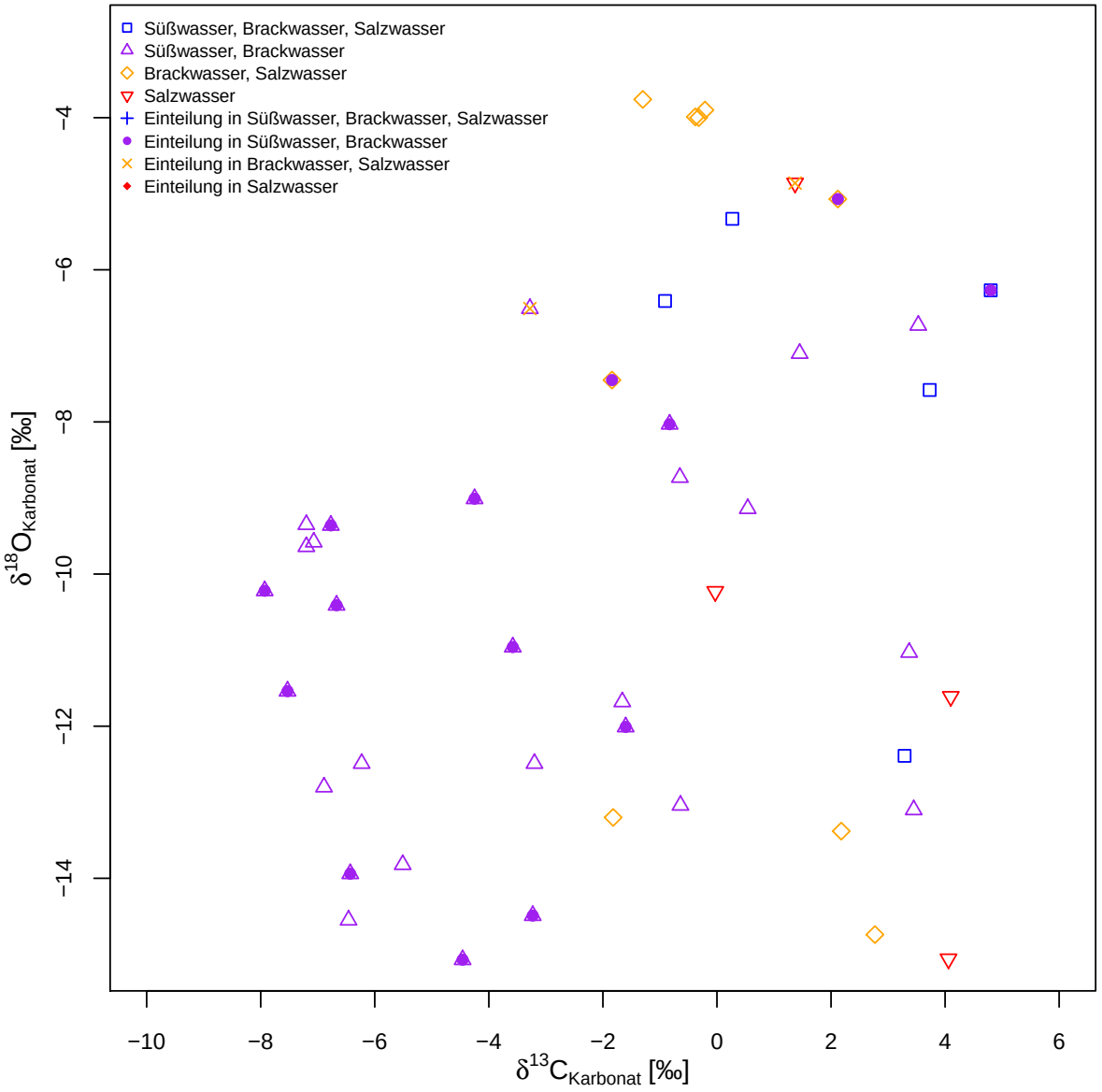
```
                "Einteilung in Salzwasser"),
       col = c("blue", "purple", "orange", "red",
               "blue", "purple", "orange", "red"),
       pch = c(0, 2, 5, 6, 3, 16, 4, 18), bty = "n", cex = 0.7)
```




```
# dev.off()


# pdf(width = 6, height = 6, "SVM_rbfdot_d13CKarbd18OKarb.pdf",
# encoding = "MacRoman")
```

```
par(mar = c(3.5, 3.5, 1, 1))
plot(Test_2[, 3], Test_2[, 4], type = "n",
     xlim = c(-10, 6), ylim = c(-15,-3),
     ann = FALSE, axes = FALSE)
points(test_SuessBrackSalz[, 3], test_SuessBrackSalz[, 4],
       pch = 0, col = "blue")
points(test_SuessBrack[, 3], test_SuessBrack[, 4],
       pch = 2, col = "purple")
points(test_BrackSalz[, 3], test_BrackSalz[, 4],
       pch = 5, col = "orange")
points(test_Salz[, 3], test_Salz[, 4],
       pch = 6, col = "red")
points(train_SuessBrackSalz[, 3], train_SuessBrackSalz[, 4],
       pch = 0, col = "blue")
points(train_SuessBrack[, 3], train_SuessBrack[, 4],
       pch = 2, col = "purple")
points(train_BrackSalz[, 3], train_BrackSalz[, 4],
       pch = 5, col = "orange")
points(train_Salz[, 3], train_Salz[, 4],
       pch = 6, col = "red")
points(predict_SuessBrackSalz[, 3], predict_SuessBrackSalz[, 4],
       pch = 3, col = "blue")
points(predict_SuessBrack[, 3], predict_SuessBrack[, 4],
       pch = 16, col = "purple")
points(predict_BrackSalz[, 3], predict_BrackSalz[, 4],
       pch = 4, col = "orange")
points(predict_Salz[, 3], predict_Salz[, 4],
       pch = 18, col = "red")
```

```
box()
axis(1, at = seq(-10, 6, 2), cex.axis = 0.8)
axis(2, at = seq(-14, -4, 2), cex.axis = 0.8)
mtext(side = 1, line = 2.1,
      expression(paste(delta^13*"C"[Karbonat]~"[\211]")))
mtext(side = 2, line = 2.1,
      expression(paste(delta^18*"O"[Karbonat]~"[\211]")))
legend("topleft",
       legend = c("Süßwasser, Brackwasser, Salzwasser",
                  "Süßwasser, Brackwasser",
                  "Brackwasser, Salzwasser",
                  "Salzwasser",
                  "Einteilung in Süßwasser, Brackwasser, Salzwasser",
                  "Einteilung in Süßwasser, Brackwasser",
                  "Einteilung in Brackwasser, Salzwasser",
                  "Einteilung in Salzwasser"),
       col = c("blue", "purple", "orange", "red",
             "blue", "purple", "orange", "red"),
       pch = c(0, 2, 5, 6, 3, 16, 4, 18), bty = "n", cex = 0.7)
```


Süßwasser, Brackwasser, Salzwasser
Süßwasser, Brackwasser
Brackwasser, Salzwasser
Salzwasser
Einteilung in Süßwasser, Brackwasser, Salzwasser
Einteilung in Süßwasser, Brackwasser
Einteilung in Brackwasser, Salzwasser
Einteilung in Salzwasser
δ18OKarbonat [‰]
δ13CKarbonat [‰]
−4
−6
−8
−10
−12
−14
−10
−8
−6
−4
−2
0
2
4
6

```
# dev.off()
```

8 „Feature Ranking“

8.1 „Adjusted Rand Index“ (ARI)

Das R-Paket

- „mclust“ (Version 5.4.1; Scrucca et al., 2016)

muss installiert sein (vgl. Kapitel 4.2).

```
# install.packages("mclust")
library(mclust)
mclust.options(hcUse = "VARS")
```

1) Detektion und Entfernen multivariater Ausreißer

vgl. Kapitel 3

2) Normalisieren der Daten

```
norm <- function(x) {
  (x - min(x, na.rm = TRUE))/(max(x, na.rm = TRUE) -
                                min(x, na.rm = TRUE))
  }
Daten_norm <- as.data.frame(lapply(Daten[, 6:9], norm))
```

3) GMM-Clusteranalyse der einzelnen Merkmalskombinationen

vgl. hierzu auch Kapitel 4.2

```
# alle Dimensionen
alle <- Mclust(Daten_norm[, c(1, 2, 3, 4)])
summary(alle)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust EVE (ellipsoidal, equal volume and orientation) model with 2
# components:
#
#  log.likelihood  n df      BIC      ICL
#       48.92404 46 22 13.61797 13.60538
#
# Clustering table:
#  1  2
# 18 28

# ohne d13CKollagen
ohned13CKoll <- Mclust(Daten_norm[, c(2, 3, 4)])
summary(ohned13CKoll)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust VII (spherical, varying volume) model with 4 components:
#
#  log.likelihood  n df       BIC       ICL
#       22.73675 46 19 -27.27068 -32.23148
```

```
#
# Clustering table:
#  1  2  3  4
# 18 16  7  5

# ohne d15NKollagen
ohned15NKoll <- Mclust(Daten_norm[, c(1, 3, 4)])
summary(ohned15NKoll)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust EVI (diagonal, equal volume, varying shape) model with 2
# components:
#
#  log.likelihood  n df       BIC       ICL
#        20.88173 46 12 -4.180243 -5.060804
#
# Clustering table:
#  1  2
# 20 26

# ohne d13CKarbonat
ohned13CKarb <- Mclust(Daten_norm[, c(1, 2, 4)])
summary(ohned13CKarb)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
```

```
# Mclust EVI (diagonal, equal volume, varying shape) model with 2
# components:
#
#  log.likelihood  n df       BIC       ICL
#       21.66144 46 12 -2.620824 -2.846349
#
# Clustering table:
#  1  2
# 18 28

# ohne d18OKarbonat
ohned18OKarb <- Mclust(Daten_norm[, c(1, 2, 3)])
summary(ohned18OKarb)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust EVE (ellipsoidal, equal volume and orientation) model with 2
# components:
#
#  log.likelihood  n df      BIC      ICL
#       43.64922 46 15 29.86881 29.84276
#
# Clustering table:
#  1  2
# 18 28

# nur d13CKollagen
nurd13CKoll <- Mclust(Daten_norm[, 1])
summary(nurd13CKoll)
```

```
# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust E (univariate, equal variance) model with 2 components:
#
#  log.likelihood  n df       BIC      ICL
#        5.490379 46  4 -4.333808 -5.44327
#
# Clustering table:
#  1  2
# 18 28

# nur d15NKollagen
nurd15NKoll <- Mclust(Daten_norm[, 2])
summary(nurd15NKoll)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust X (univariate normal) model with 1 component:
#
#  log.likelihood  n df       BIC       ICL
#        2.727351 46  2 -2.202581 -2.202581
#
# Clustering table:
#  1
# 46
```

```
# nur d13CKarbonat
nurd13CKarb <- Mclust(Daten_norm[, 3])
summary(nurd13CKarb)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust V (univariate, unequal variance) model with 2 components:
#
#  log.likelihood  n df       BIC       ICL
#      -0.4132348 46  5 -19.96968 -21.60182
#
# Clustering table:
#  1  2
# 11 35

# nur d18OKarbonat
nurd18OKarb <- Mclust(Daten_norm[, 4])
summary(nurd18OKarb)

# ----------------------------------------------------
# Gaussian finite mixture model fitted by EM algorithm
# ----------------------------------------------------
#
# Mclust X (univariate normal) model with 1 component:
#
#  log.likelihood  n df       BIC       ICL
#       -9.669572 46  2 -26.99643 -26.99643
#
```

```
# Clustering table:
#  1
# 46
```

3) Berechnen des „Adjusted Rand Index“ (ARI)

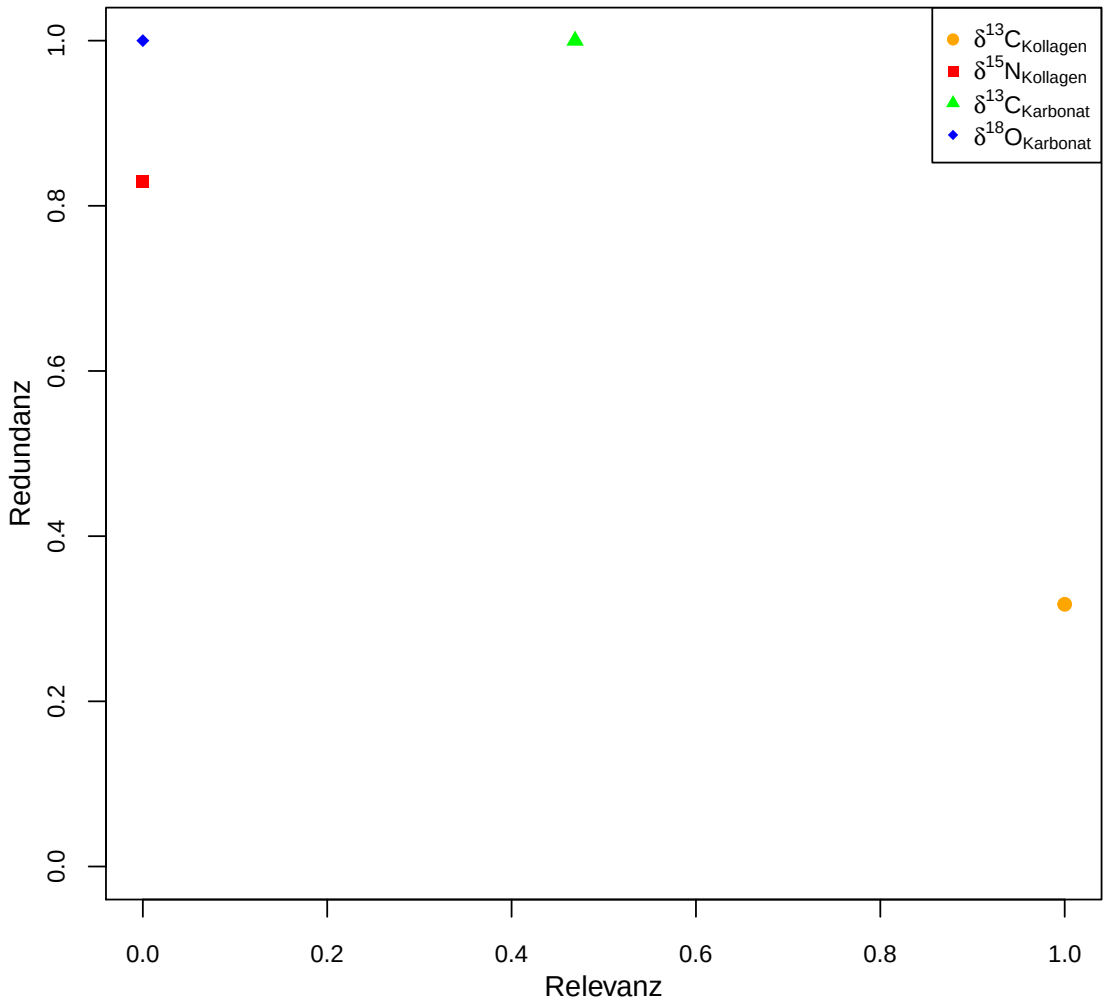
```
# d13CKollagen
# Relevanz
d13CKoll_Rel <- adjustedRandIndex(nurd13CKoll$classification,
                                  alle$classification)
# Redundanz
d13CKoll_Red <- adjustedRandIndex(ohned13CKoll$classification,
                                  alle$classification)
# d15NKollagen
# Relevanz
d15NKoll_Rel <- adjustedRandIndex(nurd15NKoll$classification,
                                  alle$classification)
# Redundanz
d15NKoll_Red <- adjustedRandIndex(ohned15NKoll$classification,
                                  alle$classification)
# d13CKarbonat
# Relevanz
d13CKarb_Rel <- adjustedRandIndex(nurd13CKarb$classification,
                                  alle$classification)
# Redundanz
d13CKarb_Red <- adjustedRandIndex(ohned13CKarb$classification,
                                  alle$classification)
```

```
# d18OKarbonat
# Relevanz
d18OKarb_Rel <- adjustedRandIndex(nurd18OKarb$classification,
                                  alle$classification)
# Redundanz
d18OKarb_Red <- adjustedRandIndex(ohned18OKarb$classification,
                                  alle$classification)

# Relevanz und Redundanz
Relevanz <- c(d13CKoll_Rel, d15NKoll_Rel, d13CKarb_Rel, d18OKarb_Rel)
Redundanz <- c(d13CKoll_Red, d15NKoll_Red, d13CKarb_Red, d18OKarb_Red)
ARI <- cbind(Relevanz, Redundanz)
rownames(ARI) <- c("d13CKollagen", "d15NKollagen", "d13CKarbonat",
                   "d18OKarbonat")
ARI

#               Relevanz Redundanz
# d13CKollagen 1.0000000 0.3174904
# d15NKollagen 0.0000000 0.8299731
# d13CKarbonat 0.4689165 1.0000000
# d18OKarbonat 0.0000000 1.0000000
```

3) Graphische Darstellung der Relevanz und Redundanz

```
# pdf(width = 6, height = 6, "ARI_Plot.pdf",
# encoding =" MacRoman")
plot(c(0, 1), c(0, 1), type = "n", ann = FALSE, axes = FALSE)
box()
axis(1, seq(0, 1, 0.2), cex.axis = 0.8)
```

```
axis(2, seq(0, 1, 0.2), cex.axis = 0.8)
mtext(side = 1, line = 2, "Relevanz")
mtext(side = 2, line = 2, "Redundanz")
points(ARI[, 1], ARI[, 2],
       col = c("orange", "red", "green", "blue"),
       pch = c(19, 15, 17, 18))
legend("topright", cex = 0.8,
       legend = c(expression(paste(delta^13*"C"[Kollagen])),
                  expression(paste(delta^15*"N"[Kollagen])),
                  expression(paste(delta^13*"C"[Karbonat])),
                  expression(paste(delta^18*"O"[Karbonat]))),
       col = c("orange", "red", "green", "blue"),
       pch = c(19, 15, 17, 18))
```


Redundanz
Relevanz
0.0
0.2
0.4
0.6
0.8
1.0
δ13CKollagen
δ15NKollagen
δ13CKarbonat
δ18OKarbonat

```
# dev.off()
```

8.2 Entropie-basiertes „Feature Ranking“

Das R-Paket

- „clusterSim“ (Version 0.47-1; Walesiak & Dudek, 2017)

muss installiert sein.

```
# install.packages("clusterSim")
library(clusterSim)
```

1) Detektion und Entfernen multivariater Ausreißer

vgl. Kapitel 3

2) Normalisieren der Daten

```
norm <- function(x) {
  (x - min(x, na.rm = TRUE))/(max(x, na.rm = TRUE) -
                                min(x, na.rm = TRUE))
  }
Daten_norm <- as.data.frame(lapply(Daten[, 6:9], norm))
```

3) Bilden von Distanzmatrizen D_{ij}

```
Dij1 <- as.matrix(dist(Daten_norm[, 1],
                       method = "euclidean",
                       diag = FALSE, upper = TRUE))
Dij2 <- as.matrix(dist(Daten_norm[, 2],
                       method = "euclidean",
                       diag = FALSE, upper = TRUE))
```

```
Dij3 <- as.matrix(dist(Daten_norm[, 3],
                       method = "euclidean",
                       diag = FALSE,upper = TRUE))
Dij4 <- as.matrix(dist(Daten_norm[, 4],
                       method = "euclidean",
                       diag = FALSE,upper = TRUE))
Dij12 <- as.matrix(dist(Daten_norm[, c(1, 2)],
                        method = "euclidean",
                        diag = FALSE, upper = TRUE))
Dij13 <- as.matrix(dist(Daten_norm[, c(1, 3)],
                        method = "euclidean",
                        diag = FALSE, upper = TRUE))
Dij14 <- as.matrix(dist(Daten_norm[, c(1, 4)],
                        method = "euclidean",
                        diag = FALSE, upper = TRUE))
Dij23 <- as.matrix(dist(Daten_norm[, c(2, 3)],
                        method = "euclidean",
                        diag = FALSE, upper = TRUE))
Dij24 <- as.matrix(dist(Daten_norm[, c(2, 4)],
                        method = "euclidean",
                        diag = FALSE, upper = TRUE))
Dij34 <- as.matrix(dist(Daten_norm[, c(3, 4)],
                        method = "euclidean",
                        diag = FALSE, upper = TRUE))
Dij123 <- as.matrix(dist(Daten_norm[, c(1, 2, 3)],
                         method = "euclidean",
                         diag = FALSE, upper = TRUE))
Dij124 <- as.matrix(dist(Daten_norm[, c(1, 2, 4)],
```

```
                          method = "euclidean",
                          diag = FALSE, upper = TRUE))
Dij134 <- as.matrix(dist(Daten_norm[, c(1, 3, 4)],
                          method = "euclidean",
                          diag = FALSE, upper = TRUE))
Dij234 <- as.matrix(dist(Daten_norm[, c(2, 3, 4)],
                          method = "euclidean",
                          diag = FALSE, upper = TRUE))
Dij1234 <- as.matrix(dist(Daten_norm[, c(1, 2, 3, 4)],
                           method = "euclidean",
                           diag = FALSE, upper = TRUE))
```

4) Normalisieren der Distanzmatrizen

```
Dij1norm <- data.Normalization(Dij1, type = "n4",
                               normalization = "column")
Dij2norm <- data.Normalization(Dij2, type = "n4",
                               normalization = "column")
Dij3norm <- data.Normalization(Dij3, type = "n4",
                               normalization = "column")
Dij4norm <- data.Normalization(Dij4, type = "n4",
                               normalization = "column")
Dij12norm <- data.Normalization(Dij12, type = "n4",
                                normalization = "column")
Dij13norm <- data.Normalization(Dij13, type = "n4",
                                normalization = "column")
Dij14norm <- data.Normalization(Dij14, type = "n4",
                                normalization = "column")
```

```
Dij23norm <- data.Normalization(Dij23, type = "n4",
                                normalization = "column")
Dij24norm <- data.Normalization(Dij24, type = "n4",
                                normalization = "column")
Dij34norm <- data.Normalization(Dij34, type = "n4",
                                normalization = "column")
Dij123norm <- data.Normalization(Dij123, type = "n4",
                                 normalization = "column")
Dij124norm <- data.Normalization(Dij124, type = "n4",
                                 normalization = "column")
Dij134norm <- data.Normalization(Dij134, type = "n4",
                                 normalization = "column")
Dij234norm <- data.Normalization(Dij234, type = "n4",
                                 normalization = "column")
Dij1234norm <- data.Normalization(Dij1234, type = "n4",
                                  normalization = "column")
```

5) Berechnen der Entropiewerte

```
Entropieformel <- function(x = 0){
  ifelse(x > 0 & x < 1, -x * log2(x), 0)
  }
Entropie<-function(x){
  sum(Entropieformel(x))
  }


E1 <- Entropie(Dij1norm)
E2 <- Entropie(Dij2norm)
E3 <- Entropie(Dij3norm)
```

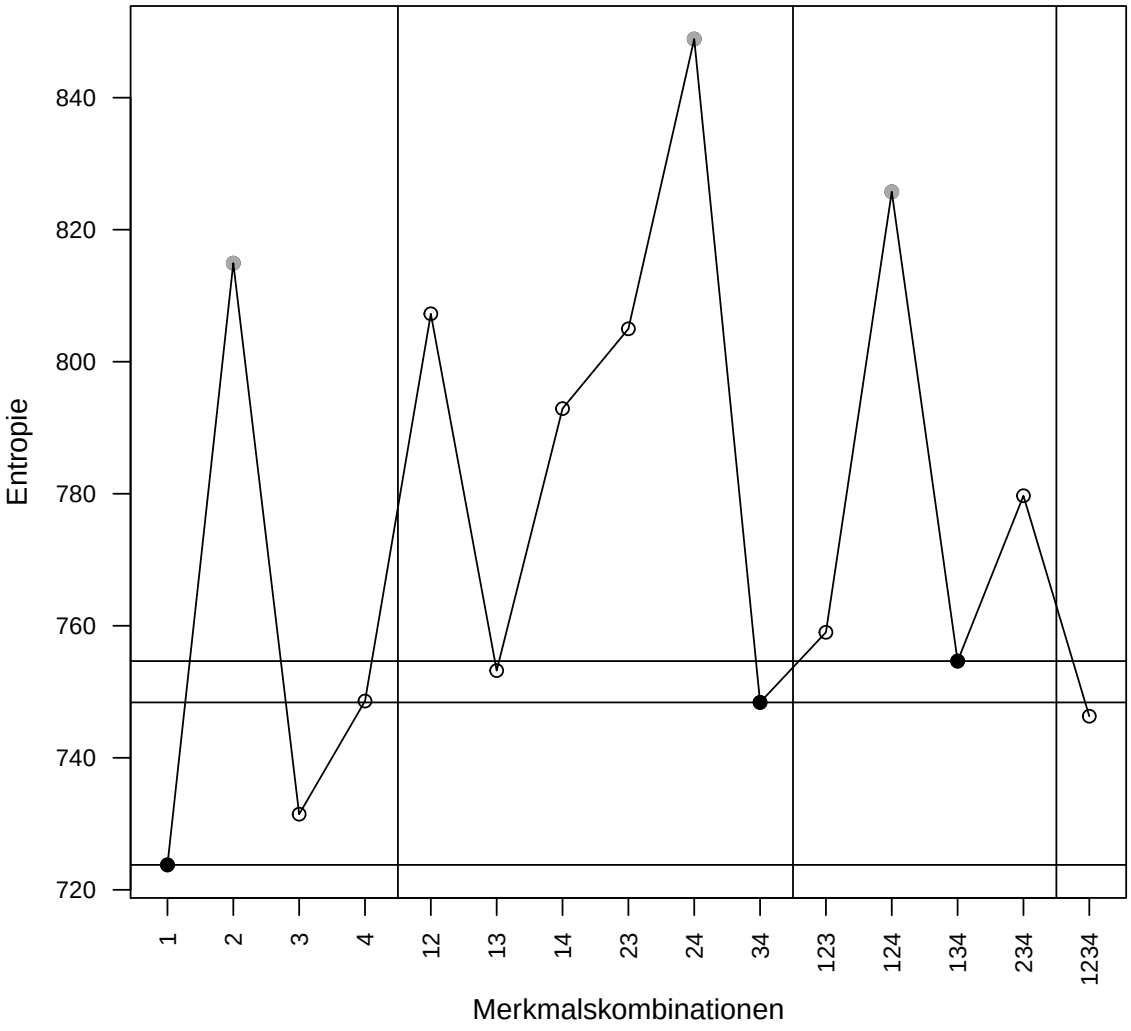
```
E4 <- Entropie(Dij4norm)

E12 <- Entropie(Dij12norm)

E13 <- Entropie(Dij13norm)

E14 <- Entropie(Dij14norm)

E23 <- Entropie(Dij23norm)

E24 <- Entropie(Dij24norm)

E34 <- Entropie(Dij34norm)

E123 <- Entropie(Dij123norm)

E124 <- Entropie(Dij124norm)

E134 <- Entropie(Dij134norm)

E234 <- Entropie(Dij234norm)

E1234 <- Entropie(Dij1234norm)
```

6) Graphische Darstellung der Entropiewerte

```
Entropie <- c(E1, E2, E3, E4, E12, E13, E14, E23, E24,
              E34, E123, E124, E134, E234, E1234)


min_1 <- min(E1, E2, E3, E4)
min_1

# [1] 723.7762

min_2 <- min(E12, E13, E14, E23, E24, E34)
min_2

# [1] 748.3893

min_3 <- min(E123, E124, E134, E234)
min_3
```

```
# [1] 754.6528

max_1 <- max(E1, E2, E3, E4)
max_1

# [1] 814.9278

max_2 <- max(E12, E13, E14, E23, E24, E34)
max_2

# [1] 848.8793

max_3 <- max(E123, E124, E134, E234)
max_3

# [1] 825.7409

# pdf(width = 6, height = 6, "Entropie_Plot.pdf",
# encoding = "MacRoman")
plot(Entropie, ann = FALSE, axes = FALSE)
box()
axis(1, at = seq(1:15), labels = c("1", "2", "3", "4", "12", "13",
                                   "14", "23", "24", "34", "123", "124",
                                   "134", "234", "1234"),
     cex.axis = 0.8, las = 2)
axis(2, at = seq(720, 840, 20), cex.axis = 0.8, las = 2)
mtext(side = 1, line=2.7, "Merkmalskombinationen")
mtext(side = 2, line=2.8, "Entropie")
abline(v = 4.5)
abline(v = 10.5)
abline(v = 14.5)
points(match(min_1, Entropie), min_1, pch = 19, col = "black")
```

```
points(match(min_2, Entropie), min_2, pch = 19, col = "black")
points(match(min_3, Entropie), min_3, pch = 19, col = "black")
abline(h = min_1, col = "black", lwd = 1)
abline(h = min_2, col = "black", lwd = 1)
abline(h = min_3, col = "black", lwd = 1)
points(match(max_1, Entropie), max_1, pch = 19, col = "darkgrey")
points(match(max_2, Entropie), max_2, pch = 19, col = "darkgrey")
points(match(max_3, Entropie), max_3, pch = 19, col = "darkgrey")
lines(Entropie)
```



```
# dev.off()
```

7) Trace-Index

Die R-Pakete

- „clusterCrit“ (Version 1.2.8; Desgraupes, 2018)
- „mclust“ (Version 5.4.1; Scrucca et al., 2016)

müssen installiert sein.

```
# install.packages(c("clusterCrit", "mclust"))
library(clusterCrit)
library(mclust)
mclust.options(hcUse="VARS")
```

1) Berechnen des Trace-Index

```
GMM1 <- Mclust(Daten_norm[,1])
matrix <- data.matrix(Daten_norm[,1], rownames.force = NA)
crit1 <- intCriteria(matrix, as.integer(GMM1$classification), "all")

GMM2 <- Mclust(Daten_norm[,2])
matrix <- data.matrix(Daten_norm[,2], rownames.force = NA)
crit2 <- intCriteria(matrix, as.integer(GMM2$classification), "all")

GMM3 <- Mclust(Daten_norm[,3])
matrix <- data.matrix(Daten_norm[,3], rownames.force = NA)
crit3 <- intCriteria(matrix, as.integer(GMM3$classification), "all")
```

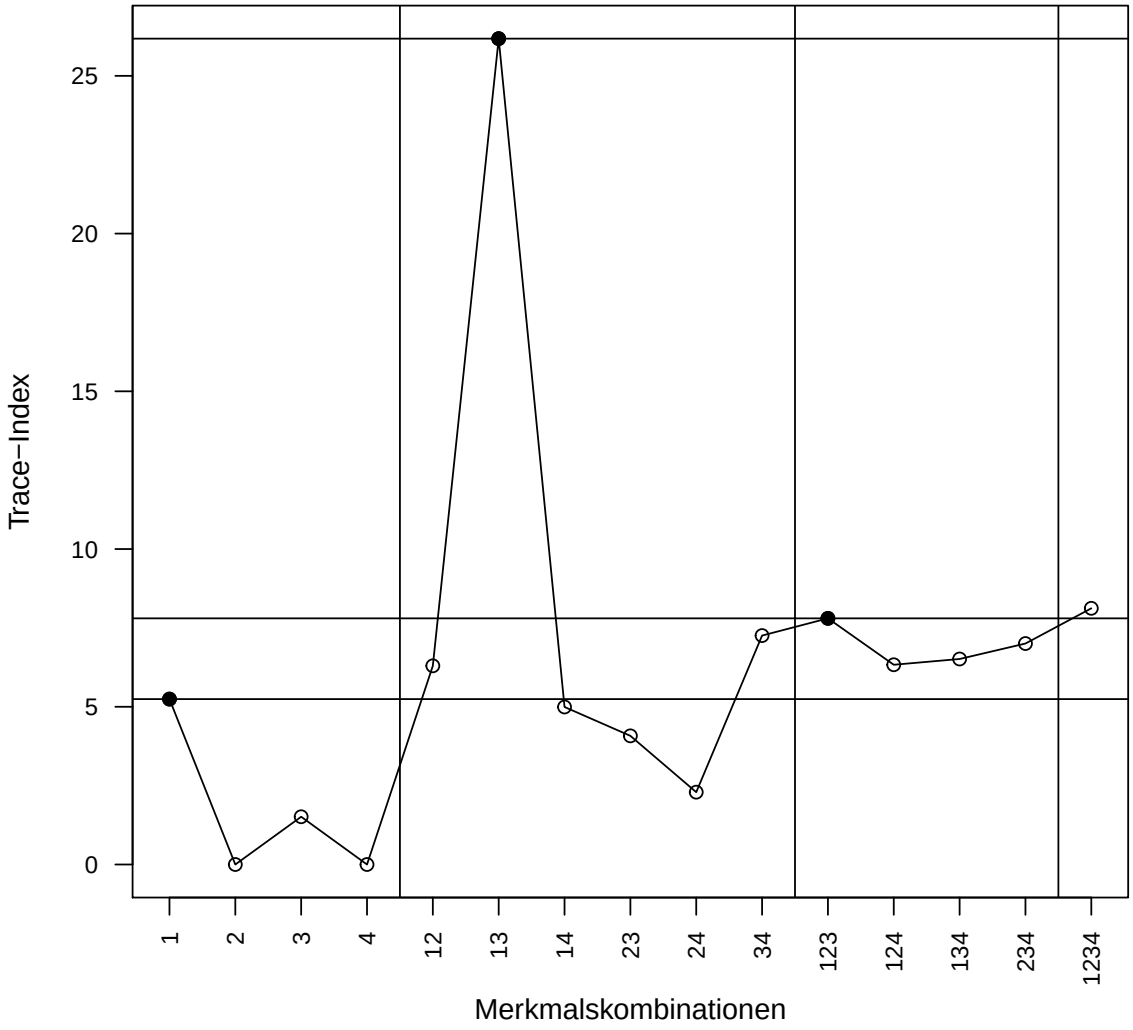
```
GMM4 <- Mclust(Daten_norm[,4])
matrix <- data.matrix(Daten_norm[,4], rownames.force = NA)
crit4 <- intCriteria(matrix, as.integer(GMM4$classification), "all")

GMM12 <- Mclust(Daten_norm[, c(1, 2)])
matrix <- data.matrix(Daten_norm[, c(1, 2)], rownames.force = NA)
crit12 <- intCriteria(matrix, as.integer(GMM12$classification), "all")

GMM13 <- Mclust(Daten_norm[, c(1, 3)])
matrix <- data.matrix(Daten_norm[, c(1, 3)], rownames.force = NA)
crit13 <- intCriteria(matrix, as.integer(GMM13$classification), "all")

GMM14 <- Mclust(Daten_norm[, c(1, 4)])
matrix <- data.matrix(Daten_norm[, c(1, 4)], rownames.force = NA)
crit14 <- intCriteria(matrix, as.integer(GMM14$classification), "all")

GMM23 <- Mclust(Daten_norm[, c(2, 3)])
matrix <- data.matrix(Daten_norm[, c(2, 3)], rownames.force = NA)
crit23 <- intCriteria(matrix, as.integer(GMM23$classification), "all")

GMM24 <- Mclust(Daten_norm[, c(2, 4)])
matrix <- data.matrix(Daten_norm[, c(2, 4)], rownames.force = NA)
crit24 <- intCriteria(matrix, as.integer(GMM24$classification), "all")

GMM34 <- Mclust(Daten_norm[, c(3, 4)])
matrix <- data.matrix(Daten_norm[, c(3, 4)], rownames.force = NA)
crit34 <- intCriteria(matrix, as.integer(GMM34$classification), "all")
```

```
GMM123 <- Mclust(Daten_norm[, c(1, 2, 3)])
matrix <- data.matrix(Daten_norm[, c(1, 2, 3)], rownames.force = NA)
crit123 <- intCriteria(matrix, as.integer(GMM123$classification), "all")

GMM124 <- Mclust(Daten_norm[, c(1, 2, 4)])
matrix <- data.matrix(Daten_norm[, c(1, 2, 4)], rownames.force = NA)
crit124 <- intCriteria(matrix, as.integer(GMM124$classification), "all")

GMM134 <- Mclust(Daten_norm[, c(1, 3, 4)])
matrix <- data.matrix(Daten_norm[, c(1, 3, 4)], rownames.force = NA)
crit134 <- intCriteria(matrix, as.integer(GMM134$classification), "all")

GMM234 <- Mclust(Daten_norm[, c(2, 3, 4)])
matrix <- data.matrix(Daten_norm[, c(2, 3, 4)], rownames.force = NA)
crit234 <- intCriteria(matrix, as.integer(GMM234$classification), "all")

GMM1234 <- Mclust(Daten_norm[, c(1, 2, 3, 4)])
matrix <- data.matrix(Daten_norm[, c(1, 2, 3, 4)], rownames.force = NA)
crit1234 <- intCriteria(matrix, as.integer(GMM1234$classification), "all")

# Vektor der Trace-Indizes je Merkmalskombination
tracewib<-c(crit1$trace_wib, crit2$trace_wib, crit3$trace_wib,
            crit4$trace_wib, crit12$trace_wib, crit13$trace_wib,
            crit14$trace_wib, crit23$trace_wib, crit24$trace_wib,
            crit34$trace_wib, crit123$trace_wib, crit124$trace_wib,
            crit134$trace_wib, crit234$trace_wib, crit1234$trace_wib)
```

2) Graphische Darstellung des Trace-Index

```
max_1 <- max(crit1$trace_wib, crit2$trace_wib, crit3$trace_wib,
             crit4$trace_wib)
max_1

# [1] 5.243777

max_2 <- max(crit12$trace_wib, crit13$trace_wib,
             crit14$trace_wib, crit23$trace_wib, crit24$trace_wib,
             crit34$trace_wib)
max_2

# [1] 26.17956

max_3 <- max(crit123$trace_wib, crit124$trace_wib,
             crit134$trace_wib, crit234$trace_wib)
max_3

# [1] 7.802783

# pdf(width = 6, height = 4, "Trace_Plot.pdf", encoding = "MacRoman")
plot(tracewib, ann=FALSE, axes=FALSE)
lines(tracewib)
box()
axis(1, at = seq(1:15), labels = c("1", "2", "3", "4", "12", "13", "14",
                                   "23", "24", "34", "123", "124", "134",
                                   "234", "1234"),
     cex.axis = 0.8, las = 2)
axis(2, at = seq(0, 30, 5), cex.axis = 0.8, las = 2)
mtext(side = 1, line = 2.7, "Merkmalskombinationen")
```

```
mtext(side = 2, line = 2.8, "Trace-Index")
abline(v = 4.5)
abline(v = 10.5)
abline(v = 14.5)
points(match(max_1, tracewib), max_1, pch = 19, col = "black")
points(match(max_2, tracewib), max_2, pch = 19, col = "black")
points(match(max_3, tracewib), max_3, pch = 19, col = "black")
abline(h = max_1, col = "black", lwd = 1)
abline(h = max_2, col = "black", lwd = 1)
abline(h = max_3, col = "black", lwd = 1)
```


Trace−Index
0
5
10
15
20
25
1
2
3
4
12
13
14
23
24
34
123
124
134
234
1234
Merkmalskombinationen

```
#dev.off()
```

9 Regressions und Korrelationsanalyse

9.1 Regressionsanalyse

```
# Beispielhafte Regressionsanalyse für den (linearen) Zusammenhang
# zwischen d13CKollagen (x) und d15NKollagen (y)
# lm(y ~ x)
Regression <- lm(Daten[, 7] ~ Daten[, 6])
summary(Regression)

#
# Call:
# lm(formula = Daten[, 7] ~ Daten[, 6])
#
# Residuals:
#     Min      1Q  Median      3Q     Max
# -5.0432 -1.6021  0.2276  1.5114  5.2671
#
# Coefficients:
#             Estimate Std. Error t value Pr(>|t|)
# (Intercept) 17.58163    1.41822  12.397 5.95e-16 ***
# Daten[, 6]   0.34371    0.07531   4.564 4.01e-05 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 2.303 on 44 degrees of freedom
# Multiple R-squared:  0.3213,Adjusted R-squared:  0.3059
# F-statistic: 20.83 on 1 and 44 DF,  p-value: 4.009e-05
```

9.2 Korrelationsanalyse

9.2.1 Marginale Korrelation

```
cor(Daten_norm)

#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen    1.0000000    0.5668361    0.7544161    0.4889518
# d15NKollagen    0.5668361    1.0000000    0.3734669    0.4529403
# d13CKarbonat    0.7544161    0.3734669    1.0000000    0.2237485
# d18OKarbonat    0.4889518    0.4529403    0.2237485    1.0000000

cor.test(Daten_norm[, 1], Daten_norm[, 2])

#
#  Pearson's product-moment correlation
#
# data:  Daten_norm[, 1] and Daten_norm[, 2]
# t = 4.564, df = 44, p-value = 4.009e-05
# alternative hypothesis: true correlation is not equal to 0
# 95 percent confidence interval:
#  0.3310055 0.7360209
# sample estimates:
#       cor
# 0.5668361

cor.test(Daten_norm[, 1], Daten_norm[, 3])

#
#  Pearson's product-moment correlation
#
# data:  Daten_norm[, 1] and Daten_norm[, 3]
```

```
# t = 7.6238, df = 44, p-value = 1.408e-09
# alternative hypothesis: true correlation is not equal to 0
# 95 percent confidence interval:
#  0.5942654 0.8570216
# sample estimates:
#       cor
# 0.7544161

cor.test(Daten_norm[, 1], Daten_norm[, 4])

#
#  Pearson's product-moment correlation
#
# data:  Daten_norm[, 1] and Daten_norm[, 4]
# t = 3.7181, df = 44, p-value = 0.0005647
# alternative hypothesis: true correlation is not equal to 0
# 95 percent confidence interval:
#  0.2315155 0.6823902
# sample estimates:
#       cor
# 0.4889518

cor.test(Daten_norm[, 2], Daten_norm[, 3])

#
#  Pearson's product-moment correlation
#
# data:  Daten_norm[, 2] and Daten_norm[, 3]
# t = 2.6705, df = 44, p-value = 0.01057
# alternative hypothesis: true correlation is not equal to 0
# 95 percent confidence interval:
```

```
#  0.09328225 0.59884064
# sample estimates:
#       cor
# 0.3734669

cor.test(Daten_norm[, 2], Daten_norm[, 4])

#
#  Pearson's product-moment correlation
#
# data:  Daten_norm[, 2] and Daten_norm[, 4]
# t = 3.37, df = 44, p-value = 0.001573
# alternative hypothesis: true correlation is not equal to 0
# 95 percent confidence interval:
#  0.1872653 0.6568682
# sample estimates:
#       cor
# 0.4529403

cor.test(Daten_norm[, 3], Daten_norm[, 4])

#
#  Pearson's product-moment correlation
#
# data:  Daten_norm[, 3] and Daten_norm[, 4]
# t = 1.5228, df = 44, p-value = 0.135
# alternative hypothesis: true correlation is not equal to 0
# 95 percent confidence interval:
#  -0.0711725  0.4826937
# sample estimates:
#       cor
# 0.2237485
```

9.2.2 Partielle Korrelation

Das R-Paket

- „ppcor“ (Version 1.1; Kim, 2015)

muss installiert sein.

```
# install.packages("ppcor")
library(ppcor)
```

```
pcor(Daten_norm[, 1:4])

# $estimate
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen    1.0000000   0.33376337   0.72814948    0.3833139
# d15NKollagen    0.3337634   1.00000000  -0.04068818    0.2277675
# d13CKarbonat    0.7281495  -0.04068818   1.00000000   -0.2373228
# d18OKarbonat    0.3833139   0.22776747  -0.23732284    1.0000000
#
# $p.value
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen 0.000000e+00   0.02682051 2.124657e-08   0.01021838
# d15NKollagen 2.682051e-02   0.00000000 7.931414e-01   0.13702348
# d13CKarbonat 2.124657e-08   0.79314139 0.000000e+00   0.12086418
# d18OKarbonat 1.021838e-02   0.13702348 1.208642e-01   0.00000000
#
# $statistic
#              d13CKollagen d15NKollagen d13CKarbonat d18OKarbonat
# d13CKollagen     0.000000    2.2946142    6.8847703     2.689594
# d15NKollagen     2.294614    0.0000000   -0.2639081     1.515948
```

```
# d13CKarbonat     6.884770   -0.2639081    0.0000000    -1.583260
# d18OKarbonat     2.689594    1.5159478   -1.5832602     0.000000
#
# $n
# [1] 46
#
# $gp
# [1] 2
#
# $method
# [1] "pearson"
```

10 Mischungsmodelle

10.1 „SISUS“

Für nähere Erklärungen zu dem R-Skript für „SISUS“ sei auf Erhardt (2014) und die unter https://statacumen.com/sisus/ verfügbare Anleitung (inkl. Beispieldaten) verwiesen.

1) Installation von „SISUS“

Das R-Paket

- „sisus“ (Version 3.9-14; Erhardt, 2014)

muss installiert sein.

Die aktuellste Version des R-Paketes „sisus“ (3.9-14) ist derzeit nicht direkt über CRAN verfügbar. Tatsächlich enthält die dort verfügbare, ältere „SISUS“-Version (3.9-13) wohl einen Fehler (persönliche Mitteilung: Dr. Erik Barry Erhardt), wodurch eine Berechnung mit weniger als drei Isotopensystemen nicht möglich ist.
Die aktuelle Version kann über https://statacumen.com/sisus/ heruntergeladen werden (sisus_3.9-14.tar.gz).
Nach **vorausgegangener** Installation von Perl und den R-Paketen „RColorBrewer“ (Version 1.1-2; Neuwirth, 2014), „coda“ (Version 0.19-2; Plummer et al., 2006), „gdata“ (Version 2.18.0; Warnes et al., 2017), „gtools“ (Version 3.8.1; Warnes et al., 2018), „moments“ (Version 0.14; Komsta & Novomestky, 2015), „polyapost“ (Version 1.5; Meeden et al., 2017), sowie „rcdd“ (Version 1.2; Geyer & Meeden, 2017) kann das SISUS-Paket über

```
install.packages("/Users/.../sisus_3.9-14.tar.gz")
```

installiert werden. Ab diesem Zeitpunkt kann die aktuelle SISUS-Version - wie jedes andere R-Paket auch - über

```
library(sisus)
```

geladen werden.

2) Berechnen des Mischungsmodelles

Das Einlesen der Daten von Konsumenten, Nahrungsquellen, sowie Konzentrationen (und ggf. Umrechnungsfaktoren) erfolgt über ein unter https://statacumen.com/sisus/ verfügbares Excel-Dokument und dem R-Befehl

```
sisus.run(Tabellenname)
```

10.2 „simmr“

Für nähere Erklärungen zu dem R-Skript für „simmr“ sei auf Parnell (2016) und die unter https://cran.r-project.org/web/packages/simmr/vignettes/simmr.html verfügbare Anleitung (inkl. Beispieldaten) verwiesen.

Das R-Paket

- „simmr“ (Version 0.3; Parnell, 2016)

muss installiert sein.

```
# install.packages(simmr)
library(simmr)
```

1) Einlesen der Daten

```
Konsumenten <- read.table("consumers.csv", header = TRUE, sep = ";")
Quellen <- read.table("sources.csv", header = TRUE, sep = ";")
Konzentration <- read.table("concentration.csv", header = TRUE, sep = ";")

mix <- matrix(c(consumerdata[, 1], consumerdata[, 2]),
              ncol = 2, nrow = 306)
colnames(mix) <- c("d13C", "d15N")
sourcesnames <- c("Quelle 1", "Quelle 2",
                  "Quelle 3", "Quelle 4")
sourcesmean <- matrix(c(sourcedata[, 2], sourcedata[, 4]),
                      ncol = 2, nrow = 4)
sourcessd <- matrix(c(sourcedata[, 3], sourcedata[, 5]),
                    ncol = 2, nrow = 4)
conc <- matrix(c(concdata[, 1], concdata[, 2]),
```

```
                ncol = 2, nrow = 4)


simmr_input <- simmr_load(mixtures = mix, source_names = sourcesnames,
                          source_means = sourcesmean, source_sds = sourcessd,
                          concentration_means = conc)
```

2) Graphische Darstellung der eingelesenen Daten

```
# pdf(width = 6,height = 6, "simmr_Input.pdf",
# encoding = "MacRoman")
plot(simmr_input,
     xlab = expression(paste(delta^13*"C"[Kollagen]~"[\211]")),
     ylab = expression(paste(delta^15*"N"[Kollagen]~"[\211]")))
# dev.off()
```

3) Berechnen des Mischungsmodelles

```
simmr_output <- simmr_mcmc(simmr_input,
                           mcmc.control = list(iter = 10000, burn = 1000,
                                               thin = 10, n.chain = 4))


# Ausgabe der Ergebnisse
summary(simmr_output, type = c("diagnostics", "quantiles", "statistics",
                               "correlations"))
```

3) Graphische Darstellung der Ergebnisse

```
# pdf(width = 8, height = 6, "simmr_Proportions_Plot.pdf",
# encoding = "MacRoman")
compare_sources(simmr_output)
# dev.off()

# pdf(width = 6,height=6, "simmr_Histogramm_Plot.pdf",
# encoding = "MacRoman")
plot(simmr_output, type = "histogram")
# dev.off()

# pdf(width = 6,height=6, "simmr_Dichte_Plot.pdf",
# encoding = "MacRoman")
plot(simmr_output, type = "density")
# dev.off()

# pdf(width = 6,height=6, "simmr_Matrix_Plot.pdf",
# encoding = "MacRoman")
plot(simmr_output, type = "matrix")
# dev.off()

# pdf(width = 6,height=6, "simmr_Boxplot_Plot.pdf",
# encoding = "MacRoman")
plot(simmr_output, type = "boxplot")
# dev.off()

# pdf(width = 6,height=6, "simmr_Konvergenz_Plot.pdf",
```

```
# encoding = "MacRoman")

plot(simmr_output, type = "convergence")

# dev.off()
```

10.3 „MixSIAR“

Für nähere Erklärungen zu dem R-Skript für „MixSIAR“ sei auf die unter https://github.com/brianstock/MixSIAR verfügbare Anleitung (Stock & Semmens, 2016, inkl. Beispieldaten) verwiesen.

Das R-Paket

- „MixSIAR“ (Version 3.1.10; Stock & Semmens, 2016)

muss installiert sein.

1) Installation von „MixSIAR“

Nach **vorausgegangener** Installation der R-Pakete „gWidgets“(Version 0.0-54; Verzani, 2014), „RGtk2“ (Version 2.20.35; Lawrence, 2010) und gWidgetsRGtk2“ (Version 0.0-86; Lawrence & Verzani, 2018) kann „MixSIAR“ installiert werden.

```
install.packages(c("gWidgets", "RGtk2", "gWidgetsRGtk2"))
# Installation von GTK+ (GUI-Toolkit; GUI $=$ "Graphical User
# Interface")
library(RGtk2)
# Neustart von R erforderlich
install.packages("MixSIAR", dependencies=TRUE)
library(MixSIAR)
```

Das Einlesen der Daten, sowie die Berechnung des Mischungsmodelles kann über

```
mixsiar_gui()
```

erfolgen. Durch diesen R-Befehl öffnet sich das MixSIAR GUI als separates Fenster. Alternativ wird nachfolgend das Einlesen und Berechnen mittels R-Befehlen erklärt.

2) Einlesen der Daten

```
mix<-load_mix_data(filename = "consumer_RUN.csv",
                   iso_names = c("d13C", "d15N"),
                   # NULL wenn kein Faktor aufgenommen wird
                   factors = "Nummer",
                   # TRUE: zufälliger Effekt, FALSE: fester Effekt
                   fac_random = FALSE,
                   fac_nested = FALSE,
                   cont_effects = NULL)

source <- load_source_data(filename = "sources_RUN.csv",
                           source_factors = NULL,
                           # Konzentrationsabhängigkeit
                           # TRUE: ja, FALSE: nein
                           conc_dep = TRUE,
                           data_type = "means", mix)
discr <- load_discr_data(filename = "discrimination_RUN.csv", mix)
```

3) Graphische Darstellung der Daten

```
plot_data(filename = "isospace_plot", plot_save_pdf = TRUE,
          plot_save_png=FALSE, mix, source,discr)

# Berechnen der normalisierten Oberfläche (für 2 Isotopensysteme)
if(mix$n.iso==2) calc_area(source = source, mix = mix, discr = discr)
```

```
# Prior-Plot
plot_prior(alpha.prior = 1, source)
```

4) Berechnen des Mischungsmodelles

```
model_filename <- "MixSIAR_model.txt"
resid_err <- FALSE # alternativ: TRUE (wenn FALSE, dann entspricht das
# "MixSIAR"-Modell dem "MixSIR"-Modell)
process_err <- TRUE # alternativ: FALSE
write_JAGS_model(model_filename, resid_err, process_err, mix, source)

# Überprüfen der Ergebnisse mit kürzerem Durchlauf
jags.test <- run_model(run = "test", mix,source, discr, model_filename,
                   alpha.prior = 1, resid_err, process_err)
# Modell-Berechnung
jags <- run_model(run = "short", mix,source, discr, model_filename,
                   alpha.prior = 1, resid_err, process_err)
```

5) Graphische Darstellung der Ergebnisse

```
# Auswahl der gewünschten Ausgabe über TRUE/FALSE
output_JAGS(jags, mix, source,
            output_options = list(summary_save = TRUE,
                                  summary_name = "summary_statistics",
                                  sup_post = FALSE,
                                  plot_post_save_pdf = TRUE,
                                  plot_post_name = "posterior_density",
                                  sup_pairs = FALSE,
```

```
                                plot_pairs_save_pdf = TRUE,
                                plot_pairs_name = "pairs_plot",
                                sup_xy = TRUE,
                                plot_xy_save_pdf = TRUE,
                                plot_xy_name = "xy_plot",
                                gelman = TRUE, heidel = TRUE,
                                geweke = TRUE,
                                diag_save = TRUE,
                                diag_name = "diagnostics",
                                indiv_effect = FALSE,
                                plot_post_save_png = FALSE,
                                plot_pairs_save_png = FALSE,
                                plot_xy_save_png = FALSE))
```

10.4 Simulierte Mischungspolygone

Nachfolgend findet sich das R-Skript von Smith et al. (2013)

(sh. auch http://www.famer.unsw.edu.au/downloads.html) mit einigen Erläuterungen.

Die R-Pakete

- „sp“ (Version 1.3-1; Pebesma & Bivand, 2005)
- „splancs“ (Version 2.01-40; Rowlingson & Diggle, 2017)

müssen installiert sein.

```
# install.packages(c("sp","splancs"))
rm(list = ls(all = TRUE))
graphics.off()
library(sp)
library(splancs)
```

1) Einlesen der Daten

```
# Quellen: der d13C-Wert muss vor dem d15N-Wert stehen
sources <- read.table("source.csv", header = T, sep = ";")
# Konsumenten: ein Fehler von "0" führt zu einer Fehlermeldung
mixture <- read.table("mixing.csv", header = T, sep = ";")
TEF <-  read.table("TEF.csv", header = T, sep = ";")
# Anzahl an Wiederholungen
its <- 1500
# Wahl der minimalen und maximalen Grenzen für d13C und d15N
# es sollten Werte außerhalb des 95 %-Bereiches gewählt werden
min_C <- -35
max_C <- -10
```

```
min_N <- 0
max_N <- 20
# Auflösung der Abbildung (beeinflusst Rechengeschwindigkeit)
res <- 250
```

2) Berechnen des Mischungsmodelles

```
step_C <- (max_C - min_C)/(res - 1)
step_N <- (max_N - min_N)/(res - 1)
C_g <- seq(min_C, max_C, by = step_C)
N_g <- seq(min_N, max_N, by = step_N)
# Erzeugen eines Rasters von Werten
mgrid <- function(a,b) {
  list(
    x = outer(b*0, a, FUN = "+"),
    y = outer(b, a*0, FUN = "+")
  )
}
m <- mgrid(C_g, N_g)
# Erzeugen von Dateien, um Daten zu speichern
Par_values <- array(0, c(its, (nrow(sources)*4+3)))
p <- array(0, c(its, (nrow(mixture))))
mix_reg <- array(0, c(res, res))
# Schleife mit "its" Wiederholungen zur Berechnung der Isotopensignaturen
# der Quellen
for (i in 1:its) {
  v <- array(0, c(nrow(sources),2))
  f <- array(0, c(nrow(TEF),2))
  for (j in 1:nrow(sources)) {
```

```
  v[j,1] <- rnorm(1, mean = sources[j, 1], sd = sources[j, 2])
  v[j,2] <- rnorm(1, mean = sources[j, 3], sd = sources[j, 4])
  f[j,1] <- rnorm(1, mean = TEF[j, 1], sd = TEF[j, 2])
  f[j,2] <- rnorm(1, mean = TEF[j, 3], sd = TEF[j, 4])
}
V <- v + f
# Erzeugen einer zwei-dimensionalen konvexen Hülle
hull <- chull(V)
hull_a <- append(hull, hull[1])
# Überprüfen, ob Punkte in das Polygon fallen
P <- point.in.polygon(mixture[, 1], mixture[, 2], V[hull_a, 1],
                      V[hull_a, 2])
P_n <- as.numeric(P)
p[i,] <- P_n
# Berechnen der Polygonfläche
poly_a <- areapl(V[hull_a,])
m$y_f <- m$y[res:1,]
# Berechnen, ob ein Datenpunkt im Polygon liegt
m_r <- point.in.polygon(m$x, m$y_f, V[hull_a, 1], V[hull_a, 2])
m_r_s <- matrix(m_r, nrow = res, byrow = F)
m_r_s[m_r_s > 1] <- 1
mix_reg <- mix_reg + m_r_s
vals <- c(v[, 1], v[, 2], f[, 1], f[, 2], 0, 0, 0)
Par_values[i,] <- vals
Par_values[i, ncol(Par_values)-2] <- poly_a
Par_values[i, ncol(Par_values)-1] <- i
Par_values[i, ncol(Par_values)] <- var(Par_values[1:i,
                                       ncol(Par_values)-2])
```

```
  if (i %% 10 == 0) cat(paste("iteration", i, "\n"))
}
```

3) Graphische Darstellung der Ergebnisse

```
# Varianz der Polygonfläche während der Simulation
Iterations <- Par_values[, ncol(Par_values)-1]
Variance <- Par_values[, ncol(Par_values)]
plot(Iterations, Variance, type = "n")  #plots
lines(Iterations, Variance, lty = 1, lwd = 1.5, col = "blue")

# Anteil der Wiederholungen, für die ein Konsument innerhalb des
# Mischungspolygons fällt
p[p > 1] <- 1
Probabilities <- colSums(p)/its
print(Probabilities)
windows()
barplot(Probabilities, xlab="Konsument",
        ylab="Wahrscheinlichkeit, dass Konsument im Polygon liegt",
        ylim=c(0,1), names.arg = seq(1, nrow(mixture), by = 1))

# Simuliertes Mischungspolygon
mix_reg <- mix_reg/its
mix_reg[mix_reg==0] <- NA
mix_regt <- t(mix_reg[ncol(mix_reg):1,])
windows()
image(C_g, N_g, mix_regt, col=colorRampPalette(c("blue", "light blue",
                                                 "green", "light green",
                                                 "yellow", "red"))(100),
```

```
      xlab="d13C", ylab="d15N", useRaster = TRUE)
cont <- c(0.05, seq(0.1, 1, by=0.1))
contour(C_g, N_g, mix_regt, levels = cont, add = TRUE, drawlabels = FALSE,
        lwd = 1.9)
sources_TEF <- sources + TEF
points(sources_TEF[, 1], sources_TEF[, 3], col = "white", pch = 4,
       lwd = 2, cex = 1.5)
points(mixture, pch = 19, cex = 1.3)
dev.copy2pdf(file="Mix_Region.pdf")

# Farbskala für die Abbildung
windows()
cust_color <- colorRampPalette(c("blue", "light blue",
                                 "green", "light green",
                                 "yellow", "red"))
z <- matrix(1:100, nrow = 1)
x <- 1
y <- seq(0, 1, len = 100)
image(x, y, z, col = colorRampPalette(c("blue", "light blue",
                                        "green", "light green",
                                        "yellow", "red"))(100),
      xaxt = "n", xlab = "", ylab = "", useRaster = TRUE, bty = "n",
      las = 1)

# Simuliertes Mischungspolygon (schwarz-weiß)
windows()
plot(C_g, N_g, type = "n", xlab = "d13C", ylab = "d15N")
cont <- c(0.05, seq(0.1, 1, by = 0.1))
```

```
contour(C_g, N_g, mix_regt, levels = cont, add = TRUE, drawlabels = FALSE,
        lwd = 1.9)
sources_TEF <- sources + TEF
points(sources_TEF[, 1], sources_TEF[, 3], col = "black", pch = 4,
       lwd = 2, cex = 1.5)
points(mixture, pch = 19, cex = 1.3)

# Biplot mit eingezeichneter 95 %-Konturlinie
windows()
plot(C_g, N_g, type = "n", xlab = "d13C", ylab = "d15N")
cont <- c(0.05)
contour(C_g, N_g, mix_regt, levels = cont, add = TRUE, drawlabels = FALSE,
        lwd = 1.9)
sources_TEF <- sources + TEF
points(sources_TEF[, 1], sources_TEF[, 3], col = "black",
       pch = 15, lwd = 2, cex = 1.5)
arrows(sources_TEF[, 1]-sources_TEF[, 2], sources_TEF[, 3],
       sources_TEF[, 1]+sources_TEF[, 2], sources_TEF[, 3],
       length = 0, angle = 90, code = 3)
arrows(sources_TEF[, 1], sources_TEF[, 3] - sources_TEF[, 4],
       sources_TEF[, 1], sources_TEF[, 3] + sources_TEF[, 4],
       length = 0, angle = 90, code=3)
points(mixture, pch = 1, cex = 1.3)
# Bezeichnung der Quellen anpassen!
labels <- c("Quelle 1", "Quelle 2", "Quelle 3", "Quelle 4",
            "...")
text(sources_TEF[, 1], sources_TEF[, 3], labels = labels, pos = 3)
```

4) Abspeichern der Ergebnisse

```
p_a <- rbind(p, Probabilities)
write.table(p_a, file = "Consumer_Probabilities_Test.csv",
            sep = ",", row.names = FALSE)
col_names <- c(rep("d13C",nrow(sources)),
               rep("d15N", nrow(sources)),
               rep("13C_TEF", nrow(sources)),
               rep("15N_TEF",nrow(sources)),
               "Poly_Area", "Iteration", "Variance")
col_nums <- c(rep(1:nrow(sources), 4), 0, 0, 0)
col_n <- paste(col_names, col_nums)
write.table(Par_values, file = "Parameter_Values_Test.csv",
            sep = ";", row.names = FALSE, col.names = col_n)
```

11 Literatur

Desgraupes, B. (2018). *clusterCrit: Clustering Indices*. URL: https://CRAN.R-project.org/package=clusterCrit.

Erhardt, E. B. (2014). *SISUS: Stable Isotope Sourcing using Sampling*. URL: https://CRAN.R-project.org/package=sisus.

Filzmoser, P., Gschwandtner, M. (2017). *mvoutlier: Multivariate outlier detection based on robust methods*. URL: http://CRAN.R-project.org/package=mvoutlier.

Geyer, C. J., Meeden, G. D. (2017). *rcdd: Computational Geometry*. URL: https://CRAN.R-project.org/package=rcdd.

Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A. (2004). „kernlab - An S4 Package for Kernel Methods in R“. *Journal of Statistical Software* 11.9, S. 1–20.

Kassambara, A. (2018). *ggpubr: 'ggplot2' Based Publication Ready Plots*. URL: https://CRAN.R-project.org/package=ggpubr.

Kim, S. (2015). „ppcor: An R Package for a Fast Calculation to Semi-partial Correlation Coefficients“. *Communications for statistical applications and methods* 22.6, S. 665–674.

Komsta, L., Novomestky, F. (2015). *moments: Moments, cumulants, skewness, kurtosis and related tests*. URL: https://CRAN.R-project.org/package=moments.

Lawrence, M. (2010). „RGtk2: A Graphical User Interface Toolkit for R“. *Journal of Statistical Software* 37.8, S. 1–52.

Lawrence, M., Verzani, J. (2018). *gWidgetsRGtk2: Toolkit Implementation of gWidgets for RGtk2*. URL: https://CRAN.R-project.org/package=gWidgetsRGtk2.

Meeden, G., Lazar, R., Geyer, C. J. (2017). *polyapost: Simulating from the Polya Posterior*. URL: https://CRAN.R-project.org/package=polyapost.

Neuwirth, E. (2014). *RColorBrewer: ColorBrewer palettes*. URL: http://cran.r-project.org/packages/RColorBrewer.

Parnell, A. (2016). *simmr: A Stable Isotope Mixing Model*. URL: https://CRAN.R-project.org/package=simmr.

Pebesma, E. J., Bivand, R. S. (2005). „Classes and methods for spatial data in R: the sp Package“. *R News* 5.2, S. 9–13.

Plummer, M., Best, N., Cowles, K., Vines, K. (2006). „CODA: Convergence Diagnosis and Output Analysis for MCMC“. *R News* 6, S. 7–11.

Rowlingson, B., Diggle, P. (2017). *splancs: Spatial and Space-Time Point Pattern Analysis*. URL: `https://CRAN.R-project.org/package=splancs`.

Scrucca, L., Fop, M., Murphy, T. B., Raftery, A. E. (2016). „mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models“. *The R journal* 8.1, S. 289–317.

Smith, J. A., Mazumder, D., Suthers, I. M., Taylor, M. D. (2013). „To fit or not to fit: evaluating stable isotope mixing models using simulated mixing polygons“. *Methods in Ecology and Evolution* 4.7, S. 612–618.

Stock, B. C., Semmens, B. X. (2016). „Unifying error structures in commonly used biotracer mixing models“. *Ecology* 97.10, S. 2562–2569.

Verzani, J. (2014). *gWidgets: gWidgets API for building toolkit-independent, interactive GUIs*. URL: `https://CRAN.R-project.org/package=gWidgets`.

Vu, V. Q. (2011). *ggbiplot: A ggplot2 based biplot*. URL: `http://github.com/vqv/ggbiplot`.

Walesiak, M., Dudek, A. (2017). *clusterSim: Searching for Optimal Clustering Procedure for a Data Set*. URL: `https://CRAN.R-project.org/package=clusterSim`.

Warnes, G. R., Bolker, B., Gorjanc, G., Grothendieck, G., Korosec, A., Lumley, T., MaxQueen, D., Magnusson, A., Rogers, J. (2017). *gdata: Various R Programming Tools for Data Manipulation*. URL: `https://CRAN.R-project.org/package=gdata`.

Warnes, G. R., Bolker, B., Lumley, T. (2018). *gtools: Various R Programming Tools*. URL: `https://CRAN.R-project.org/package=gtools`.

Wickham, H. (2016). *ggplot2: elegant graphics for data analysis*. New York: Springer.

Wickham, H., Hester, J., Chang, W. (2018a). *devtools: Tools to Make Developing R Packages Easier*. URL: `https://CRAN.R-project.org/package=devtools`.

Wickham, H., François, R., Henry, L., Müller, K. (2018b). *dplyr: A Grammar of Data Manipulation*. URL: `https://CRAN.R-project.org/package=dplyr`.

Xie, Y. (2018). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. URL: `https://CRAN.R-project.org/package=knitr`.

A Appendix

Tabelle A1: Übersicht über den Teildatensatz der Fische aus Haithabu und Schleswig (Datensatz I).

Spezies	Probenmr.	Habitat	Ernährung	Fundort	d13CKollagen	d15NKollagen	d13CKarbonat	d18OKarbonat
Brachse	35 B2Op	Süß-, Brackwasser	omnivor	Schleswig	-27,73	6,02	-0,64	-13,04
Brachse	36 B3Pop	Süß-, Brackwasser	omnivor	Schleswig	-25,07	6,17	-1,66	-11,68
Brachse	37 B4Pop	Süß-, Brackwasser	omnivor	Schleswig	-22,41	5,38	-1,60	-12,01
Brachse	38 B5C	Süß-, Brackwasser	omnivor	Schleswig	-27,37	10,72	-4,46	-15,07
Dorsch	1 D1V	Brackwasser, Salzwasser	omnivor	Haithabu	-16,06	13,68	-1,82	-13,20
Dorsch	3 D3V	Brackwasser, Salzwasser	omnivor	Haithabu	-16,21	15,89	-1,30	-3,76
Dorsch	4 D4V	Brackwasser, Salzwasser	omnivor	Haithabu	-14,74	15,57	-0,21	-3,90
Dorsch	5 D5V	Brackwasser, Salzwasser	omnivor	Haithabu	-15,29	11,99	-0,32	-4,00
Dorsch	40 D2V	Brackwasser, Salzwasser	omnivor	Schleswig	-15,64	12,24	2,18	-13,38
Dorsch	42 D4V	Brackwasser, Salzwasser	omnivor	Schleswig	-15,94	17,37	-0,38	-3,99
Flussbarsch	6 FB1C	Süß-, Brackwasser	carnivor	Haithabu	-15,88	12,23	1,45	-7,10
Flussbarsch	7 FB2C	Süß-, Brackwasser	carnivor	Haithabu	-11,66	10,85	3,53	-6,73
Flussbarsch	8 FB3C	Süß-, Brackwasser	carnivor	Haithabu	-15,72	15,28	3,37	-11,03
Flussbarsch	9 FB4C	Süß-, Brackwasser	carnivor	Haithabu	-11,50	11,69	3,45	-13,10
Flussbarsch	44 FB1C	Süß-, Brackwasser	carnivor	Schleswig	-17,22	12,32	-3,58	-10,96
Flussbarsch	45 FB2C	Süß-, Brackwasser	carnivor	Schleswig	-13,01	9,02	0,54	-9,14
Flussbarsch	46 FB3Op	Süß-, Brackwasser	carnivor	Schleswig	-16,21	10,35	-0,83	-8,03
Flussbarsch	47 FB4Pop	Süß-, Brackwasser	carnivor	Schleswig	-23,95	9,71	-7,20	-9,35
Flussbarsch	48 FB5Pop	Süß-, Brackwasser	carnivor	Schleswig	-13,45	11,30	-0,65	-8,73
Hecht	10 H1C	Süß-, Brackwasser	piscivor	Haithabu	-18,62	13,38	-4,25	-9,01
Hecht	11 H2C	Süß-, Brackwasser	piscivor	Haithabu	-25,66	10,77	-7,53	-11,54
Hecht	12 H3C	Süß-, Brackwasser	piscivor	Haithabu	-24,66	10,25	-7,20	-9,64
Hecht	13 H4C	Süß-, Brackwasser	piscivor	Haithabu	-22,23	11,63	-6,77	-9,36
Hecht	14 H5C	Süß-, Brackwasser	piscivor	Haithabu	-16,32	14,43	-3,28	-6,51
Hecht	49 H1C	Süß-, Brackwasser	piscivor	Schleswig	-22,40	11,52	-6,23	-12,49

Fortsetzung von Tabelle A1

Spezies	Probennr.	Habitat	Ernährung	Fundort	d13CKollagen	d15NKollagen	d13CKarbonat	d18OKarbonat
Hecht	51 H3C	Süß-, Brackwasser	piscivor	Schleswig	-22,38	5,40	-6,89	-12,80
Hecht	52 H4C	Süß-, Brackwasser	piscivor	Schleswig	-20,16	9,65	-3,20	-12,49
Hecht	53 H5De	Süß-, Brackwasser	piscivor	Schleswig	-24,31	8,02	-5,51	-13,82
Hornhecht	54 HH1De	Brackwasser, Salzwasser	piscivor	Schleswig	-13,47	11,52	2,12	-5,07
Hornhecht	58 HH5De	Brackwasser, Salzwasser	piscivor	Schleswig	-14,31	7,62	-1,84	-7,45
Schellfisch	59 SF1C	Salzwasser	carnivor	Schleswig	-15,95	13,29	1,37	-4,86
Schellfisch	60 SF2C	Salzwasser	carnivor	Schleswig	-15,93	12,91	-0,03	-10,23
Schellfisch	61 SF3C	Salzwasser	carnivor	Schleswig	-14,90	11,99	4,10	-11,61
Schellfisch	63 SF5C	Salzwasser	carnivor	Schleswig	-16,09	14,98	4,06	-15,06
Schleie	64 SLOp	Süß-, Brackwasser	omnivor	Schleswig	-24,79	6,34	-3,23	-14,49
Scholle	65 SOC	Brackwasser, Salzwasser	carnivor	Schleswig	-14,90	10,70	2,77	-14,74
Stör	66 ST1Rü	Süß-, Brack-, Salzwasser	carnivor	Schleswig	-13,52	12,80	3,29	-12,39
Stör	67 ST2Rü	Süß-, Brack-, Salzwasser	carnivor	Schleswig	-14,27	12,91	3,73	-7,58
Stör	68 ST3Rü	Süß-, Brack-, Salzwasser	carnivor	Schleswig	-13,73	13,52	4,80	-6,27
Stör	69 ST4Rü	Süß-, Brack-, Salzwasser	carnivor	Schleswig	-14,93	12,89	0,27	-5,33
Stör	67 ST5Rü	Süß-, Brack-, Salzwasser	carnivor	Schleswig	-13,72	13,18	-0,91	-6,41
Zander	15 Z1C	Süß-, Brackwasser	piscivor	Haithabu	-22,67	9,67	-7,93	-10,22
Zander	16 Z2C	Süß-, Brackwasser	piscivor	Haithabu	-19,96	10,29	-6,67	-10,41
Zander	17 Z3C	Süß-, Brackwasser	piscivor	Haithabu	-20,99	11,48	-6,46	-14,55
Zander	18 Z4C	Süß-, Brackwasser	piscivor	Haithabu	-22,94	10,58	-6,43	-13,94
Zander	19 Z5C	Süß-, Brackwasser	piscivor	Haithabu	-22,21	10,17	-7,07	-9,58