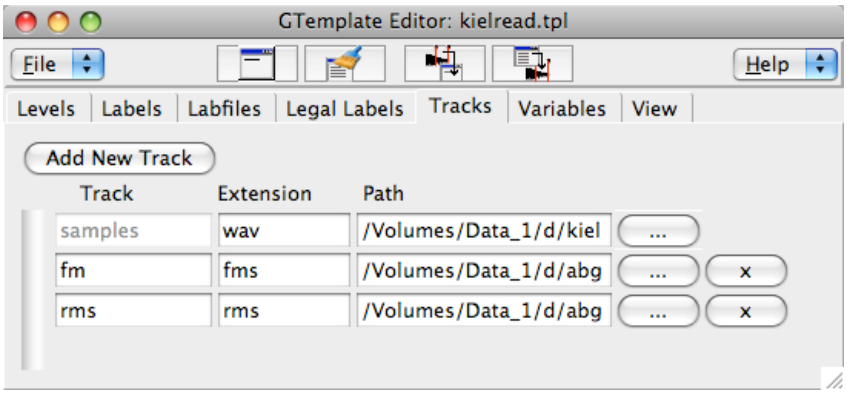
Manipulation und Abbildung von Trackdateien

0. Allgmein

Eine Trackdatei enthält Signalwerte (Tracks) zwischen den Start- und Endzeiten von Segmenten.

0.1 Welche Signale (Tracks) sind als Trackdateien in R erstellbar?

Die möglichen Signale (oder Tracks), die sich in R als Trackdateien einlesen lassen, sind in der Template-Datei bestimmt, z.B. für die `kielread` Sprachdatenbank unten:

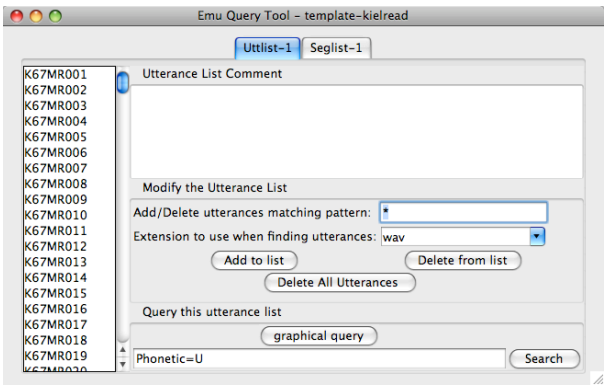


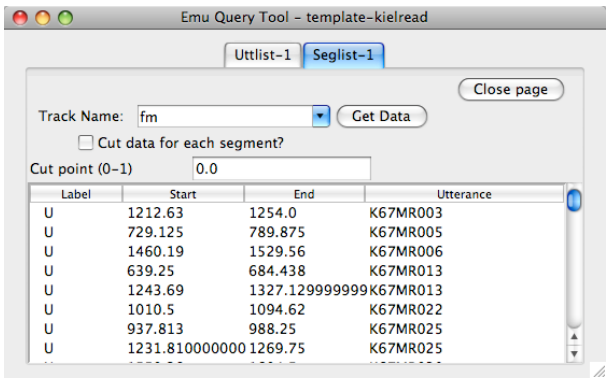
wären die Tracks `samples`, `fm`, `rms` (Zeitsignale, Formanten, Intensität) vorhanden.

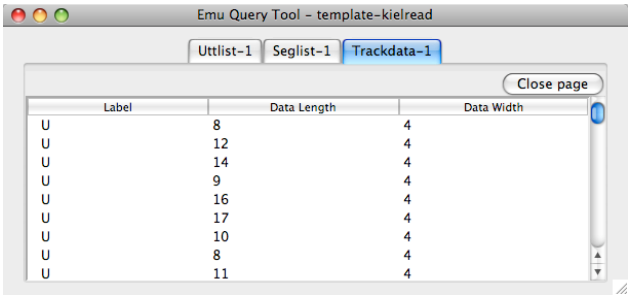
0.2 Wie wird eine Trackdatei erstellt?

Eine Trackdatei kann **nur von einer Segmentliste erstellt werden**, und es gibt zwei Möglichkeiten dies zu tun.

(1) Die Trackdatei wird in Emu abgefragt und in R mit der read.trackdata() Funktion eingelesen.







Dann `save Eigene Dateien -> form.txt`. R starten, library(emu):

fwerte = read.trackdata("H:/form.txt")

Es gibt aber eine identische, und direktere Methode, die Trackdatei mit emu.track() in R zu erzeugen:

seg.U = emu.query("kielread", "*", "Phonetic=U")
fwerte2 = emu.track(seg.U, "fm")

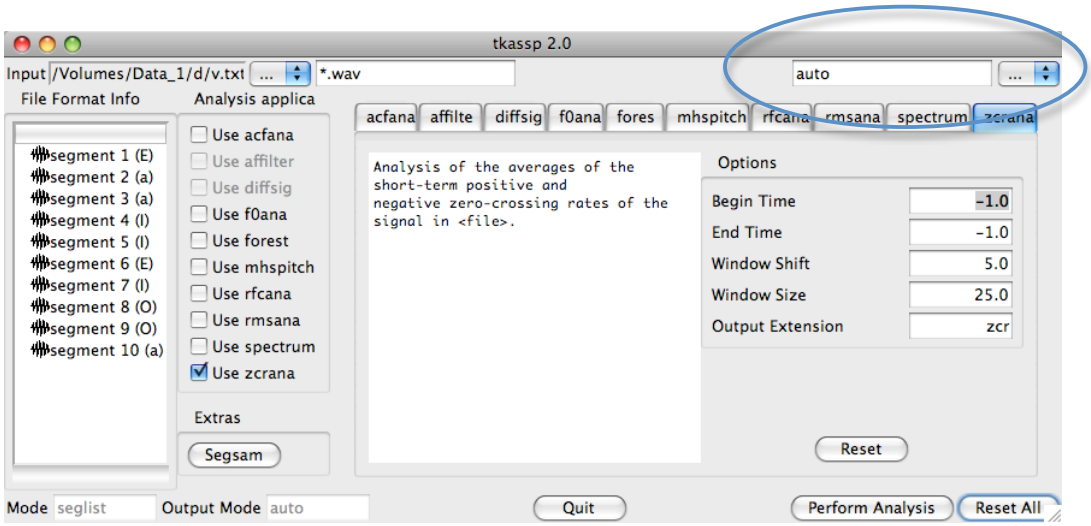
0.3 Was ist wenn die Signale in der Sprachdatenbank nicht vorhanden sind?
zB wenn ich ZCR-Daten für eine Segmente-Liste will.

Zwei Möglichkeiten
1. ZCR für die Sprachdatenbank berechnen und die Template-Datei ändern, sodass ZCR also Parameter definiert wird.

2. Die Segmentliste direkt in EMU-tkassp einlesen.
zB wenn ich ZCR für die ersten 10 Segmente in vowlax berechnen will.

# Diese Segmente exportieren
write.emusegs(vowlax[1:10,], "H:/v.txt")

Die Segmentliste in Emu-tkassp importieren



Verzeichnis eingeben, wo die Trackdatei gespeichert werden soll: zB `H:`

# Trackdatei in R einlesen
vowlax.zcr = read.trackdata("H:/v-zcr.txt")

0.4 Woran lässt sich in R eine Trackdatei erkennen: class(), is.trackdata(), summary()

Mit der (generischen) class() Funktion[1]:
class(vowlax.fdat)
class(vowlax.fund)
class(vowlax.zcr)
```
[1] "trackdata"
```

is.trackdata(vowlax.fdat) [2]

Die (generische) summary() Funktion gibt eine Zusammenfassung der Informationen in einer Trackdatei:

summary(vowlax.fdat)
summary(coutts.epg)
summary(vowlax.zcr)
```
Emu track data from 10 segments

Data is  1 dimensional from track data
Mean data length is  12.2  samples
```

1. Struktur einer Trackdatei

Eine Trackdatei ist zugleich eine Liste:

is.list(vowlax.zcr)

die sich aber auch in vielen Hinsichten wie eine Matrix behandeln lässt.

1.1 Eine Trackdatei als Liste

Eine Trackdatei ist **eine Liste**, die aus Komponenten $index, $ftime, $data besteht.

$data enthält die **Frames der Trackdaten** als eine Matrix. zB wenn ein Segment eine Dauer von 50 ms hat und ich wende mit Emu-tkassp die Signalverarbeitung mit einer Fensterverschiebung von 5 ms, dann wird für dieses Segment in der Trackdatei ca. 50/5 = 10 Frames sein. Jeder Frame bildet eine Reihe in $data

$index enthält die **Indizierung** (als eine Matrix), um die Frames dem richtigen Segment zuzuordnen

$ftime enthält den **Zeitpunkt vom ersten Frame** und den **Zeitpunkt vom letzten Frame** (auch als eine Matrix) pro Segment.

---

[1] angenommen library(emu) ist schon gestartet worden
[2] (Siehe auch as.trackdata() um selbst aus R-Daten eine Trackdatei zu gestalten).

Die bereits vorhandene Trackdatei vowlax.fdat wurde aus der Segmentliste vowlax erstellt:
vowlax.fdat = emu.track(vowlax, "fm")

Die entsprechende Segmentliste
vowlax[10,]

```
segment  list from database:
kielread
query was:  Kanonic=a | E | I | O
   labels   start     end     utts
10      a 1835.94 1884.81 K67MR003
```

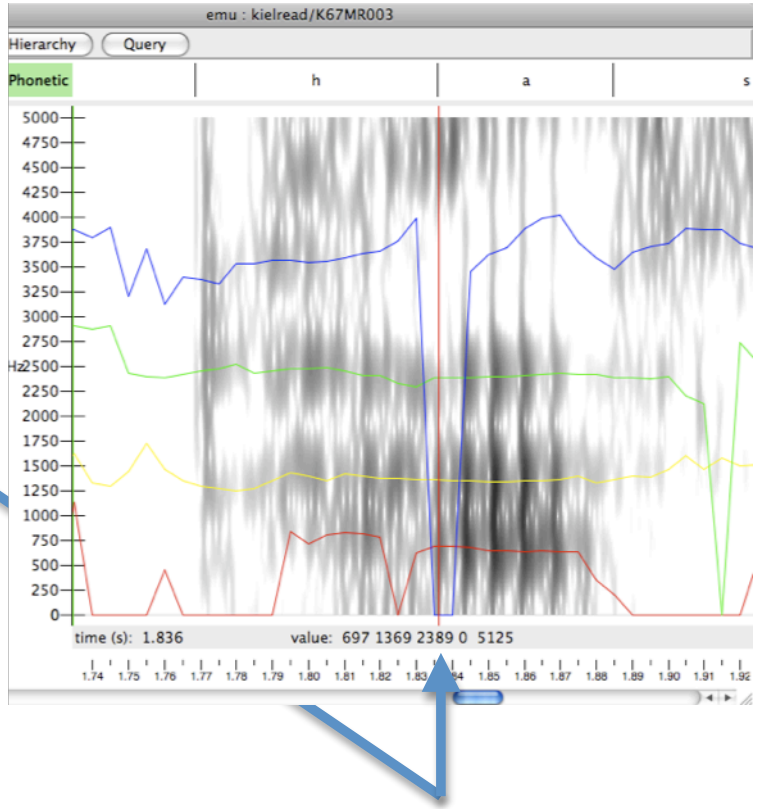
Die Formant-Werte vom 10en Segment sind hier enthalten:
vowlax.fdat[10,]

```
trackdata from track: fm
index:
 left right
    1    10
ftime:
      start    end
[1,] 1837.5 1882.5
data:
       T1   T2   T3   T4
1837.5 697 1369 2389    0
1842.5 698 1363 2397    0
1847.5 686 1359 2391 3457
1852.5 650 1349 2398 3628
1857.5 649 1352 2408 3700
1862.5 641 1354 2415 3892
1867.5 653 1361 2429 3997
1872.5 645 1371 2436 4030
1877.5 640 1408 2422 3756
1882.5 355 1330 2426 3602
```

Die Zeiten vom ersten und letzten Frame
vowlax.fdat[10,]$ftime

Die Frames:
vowlax.fdat[10,]$data

Die Zeiten, zu denen die Frames vorkommen
tracktimes(vowlax.fdat[10,])



Frage: Wieso sind die Start- und Endzeiten der Segmentliste:
start(vowlax[10,])
1835.94
end(vowlax[10,])
1884.81

nicht dieselben wie der erste und letzte Frame dieser Trackdatei
start(vowlax.fdat[10,])
1837.5
end(vowlax.fdat[10,])
1882.5

1.2 Eine Trackdatei mit Eigenschaften einer Matrix
Meistens wird der Benutzer nicht auf die Komponente $data, $ftime, $index zugreifen müssen, **weil sich eine Trackdatei in vielen Hinsichten wie eine Matrix behandeln lässt**[3].

> ***Was vor dem Komma erscheint, bezieht sich auf die Reihen; was nach dem Komma erscheint, auf die Spalten***

Die ZCR-Daten für den 5en Segment
vowlax.zcr[5,]

Die ZCR-Daten für die ersten 9 Segmente?

# Anzahl der Reihen, Spalten, Dimensionen
nrow(vowlax.zcr)

ncol(vowlax.zcr)

dim(vowlax.zcr)

# Wieso gibt es hier 4 Spalten?
ncol(vowlax.fdat)

F1 und F2 vom 10en Segment

Alle Formanten außer F3 der Segmente 12, 18, und 20?

F1 und F2 von allen Segmenten?

[3] Wegen einer object-oriented Implementierung und genauer wegen der Funktion [.trackdata(). Geben Sie `[.trackdata` ein, wenn Sie diese Funktion sehen möchten.

Der Zugriff auf Segmente von Trackdateien kann auch - wie bei einer Matrix - **durch logische Vektoren** erfolgen. zB. Diese Objekte sind zueainder alle parallel:

vowlax	Segmentliste
vowlax.fdat	Trackdatei, F1-F4
vowlax.fund	Grundfrequenz
vowlax.l	Vokal-Etikettierungen
vowlax.spkr	Sprecher-Etikettierungen

```
# Daher sind alle Objekte von derselben Länge
nrow(vowlax.fdat) == nrow(vowlax)
nrow(vowlax.fdat) == nrow(vowlax.fund)
nrow(vowlax.fund)==length(vowlax.l)
nrow(vowlax.fund)==length(vowlax.spkr)

# Trackdatei F1-F4 der "I" Vokale?
temp = vowlax.l=="I"
?

# Trackdatei F1 und F3 der "I" und "a" Vokale?

# Trackdatei der Grundfrequenz, "a" Vokale, Sprecherin "68"?
```

2. Abbildungen von Trackdateien
plot(): um einen Track, oder Tracks **einzelner Segmente** abzubilden
dplot() um Track(s) **mehrerer Segmente aufeinander zu überlagern**

```
plot(vowlax.zcr[5,])
```

Die meisten Defaults mit par() können auch verwendet werden- siehe: 0.1 Einige Parameter für R-Abbildungen in

http://www.phonetik.uni-muenchen.de/~jmh/lehre/sem/ws0809/R/Rabb1.pdf

zB

```
plot(vowlax.zcr[5,], main="Nulldurchgangsdichte", ylab="Frequenz (Hz)", xlab="Dauer (ms)", type="b")
```

Siehe help(plot.trackdata) für weitere Argumente. zB um die Startzeit in der Abbildung auf 0 ms zu setzen:

```
plot(vowlax.zcr[5,], main="Nulldurchgangsdichte", ylab="Frequenz (Hz)", xlab="Dauer (ms)", type="b", time=0)
```

Mit plot() lassen sich Tracks mehrerer Segmente (einzeln) abbilden. Dann aber contig=F (contiguous is False) hinzufügen, damit plot() weiß, es handelt sich nicht um aufeinanderfolgende Segmente derselben Äußerung.

```
par(mfrow=c(2,5))
plot(vowlax.fdat[1:10,], type="l", ylab="Frequenz (Hz)", xlab="Time (ms)", contig=F)
```

Mit dplot() werden die Segmente in derselben Abbildung aufeinander überlagert. Viele der Parameter die in help(par) gelistet sind (xlim, ylim, main, xlab, ylab) funktonieren hier auch.

```
# F2 der ersten 10 Segmente
dplot(vowlax.fdat[1:10,2])

# Zweites Argument: Vektor nach Etikettierung kodieren
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10])

# Unterschiedliche Linien
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], lty=T)

# schwarz-weiss mit verschiedenen Plot-Symbolen
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], col=F, type="b",  pch=1:4)
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], col=F, type="p",  pch=1:4)

# schwarz-weiss mit verschiedenen Plot-Symbolen 2 Grau-Stufen,
# unterschiedliche Liniendichten
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], type="b", pch=1:4, lwd=c(1,2,1,2),
col=c("black", "black", "slategray", "slategray")  )

# synchronisiert zum zeitlichen Mittelpunkt (0 ms)
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10],  0.5)

# synchronisiert 30 ms nach dem Vokalonset
times = start(vowlax.fdat) + 30
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10],  times[1:10], prop=F)

# gemittelt ('ensemble averaged')
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], average=T)
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], 0.5, average=T)

# linear zeit-normalisiert
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], norm=T, ylab="Proportionale Dauer")

# linear zeit-normalisiert und gemittelt
dplot(vowlax.fdat[1:10,2], vowlax.l[1:10], norm=T, average=T)
```

3. Funktionen auf eine Trackdatei anwenden

3.1 Funktion auf $data (Frames) anwenden

Oft können Funktionen auf die Frames von einer Trackdatei angewandt werden, ohne berücksichtigen zu müssen, dass die Frames in verschiedene Segmente aufgeteilt sind.

zB wir wollen die ZCR-Hz Werte in Logarithmen umsetzen

```
# Trackdatei kopieren (wenn man nicht die Trackdatei überschreiben will)
vowlax.logzcr = vowlax.zcr
# Funktion (in diesem Fall log() ) auf die Frames anwenden
vowlax.logzcr$data = log(vowlax.zcr$data)
```

```
# Abbilden
par(mfrow=c(1,2))
plot(vowlax.logzcr[1,], main="Log ZCR", type="l")
plot(vowlax.zcr[1,], main="ZCR", type="l")
```

Zweites Beispiel. Eine neue Trackdatei aus F1 - f0

```
# Eine neue Trackdatei erzeugen aus F1
vowlax.neu = vowlax.fdat[,1]
# Die f0-Frames von den F1-Frames abziehen
vowlax.neu$data = vowlax.neu$data - vowlax.fund$data
```

3.2 Wenn man zwischen Segmenten trennen muss

z.B. wie bekomme ich den ZCR-Mittelwert der ersten 5 Segmente?

Hier sind die ZCR-Werte (Frames) vom ersten Segmente

```
vowlax.zcr[1,]$data
```

```
T1
857.5 1031.6328
862.5  753.8822
867.5  678.5849
872.5  742.2772
877.5  667.3444
882.5  631.6155
887.5  610.0757
892.5  628.8128
897.5  554.3763
902.5  523.5935
907.5  561.9388
912.5  546.8322
917.5  461.4215
922.5  480.5934
927.5  493.7523
932.5  504.6808
937.5  546.0294
942.5  600.9991
```

Der Mittelwert ist daher

```
mean(vowlax.zcr[1,]$data)
```

Die Mittelwerte für die ersten 5 Segmente könnten mit einer for-Schleife errechnet werden:

```
ergebnis = NULL
for(j in 1:5){
ergebnis = c(ergebnis, mean(vowlax.zcr[j,]$data))
ergebnis
}
ergebnis
```

```
[1] 612.1357 806.5225 884.9094 435.0441 730.2391
```

Eine einfachere Weise das gleiche Ergebnis zu bekommen, ohne eine for-schleife selbst erstellen zu müssen, ist mit der trapply(Trackdatei, fun) Funktion. Diese Funktion wendet eine Funktion, fun() , auf die Trackdatei an, getrennt pro Segment:

```
trapply(vowlax.zcr[1:5,], mean, simplify=T)
```

```
[1] 612.1357 806.5225 884.9094 435.0441 730.2391
```

fun() kann eine beliebige Funktion in trapply(Trackdatei, fun) sein (auch eine selbst erstellte), die sich auf die Frames von einem Segment anwenden lässt. zB sd() könnte so eine Funktion sein, da

```
sd(vowlax.zcr[1,]$data)
```

anwendbar ist.

```
# F1-Standardabweichung, Segmente 20:30
trapply(vowlax.fdat[20:30,1], sd, simplify=T)

# F2-Minimum alle Segmente
trapply(vowlax.fdat[,2],  min, simplify=T)

# F1-Bereich Segmente 4, 8, 12
trapply(vowlax.fdat[c(4, 8, 12), 1], range, simplify=T)
```

usw.

Per Default gibt trapply() **eine Liste** zurück und simplify=T wandelt die Liste in einen Vektor oder eine Matrix um. simplify=T kann angewandt werden, wenn man weiß, dass die Funktion **einen Wert** (mean(), sd(), min(), max(), ...) oder **eine Reihe** (range()) **pro Segment** zurückgibt. Dies ist nicht immer der Fall. zB diff() berechnet die erste Ableitung:

```
a = c(9, 10, 15)
diff(a)
```

und da diff()auf die Frames von einem Segment angewandt werden kann:

```
diff(vowlax.zcr[1,]$data)
```

ist diff() daher mit der trapply() Funktion anwendbar. Da die Anzahl der Werte pro Segment unterschiedlich sein wird (da Segmente nicht dieselbe Dauer haben) benötigen wir eine Liste:

```
d = trapply(vowlax.zcr[1:5,], diff)
```

Es gibt eine Funktion buildtrack() um Listen dieser Art in Trackdateien umzubauen:

```
dneu = buildtrack(d)

par(mfrow=c(1,2))
plot(vowlax.zcr[1,], type="b", main="ZCR", ylab="Frequenz (Hz)")
plot(dneu[1,], type="b", main = "Erste Ableitung von ZCR")
```

4. Werte aus einer Trackdatei schneiden

4.1 Zwischen zwei Zeitpunkten

In diesem Fall ist die Ausgabe **eine Trackdatei**

```
# Proportional: Die ersten und letzten 20% einer Trackdatei wegschneiden
vowcut = dcut(vowlax.fdat, 0.2, 0.8, prop=T)

par(mfrow=c(1,2))
```

```
plot(vowlax.fdat[1,])
plot(vowcut[1,])

# Selbe x-Skala
xlim = c(850, 950)
plot(vowlax.fdat[1,], xlim=xlim)
plot(vowcut[1,], xlim=xlim)


# Die ersten 10 ms und die letzten 15 ms wegschneiden
links = start(vowlax.fdat)+10
rechts = end(vowlax.fdat) - 10
vowcut2 = dcut(vowlax.fdat, links, rechts)
plot(vowlax.fdat[1,], xlim=xlim)
plot(vowcut2[1,], xlim=xlim)
```

4.2 Werte zu einem Zeitpunkt aus der Trackdatei schneiden
Die Ausgabe ist **eine Matrix** oder **ein Vektor**

```
# F1-F4 zum zeitlichen Mittelpunkt fwerte ist eine Matrix:
fwerte = dcut(vowlax.fdat, 0.5, prop=T)
class(fwerte)
nrow(fwerte) == nrow(vowlax.fdat)

# F1 zum 0.75 Punkt (fwerte ist ein Vektor)
f1werte = dcut(vowlax.fdat[,1], 0.75, prop=T)
length(f1werte)==nrow(vowlax.fdat)

# F1 30 ms vor dem Offset
zeit = end(vowlax.fdat)-30
f1 = dcut(vowlax.fdat[,1], zeit)
```

5. Zwei-Dimensionale Abbildungen

```
# F1-F2 zum zeitlichen Mittelpunkt
form = dcut(vowlax.fdat, .5, prop=T)
```

5.1 Zwei Parameter 'Scatter-Plot'

```
plot(form[,1:2], pch=vowlax.l)
# oder
plot(form[,1], form[,2], type="n")
text(form[,1], form[,2], vowlax.l)

# nach Farben kodiert
code = as.numeric(factor(vowlax.l))
plot(form[,1], form[,2], pch=vowlax.l, col=code)

# Farbenauswahl
farben =c("brown", "grey", "blue", "orange")[code]
plot(form[,1], form[,2], pch=vowlax.l, col=farben)

# Zeichenauswahl
```

```
plot(form[,1:2], pch=code)
plot(form[,1:2], pch=code, col=code)
```

zB mit 'plotting characters' 15-18:

```
p = c(15:18)[code]
plot(form[,1:2], pch=p)

# mit Farbe
plot(form[,1:2], pch=p, col=farben)

# selbe Symbolen, unterschiedliche Farben
plot(form[,1:2], pch=21, col=code)
```

5.2 Mit Ellipsen

```
#
eplot(form[,1:2], vowlax.l, centroid=T)

# Mit Werten
eplot(form[,1:2], vowlax.l, centroid=T, dopoints=T)

# Achsen drehen (für eine F1 x F2 Abbildungen im Verhältnis zum Vokalraum)
eplot(form[,1:2], vowlax.l, centroid=T, dopoints=T, form=T)

# Standard-Abweichung der Ellipsen ändern
eplot(form[,1:2], vowlax.l, centroid=T, dopoints=T, form=T, nsdev=1)

# ohne Ellipsen
eplot(form[,1:2], vowlax.l, centroid=T, dopoints=T, form=T, doellipse=F)
```

5.3 Mehrere Parameter

```
pairs(form)
lab = paste("Formant", 1:4, sep="")
pairs(form, labels=lab)

pairs(form, labels=lab, col=code)
pairs(form, labels=lab, pch=15, col=code)
```

6. Zusammenfassung

```
# Nach Trackdatei prüfen
class(vowlax.fdat)
is.trackdata(vowlax.fdat)
summary(coutts.epg)

# Zeiten abfragen
end(vowlax.fdat)                        # Endzeit der Frames, eine Endzeit pro Segment
start(vowlax.fund[10,])                 # Startzeit der Frames, Segmente 1-10
tracktimes(vowlax.fdat[1:2,])           # Zeiten der Frames, Segmente 1-2

# Frames abfragen
vowlax.fdat[9:10,]$data                 # Die Frames von Segmenten 9 und 10

# Reihen, Spalten, Dimensionen
nrow(coutts.epg)                        # Anzahl der Segmente
ncol(vowlax.fund)                       # Anzahl der Tracks
dim(vowlax.fdat)                        # Anzahl der Segmente, Anzahl der Tracks

# Indizierung
vowlax.fund[1,]                         # Trackdatei vom ersten Segment
vowlax.fund[-c(1:10),]                  # Trackdatei aller Segmente außer den ersten 10
vowlax.fdat[,3]                         # Trackdatei vom dritten Parameter (F3)
temp = vowlax.l == "a"                  # Trackdatei, F1 und F2 aller "a" Vokale
vowlax.fdat[temp,1:2]

# Funktionen anwenden auf alle Frames
log(vowlax.fdat$data[,1], base=10)      # Logarithmus Basis 10 aller Frames

# Funktionen anwenden: getrennt pro Segment
trapply(vowlax.fund, mean, simplify=T)          # Ein f0-Mittelwert pro Segment
trapply(vowlax.fdat[,2], range, simplify=T)     # F2-Bereich pro Segment
trapply(vowlax.fdat[1:10,1], diff)              # Erste Ableitung von F1 pro Segment

# Abbildungen von Trackdateien
plot(vowlax.fund[10,])                  # f0 vom zehnten Segment
par(mfrow=c(2,2))                       # F1-F2 der ersten 4 Segmente
plot(vowlax.fdat[1:4,1:2], contig=F)
dplot(vowlax.fdat[,1], vowlax.l, 0.5)   # F1 überlagert alle Vokale synchronisiert
                                        # zum zeitlichen Mittelpunkt
dplot(vowlax.fdat[,1], vowlax.l, 0.5,   # wie oben aber zeit-normalisiert und
average=T, norm=T)                      # 'ensemble' gemittelt.

# Abbildungen von 2D Matrizen
plot(vowlax.fdat.5[,1:2])               # F1 x F2
eplot(vowlax.fdat.5[,1:2], vowlax.l, centroid=T)    # Wie oben mit Ellipsen pro Kategorie
```

Fragen

0. Erklären Sie die Beziehung zwischen einer Segmentliste, einer Trackdatei, einem Segment, und Frames.

Frage 1 bezieht sich auf die folgenden Objekte

- vowlax Segmentliste
- vowlax.l Etikettierungen (derived from label(vowlax)).
- vowlax.fdat Trackdatei der Formanten.
- vowlax.fund Trackdatei der Grundfrequenz.
- vowlax.rms Trackdatei dB-RMS (Intensität)

1. Erzeugen Sie Trackdateien (Fragen 1.1 - 1.8) oder Matrizen/Vektoren (Fragen 1.9 - 1.12) für die folgenden Fälle (siehe insbesondere die Beschreibung in 1.2 S. 5-6 und 4.2 S. 10)

Trackdateien:
1.1. F2 der ersten 10 Segmente
vowlax.fdat[1:10,2]

1.2 F1 und F3 vom letzten Segment

1.3 Grundfrequenz Segmente 10, 20, 28

1.4 Grundfrequenz aller "a" Vokale

1.5 F2 und F3 aller "I" und "O" Vokale

1.6 F2 der "a" und "E" Vokale Sprecherin "68"

1.7 F1-F4 zwischen für die erste Hälfte der Segmente (zwischen der Startzeit und zeitlichem Mittelpunkt)

1.8 Grundfrequenz für den mittleren 50% (zwischen dem 25% und 75% Zeitpunkten)

Vektoren/Matrizen
1.9 dB-RMS zum zeitlichen Mittepunkt

1.10 F1 und F2 der "i:" Vokale zum Segment-Onset

1.11 dB-RMS 15 ms vor dem Segment-Offset

1.12 F1 und F3 15 ms vor dem Segment-Offset für "a" und "I" Vokale, Sprecher "67"

2. (Siehe insbesondere die Beschreibung in 2. (S. 7-8), um diese Frage zu beantworten). Erzeugen Sie die folgenden Abbildungen:

2.1 Grundfrequenz als Funktion der Zeit vom 10en Segment (Abb. 2.1)

2.2 F1 und F2 zusammen als Funktion der Zeit der ersten 6 Segmente (in einer 2 x 3 Abbildung (Abb 2.2)

2.3 F2 von "I" und "a" als Funktion der Zeit, Sprecher "67" (Abb 2.3)

2.4 F2 von "E" als Funktion der Zeit getrennt für die beiden Sprecher (Abb. 2.4). Inwiefern kann die Abbildung durch Sprecherunterschiede erklärt werden? (Sprecher 67 ist männlich, Sprecherin 68 weiblich).

2.5 F2 von "a", Sprecher "67" getrennt für den davorkommenden Kontext "m" und "d" (Abb. 2.5). Welche phonetischen Faktoren erklären die Unterschiede vor allem zwischen proportionale Zeiten 0.0 - 0.4?

Abb 2.1

Abb. 2.2

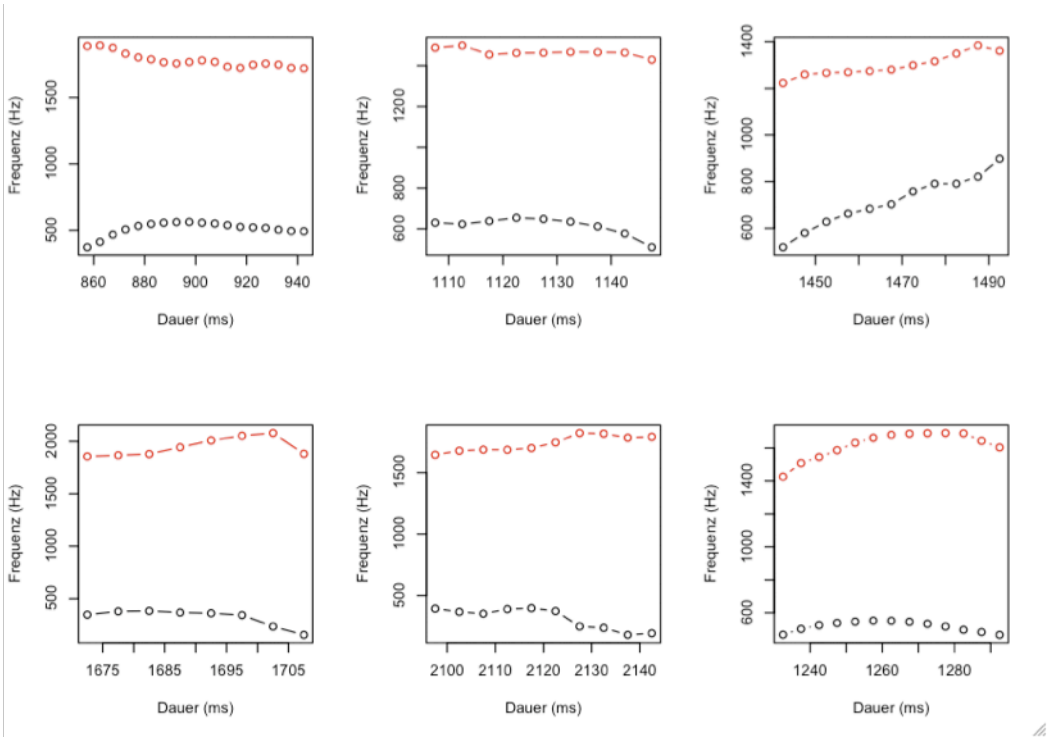


Abb 2.3

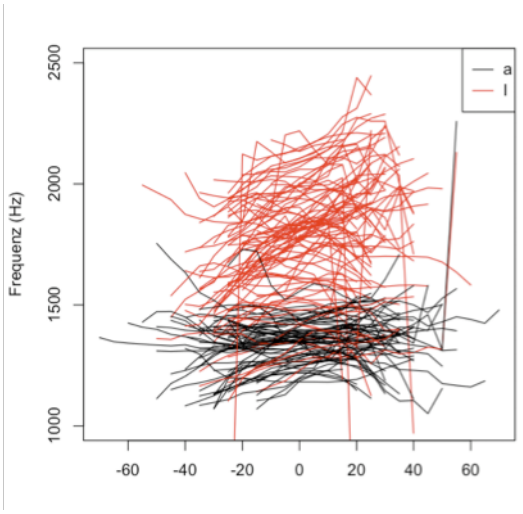


Abb 2.4

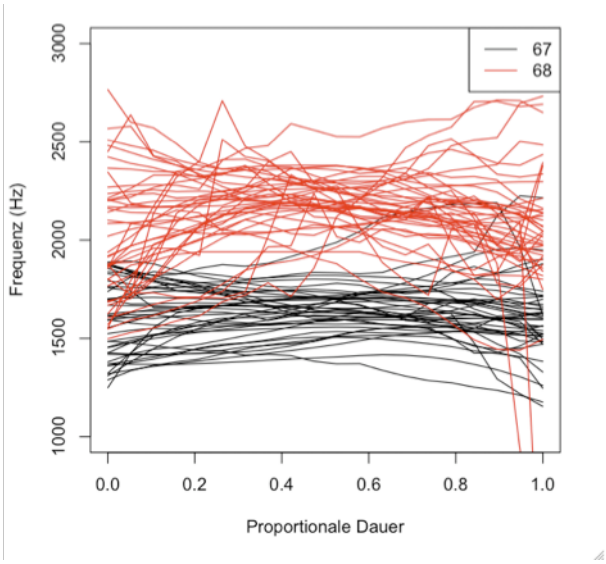
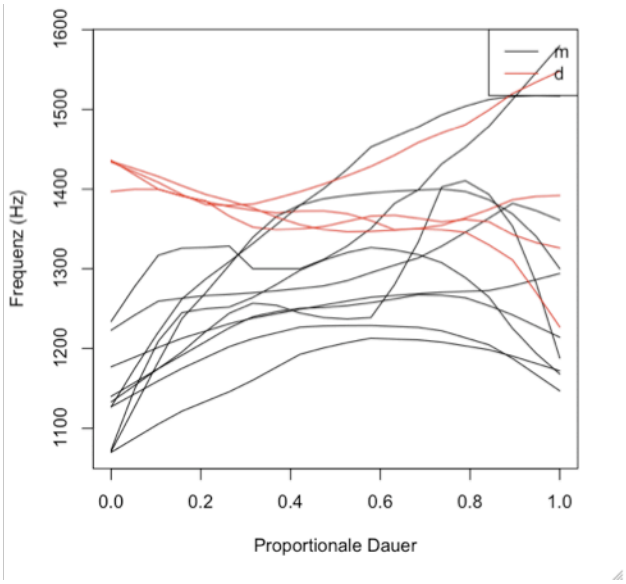


Abb. 2.5



3. (Für weitere Informationen zu Fragen 3 und 4 siehe insbesondere 4.2, 5.1, 5.2 S. 9-11)

Für Frage 3 benötigen Sie diese Objekte, die zueinander parallel sind:

vowlax	Segmentliste (4 Vokale)
vowlax.fdat	F1-F4 Trackdatei
vowlax.l	Vokal-Etikettierungen
vowlax.spkr	Sprecher-Etikettierungen

Erstellen Sie in einer Reihe und 2 Spalten die folgenden zwei Abbildungen (Abb 3.1):

(a) Die Dauer als Funktion der F1-Werte zum zeitlichen Mittelpunkt der "a" Vokalen von Sprecher 67. Würden Sie sagen, es liegt eine Beziehung zwischen diesen beiden Variablen vor? Was könnte die physiologische Erklärung für diese Beziehung sein?

(b) Ellipse Abbildungen aller Diphthonge für Sprecherin 68 für die Werte zum 9/10 der gesamten Dauer (Zeitpunkt 90%) entnommen. Welche phonetischen Faktoren könnten diese Unterschiede zwischen den Diphthongen in dieser Abbildungen verursachen?

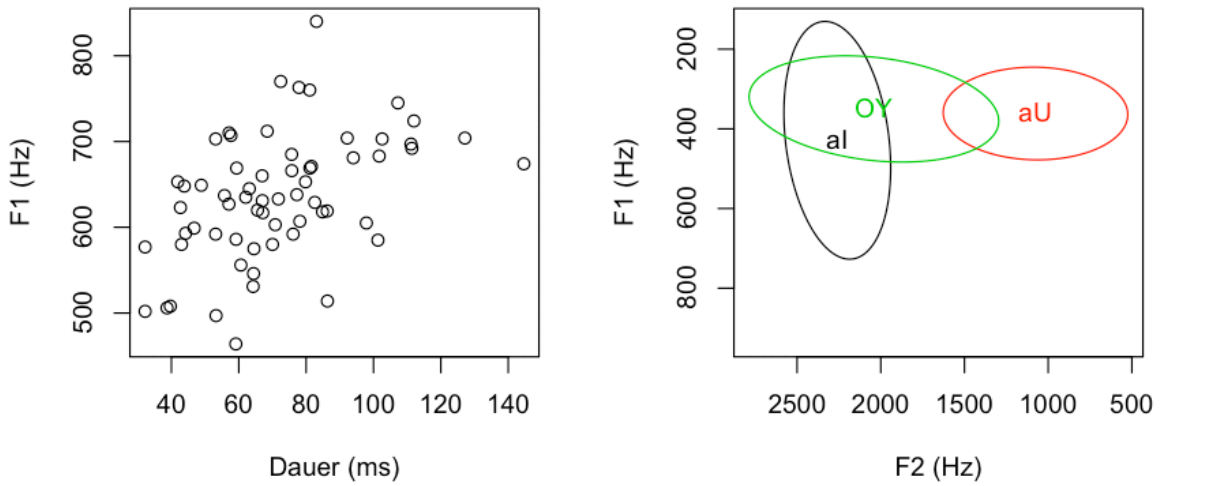


Abb. 3.1

4.

Für Frage 4 benötigen Sie zusätzlich zu den vowlax Objekten diese Objekte, die zueinander parallel sind:

dip.fdat	Trackdatei, F1-F4
dip.l	Diphthong-Etikettierungen
dip.spkr	Sprecher-Etikettierungen

Erstellen Sie zwei Matrizen wie folgt:

(a) eine zweispaltige Matrix der F1 und F2 Werte den "aI" und "aU" Diphthongen 1/3 der Gesamtdauer nach dem Diphthong-Onset entnommen für Sprecherin 68. Erstellen Sie einen Vektor der Diphthong-Etikettierungen, der zu dieser Matrix parallel ist.

(b) ebenfalls eine zweispaltige Matrix der F1 und F2 Werte dem zeitlichen Mittelpunkt von "a" Vokalen entnommen auch für Sprecherin 68. Erstellen Sie einen Vektor der Etikettierungen, der zu dieser Matrix parallel ist.

(c) Indem Sie die rbind() und c() Funktionen auf die Matrizen und Vektoren in (a) und (b) anbringen, erstellen Sie in einer Abbildung (Abb. 4) im F2 x F1 Raum die Werte und Ellipsen dieser "ai", "aU", und "a" Daten. Die Abbildung zeigt, dass "aI" und "aU" etwas nach links und rechts im Vergleich zu "a" in diesem Raum verteilt sind. Welche phonetischen Faktoren könnten diesen Unterschieden zugrunde liegen?

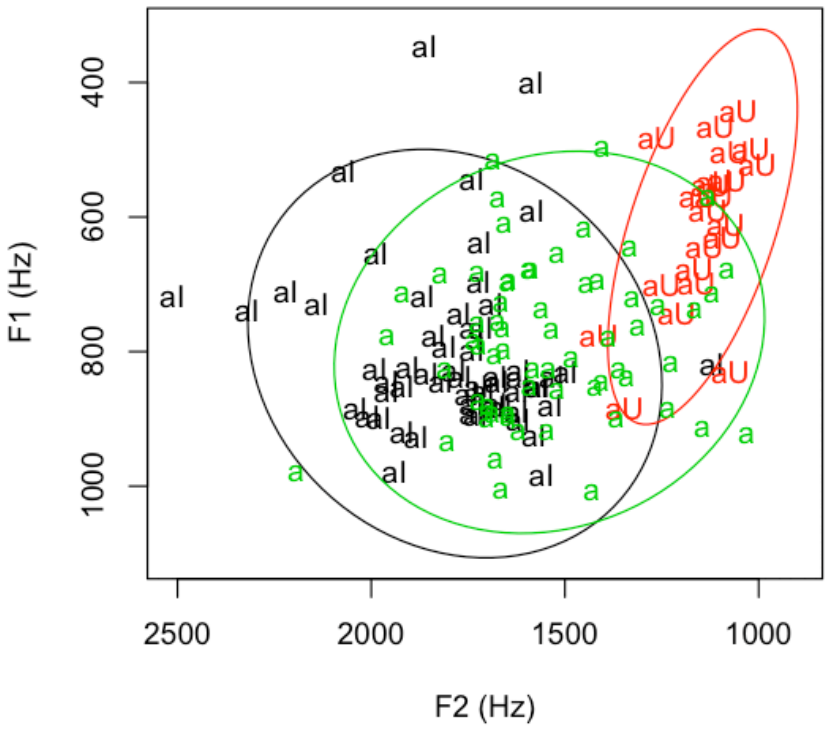


Abb. 4

5. (Diese Frage bezieht sich auf 3.2 (S. 8-9) und teilweise auf 4.1 und 4.2 (S. 9-10). Wenden Sie Funktionen an auf Trackdateien für die folgenden Fälle.

5.1 (Beispiel). F3 Mittelwert der ersten zwei Segmente in der Trackdatei vowlax.fdat.

Antwort:

```
trapply(vowlax.fdat[1:2,3], mean, simplify=T)
# Zwei Werte, einen Wert pro Segment
[1] 2329.556 2534.333
```

Zur Information: Diese Werte entsprechen den blauen Linien in Abb. 5:

```
m = trapply(vowlax.fdat[1:2,3], mean, simplify=T)
par(mfrow=c(1,2))
plot(vowlax.fdat[1,3], type="l", ylab="Frequenz (F3)", xlab="Dauer (ms)",main="Segment 1")
abline(h=m[1], col="blue")
```

```
plot(vowlax.fdat[2,3], type="l", ylab="", xlab="Dauer (ms)", main="Segmente 2")
abline(h=m[2], col="blue")
```

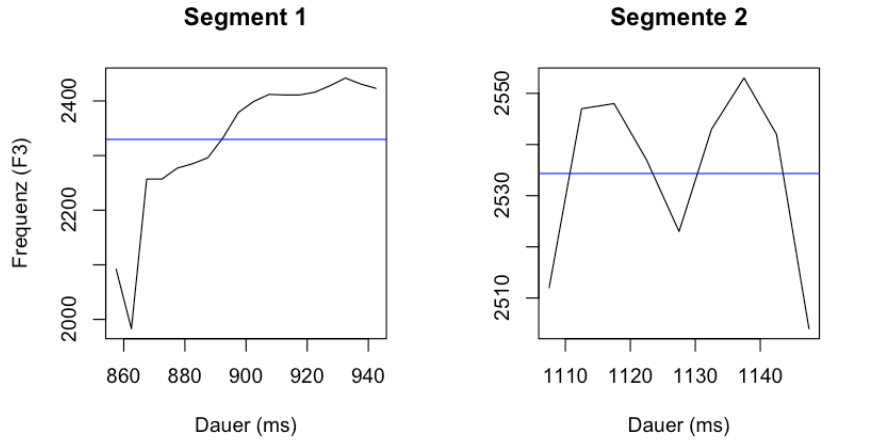


Abb. 5. Die blaue Linien ist der Mittelwert über alle Frames.

N.B. *jeweils einen Wert pro Segment...*
5.2 Die Standardabweichung von F2 für die ersten 10 Segmente der Diphthonge in der Trackdatei dip.fdat (*...daher 20 Standardabweichungen, eine pro Segment*).

5.3 Das f0-Maximum in Segmenten 14, 19, 23 der Diphthonge in der Trackdatei vowlax.fund

5.4 Der F1-Median in dem zeitlichen ersten Drittel der Diphthonge (einen Wert pro Diphthong)

5.5 Der F2-Bereich für ein Intervall ± 10 ms jenseits des zeitlichen Mittelpunkts vom Vokal in der Trackdatei vowlax.fdat

5.6 Das dB-RMS Maximum für "a" Vokale von Sprecherin 68 in der Trackdatei vowlax.rms (NB vowlax.l und vowlax.spkr sind die Etikettierungen, die zu der Trackdatei parallel sind).

5.7 Erstellen Sie eine neue Trackdatei F2 - F1 (F2 minus F1) aus der Trackdatei dip.fdat und berechnen Sie den Mittelwert davon (einen Wert pro Segment) für den Diphthong "aU" (dip.l ist der Vektor der Etikettierungen, der dazu parallel ist).

5.8 Diese Funktion berechnet den Zeitpunkt, zum dem ein Maximum oder Minimum in der Frames von einem Segment vorkommt.

```
mfun <- function(daten, fun=max)
{
zeiten = tracktimes(daten)
temp = daten == fun(daten)
zeiten[temp]
}
```

z.B. Hier sind die F1-Frames für den 10en Segment in vowlax.fdat:

vowlax.fdat[10,]$data

```
        T1   T2   T3   T4
1837.5 697 1369 2389    0
1842.5 698 1363 2397    0
1847.5 686 1359 2391 3457
1852.5 650 1349 2398 3628
1857.5 649 1352 2408 3700
1862.5 641 1354 2415 3892
1867.5 653 1361 2429 3997
1872.5 645 1371 2436 4030
1877.5 640 1408 2422 3756
1882.5 355 1330 2426 3602
```

Der Zeitpunkt, zudem F2 ein Maximum hat, kann dann wie folgt berechnet werden:
mfun(vowlax.fdat[10,2]$data)
[1] 1877.5

(a) Berechnen Sie die Zeitpunkte, zu denen F2 ein Maximum hat in allen Segmenten von vowlax.fdat (einen Zeitpunkt pro Segment).

(b) Erstellen Sie eine Matrix der F1 und F2 Werte (410 Reihen, eine Reihe pro Segment und zwei Spalten), die zu diesen Zeitpunkten vorkommen (siehe 4.2, S. 10).

3.9 Hier ist eine Funktion um die ersten 3 Werte von einem Vektor zu mitteln:

```
dreifun <- function(x, k=3)
{
mean(mean(x[1:k]))
}

# Zehn randomisierte Werte
x = runif(10)
# Mittelwert der ersten drei
dreifun(x)
# das gleiche
mean(x[1:3])
```

Berechnen Sie den Mittelwert der ersten 3 Frames von F2 für die ersten 20 Segmente von vowlax.fdat (also einen Wert, den Mittelwert der ersten 3 Frames, pro Segment).