

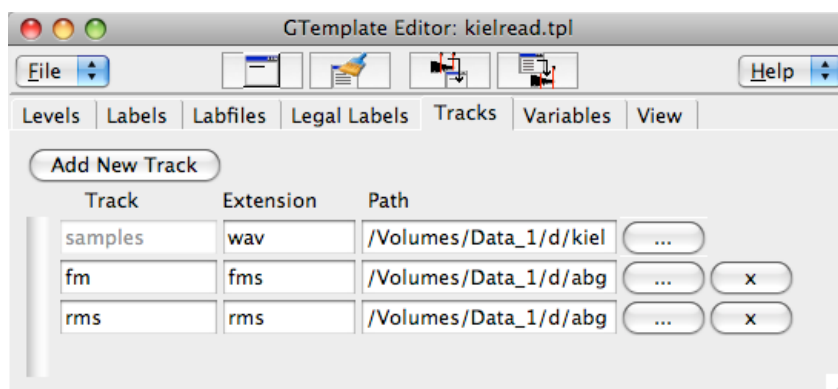
## Manipulation und Abbildung von Trackdateien

### 0. Allgemein

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

#### 0.1 Welche Signale (Tracks) können als Trackdateien in R erstellt werden?

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:

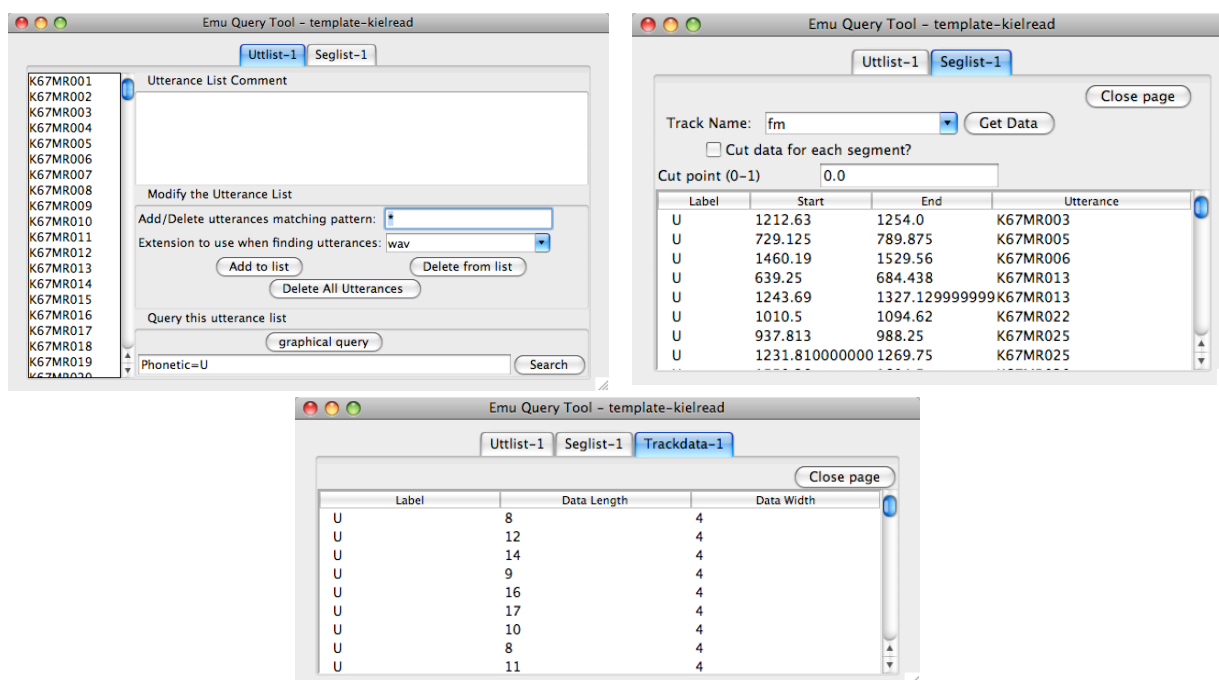


wären die Tracks `samples`, `fm`, `rms` (Zeitsignale, Formanten, Intensität) vorhanden.

#### 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 H:/form.txt`. (H: ist irgendein von Ihnen gewähltes Verzeichnis). 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 als Parameter definiert wird.

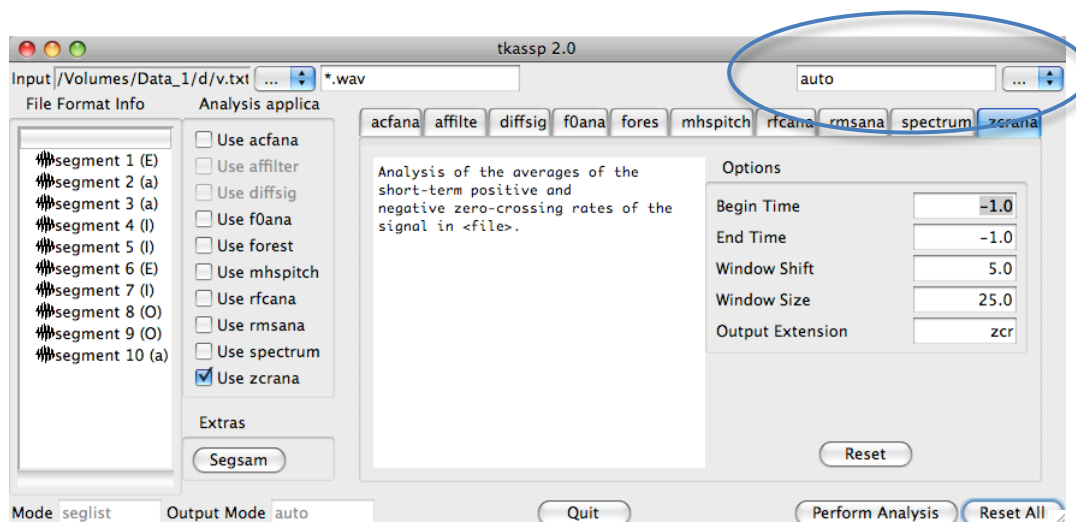
2. Die Segmentliste direkt in EMU-tkassp einlesen.

**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<sup>1</sup>:

```
class(vowlax.fdat)
class(vowlax.fund)
class(vowlax.zcr)
[1] "trackdata"
```

```
is.trackdata(vowlax.fdat)
```

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
```

---

<sup>1</sup> angenommen `library(emu)` ist schon gestartet worden

## 1. Trackdatei, Frames, Tracktimes

Eine Trackdatei enthält für mehrere Segmente die Signalwerte und die Zeiten, zu denen diese Werte vorkommen.

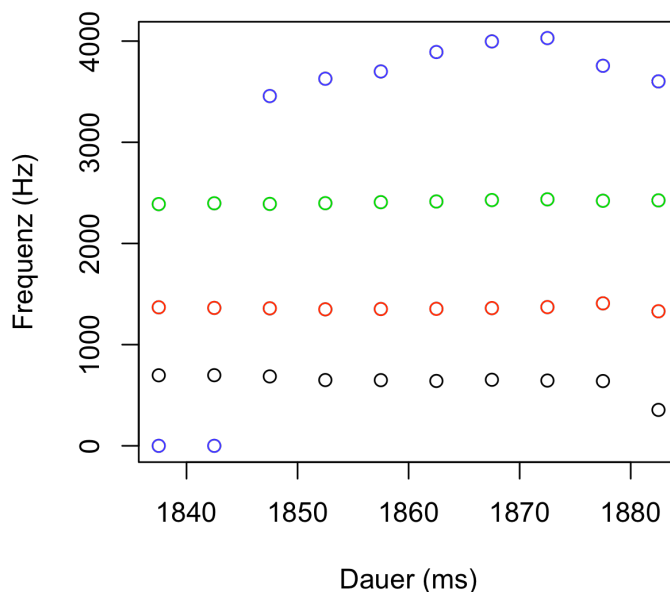
Die Signalwerte können für beliebige Segmente mit `frames()` abgefragt werden

Die Zeiten, zu denen diese Signalwerte vorkommen, mit `tracktimes()`

z.B.

# Abbildung F1-F4 vom zehnten Segment

```
plot(vowlax.fdat[10,], ylab="Frequenz (Hz)", xlab="Dauer (ms)")
```



# Die Werte

```
frames(vowlax.fdat[10,])
```

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

# Deren Zeiten

```
tracktimes(vowlax.fdat[10,])
```

1837.5 1842.5 1847.5 1852.5 1857.5 1862.5 1867.5 1872.5 1877.5 1882.5 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

Eine Trackdatei lässt sich in vielen Hinsichten wie eine Matrix behandeln<sup>2</sup>.

*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?
```

```
# Die F1 und F2 Frames der Segmente 10, 12, und 18?
```

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

vowlax	Segmentliste
vowlax.fdat	Trackdatei, F1-F4
vowlax.fund	Grundfrequenz

---

<sup>2</sup> Wegen einer object-oriented Implementierung und genauer wegen der Funktion [[.trackdata\(\)](#)]. Geben Sie ``[.trackdata`` ein, wenn Sie diese Funktion sehen möchten.

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/ws0910/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`) funktionieren 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 Vokalonsset
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 einer Trackdatei anwenden

#### 3.1 Arithmetische Berechnungen

werden direkt auf die Frames einer Trackdatei angewandt, ohne die Frames explizit angeben zu müssen.

```

# F2/F1 von vowlax.fdat
v = vowlax.fdat[,2]/vowlax.fdat[,1]
dplot(v[1:10,], ylab = "F2 / F1")

```

```

# F1 - f0 (F1 minus f0) der ersten 10 Segmente
d = vowlax.fdat[1:10,1] - vowlax.fund[1:10]

```

#### 3.2 Funktion anwenden: Die Segmentgrenzen spielen keine Rolle

Alle Funktionen in Math, Math2, Summary, Complex in [help\(Ops\)](#) werden direkt auf die Frames angewandt.

```

# Logarithmus von F1
flogeins = log (vowlax.fdat[,1])

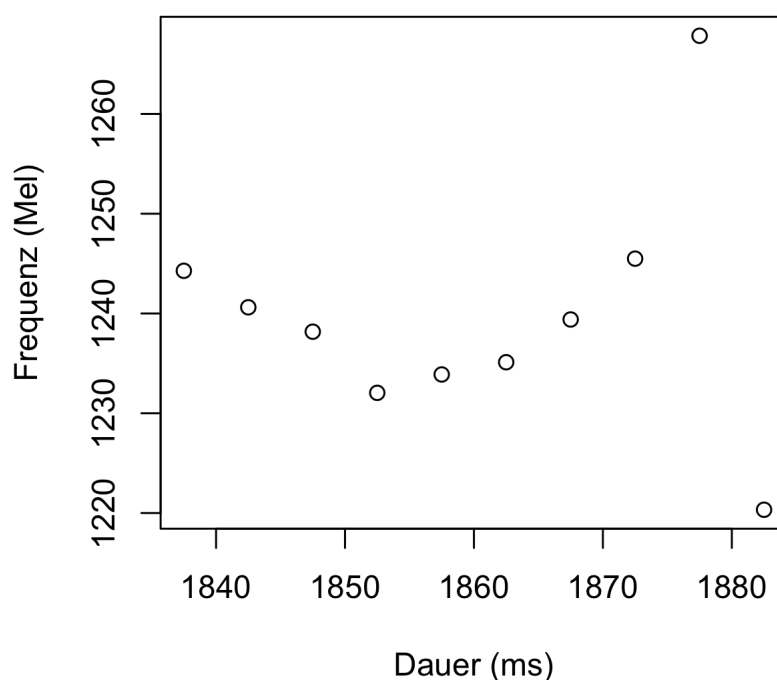
```

```
# Wurzel von F2
w = sqrt(vowlax.fdat[,2])

# Hier ist die Formel, um Hertz in Mel zu konvertieren.
1/log(2) * (log(1 + (hz/1000))) * 1000

# zB um 800 Hz in Mel zu konvertieren
hz = 800
1/log(2) * (log(1 + (hz/1000))) * 1000
```

Ich will F2 vom 10<sup>en</sup> Segment von `vowlax.fdat` in Mel konvertieren. Und dann die Abbildung davon wie unten machen. Wie geht das?



Es gibt übrigens auch eine Emu-R Funktion um die Hz-Mel Konvertierung durchzuführen:

```
m = mel(vowlax.fdat)
plot(m[10,2], ylab="Frequenz (Mel)", xlab = "Dauer (ms)")

# Und Hz-Bark:
b = bark(vowlax.fdat)
plot(b[10,2], ylab="Frequenz (Mel)", xlab = "Dauer (ms)")
```

### 3.2 Wenn man zwischen Segmenten trennen muss

Andere Funktionen, die nicht `in help(Ops)` gelistet sind, haben meistens nur eine sinnvolle Interpretation, **wenn sie auf Segmente getrennt angewandt werden**. z.B. ich will den Grundfrequenzmittelwert der ersten 5 Segmente wissen (einen Mittelwert pro Segment, 5 Werte). Wie geht das?



```
mean(vowlax.fund)
```

geht nicht, weil `mean()` nicht in `help(Ops)` gelistet ist.

```
mean(frames(vowlax.fund[1:5,]))
```

ginge, aber das ist der Mittelwert **aller Frames der ersten 5 Segmente** (einen Wert - und nicht was ich will).

Daher:

```
mean(frames(vowlax.fund[1,]))
```

```
mean(frames(vowlax.fund[2,]))
```

```
mean(frames(vowlax.fund[3,]))
```

```
mean(frames(vowlax.fund[4,]))
```

```
mean(frames(vowlax.fund[5,]))
```

Schneller ginge das mit einer sogenannten For-Schleife:

```
ergebnis = NULL
```

```
for(j in 1:5){
```

```
  ergebnis = c(ergebnis, mean(frames(vowlax.fund[j,])))
```

```
}
```

```
ergebnis
```

Noch viel einfacher geht das mit der `trapply()` Funktion, die eine Funktion auf eine Trackdatei anwendet:

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

`fun()` kann eine beliebige Funktion in `trapply(Trackdatei, fun)` sein (auch eine selbst erstellte), **solange sie sich auf die Frames von irgendeinem Segment anwenden lässt**. z.B.

funktioniert `sd()` innerhalb von `trapply()` ? Zuerst prüfen, ob sie sich auf die Frames von irgendeinem Segment anwenden lässt.

```
sd(frames(vowlax.fund[10,]))
```

Ja. Daher die Standardabweichung von `f0` der ersten 5 Segmente:

```
trapply(vowlax.fund[1:5,], sd, simplify=T)
```

```
# F1-Standardabweichung, Segmente 20:30
```

```
# F2-Minimum alle Segmente
```

```
# F1-Bereich Segmente 4, 8, 12 (Die Funktion range() verwenden)
```

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(frames(vowlax.rms[1,]))
```

ist `diff()` 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.rms[1:5,], diff)
```

Das Argument `returntrack=T` baut gleichzeitig daraus eine neue Trackdatei:

```
dneu = trapply(vowlax.rms[1:5,], diff, returntrack=T)
```

```
par(mfrow=c(1,2))
plot(vowlax.rms[1,], type="b", main="dB-RMS", ylab="dB")
plot(dneu[1,], type="b", main = "Erste Ableitung von dB-RMS")
```

## 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)
```

```
# Farbauswahl
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], vowelax.l, centroid=T, dopoints=T, form=T, nsdev=1)
```

```
# ohne Ellipsen
eplot(form[,1:2], vowelax.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(vowelax.fdat)
is.trackdata(vowelax.fdat)
summary(coutts.epg)
```

### # Zeiten abfragen

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

### # Frames abfragen

```
frames(vowelax.fdat[9:10,]) # Die Frames von Segmenten 9 und 10
```

### # Reihen, Spalten, Dimensionen

```
nrow(coutts.epg)           # Anzahl der Segmente
ncol(vowelax.fund)         # Anzahl der Tracks
dim(vowelax.fdat)          # Anzahl der Segmente, Anzahl der Tracks
```

### # Indizierung

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

### # Funktionen anwenden auf alle Frames

```
vowelax.fdat[,1]^2         # Trackdatei, F12
vowelax.fdat[,1] + 2 * vowelax.fdat[,3] # Trackdatei, F1 + (2 x F3)
log(vowelax.fdat[,2])      # Trackdatei, Log. von F2
```

### # Funktionen anwenden: getrennt pro Segment

```
trapply(vowlax.fund, mean, simplify=T)
trapply(vowlax.fdat[,2], range, simplify=T)
trapply(vowlax.fdat[1:10,1], diff)
```

### # Abbildungen von Trackdateien

```
plot(vowlax.fund[10,])
par(mfrow=c(2,2))
plot(vowlax.fdat[1:4,1:2], contig=F)
dplot(vowlax.fdat[,1], vowlax.l, 0.5)
```

```
dplot(vowlax.fdat[,1], vowlax.l, 0.5,
average=T, norm=T)
```

### # Abbildungen von 2D Matrizen

```
plot(vowlax.fdat.5[,1:2])
eplot(vowlax.fdat.5[,1:2], vowlax.l, centroid=T)
```

```
# Ein f0-Mittelwert pro Segment
# F2-Bereich pro Segment
# Erste Ableitung von F1 pro Segment
```

```
# f0 vom zehnten Segment
# F1-F2 der ersten 4 Segmente
```

```
# F1 überlagert alle Vokale synchronisiert
# zum zeitlichen Mittelpunkt
# wie oben aber zeit-normalisiert und
# 'ensemble' gemittelt.
```

```
# F1 x F2
# 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 Mittelpunkt

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?

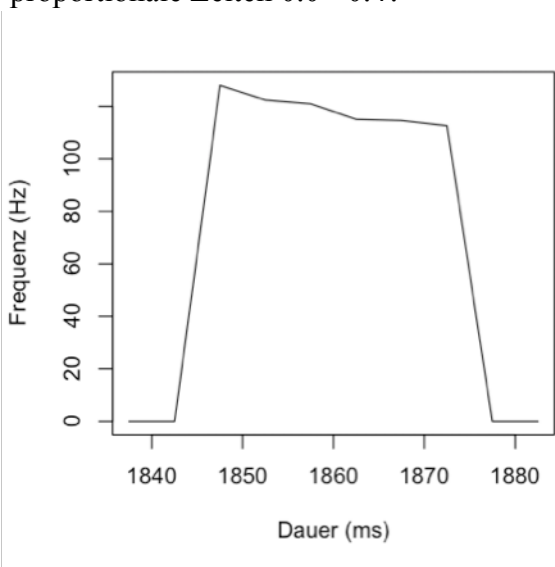


Abb 2.1

Abb. 2.2

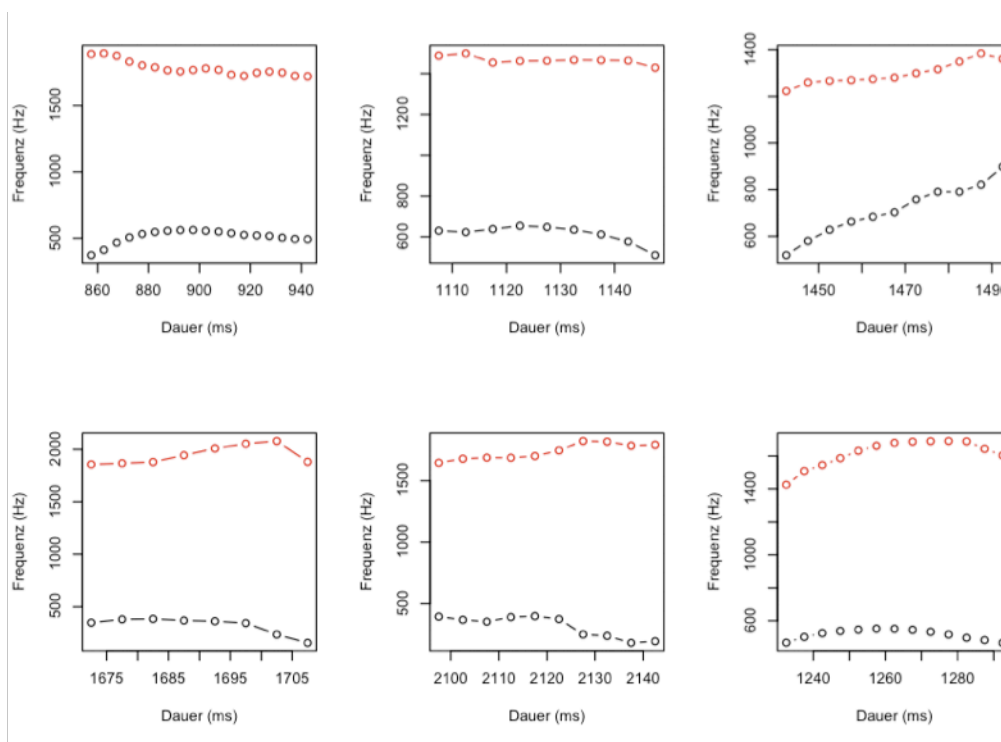


Abb 2.3

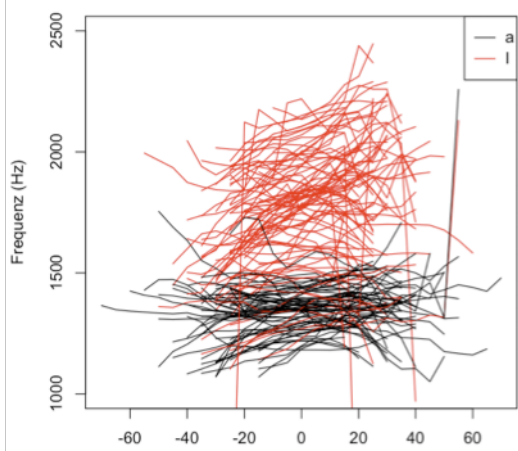


Abb 2.4

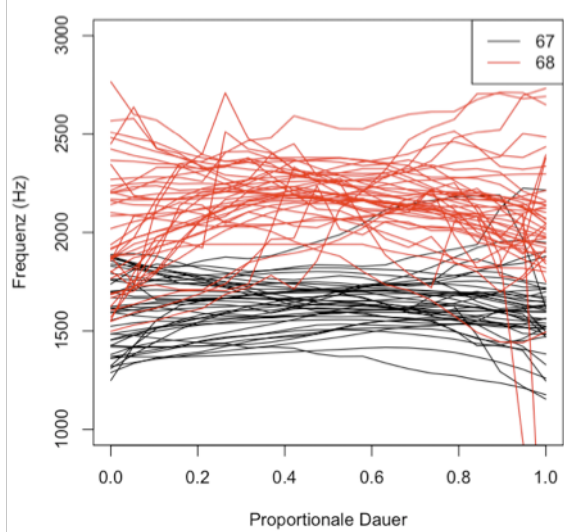
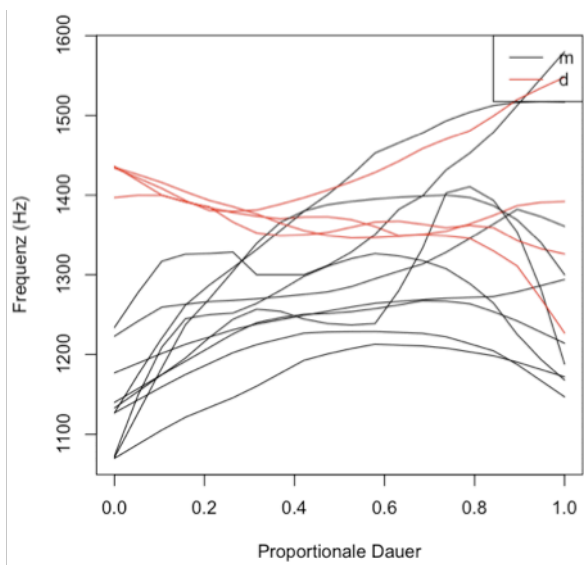


Abb. 2.5





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?

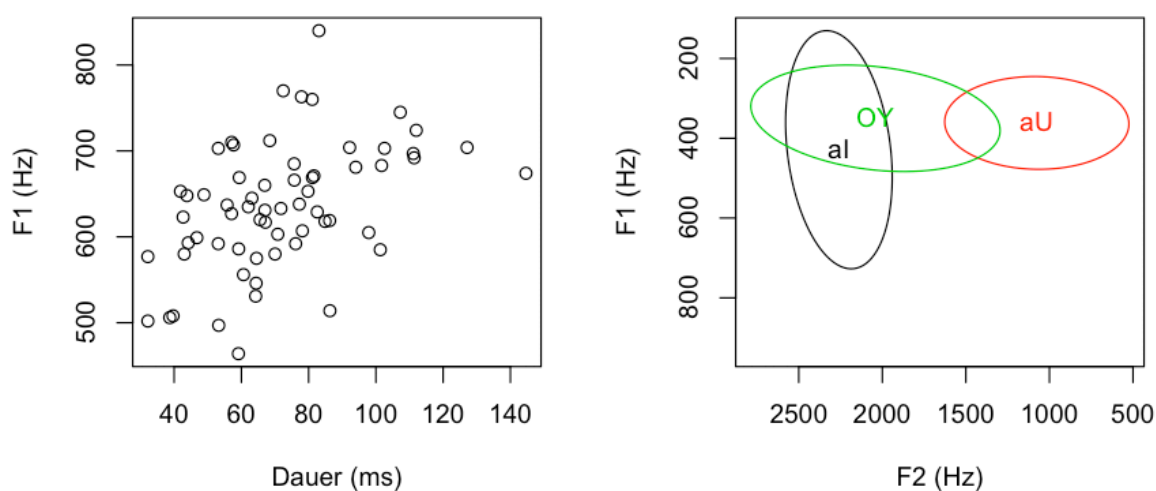


Abb. 3.1

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?

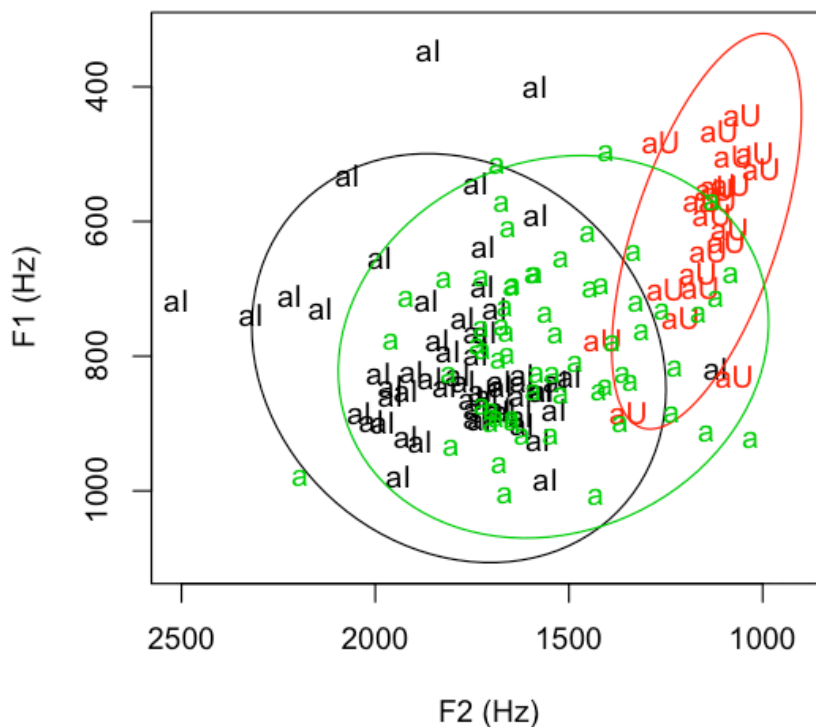


Abb. 4

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")
```

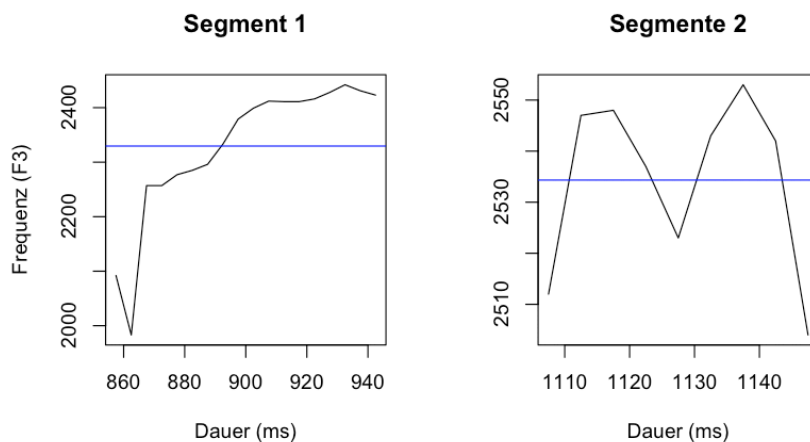


Abb. 5. Die blaue Linie ist der Mittelwert über alle Frames.

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 10 Standardabweichungen, eine pro Segment).

5.3 Das  $f_0$ -Maximum in Segmenten 14, 19, 23 der Vokale 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  $\pm 10$  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  $F_2 - F_1$  ( $F_2$  minus  $F_1$ ) 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][1]
}
```

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

```
frames(vowlax.fdat[10,])
```

	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, zu dem F2 ein Maximum hat, kann dann wie folgt berechnet werden:

```
mfun(frames(vowlax.fdat[10,2]))
```

```
[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).

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).