# Berechnung von spektralen Parametern

**0. Vorverarbeitung**

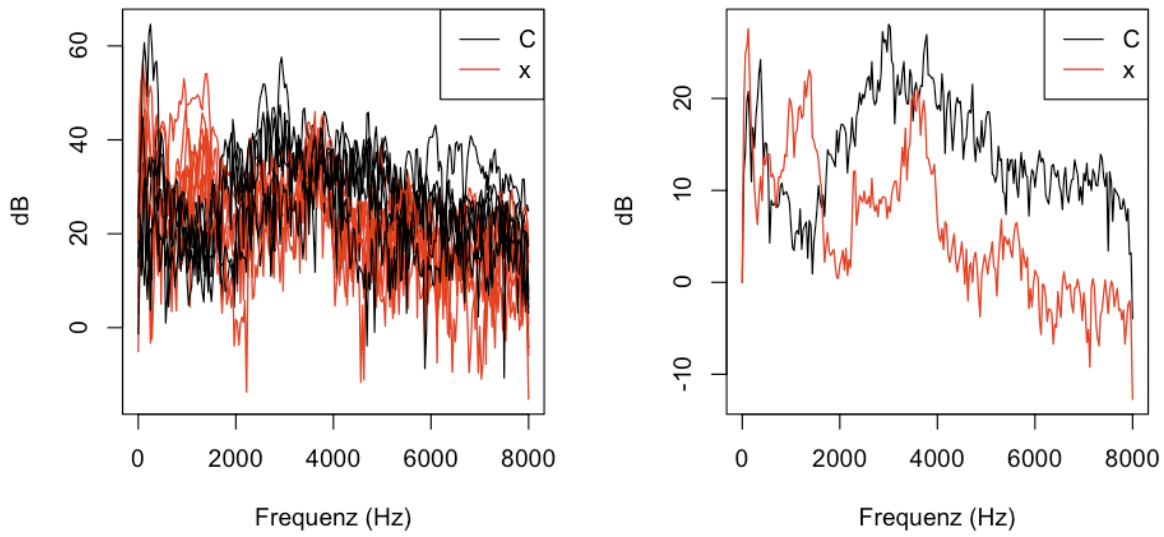
# Segmentliste aller [C, x] Frikative (Ebene Phonetik) der `timetable`-Datenbank
dor =

# Label-Vektor
dor.l =

# Spektra für die Segmentliste
dor.dft =

# Spektra zum zeitlichen Mittelpunkt
dor.dft5 =

# Abbildung, Spektra zum zeitlichen Mittelpunkt als Funktion der Kategorie (unten links=
# Mittelwerte (Ensemble-Averaged, oben rechts)



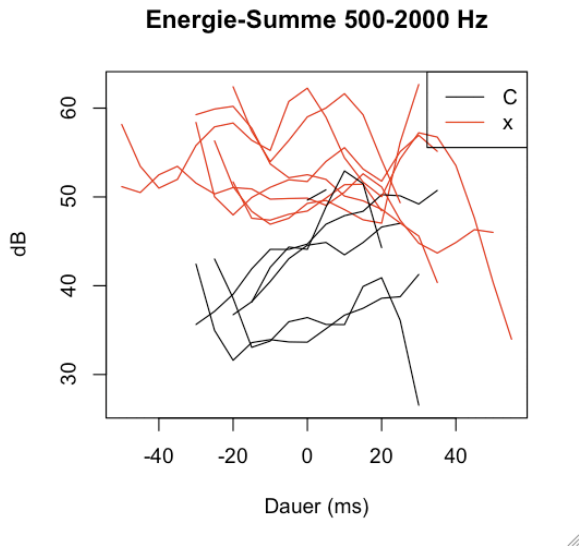
**1. Energie-Berechnungen**

Summe: sum()
Mittelwert: mean()
Varianz: var()
Median: median()
usw.

Meistens soll power=T eingesetzt werden, damit die Berechnungen nicht in der logarithmischen, sondern linearen Skala durchgeführt werden

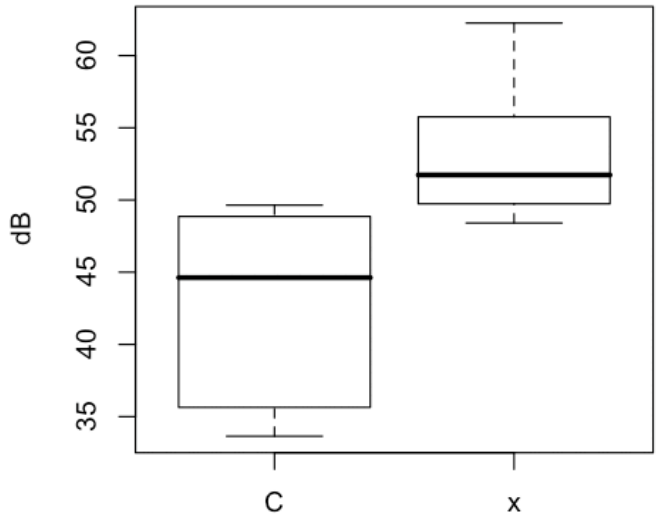
Syntax: `fapply(spec, Funktion)`
`spec` ist ein Spektral-Objekt (mit `is.spectral()` ) prüfen, `Funktion` ist die Funktion, die man anwenden möchte

```
# prüfen, ob wir ein Spektral-Objekt haben
is.spectral(dor.dft)
TRUE
is.spectral(dor.dft5)
TRUE

# zB Energie summieren im Bereich 500 – 2000 Hz. Hier wird das für die alle Spektra
# der Trackdatei durchgeführt.
summe = fapply(dor.dft[,500:2000], sum, power=T)

# Als Funktion der Zeit zum zeitlichen Mittelpunkt abbilden
dplot(summe, dor.l, main="Energie-Summe 500-2000 Hz", offset=.5)
```




```
# A. Energie-Summe 500-2000 Hz zum zeitlichen Mittelpunkt der Frikative?
summe5 =
```

# Boxplot-Abbildung




# **fapply()** kann auf spektrale Matrizen angewandt werden. Vorher hatten wir
# mit **dcut()** eine spektrale Matrix zum zeitlichen Mittelpunkt erzeugt.
# B. Energie-Summe 500 – 2000 Hz

```
werte  = fapply(dor.dft5[,500:2000], sum, power=T)
```

# A. und B. müssten genau das gleiche sein

```
summe5 -  werte
```

**2. Differenz-Spektra**

[ba, da] Silben.
(a) Spektrum berechnen im Verschluss
(b) Spektrum berechnen kurz nach der Lösung
*a* – *b* berechnen (die Spektra voneinander subtrahieren).

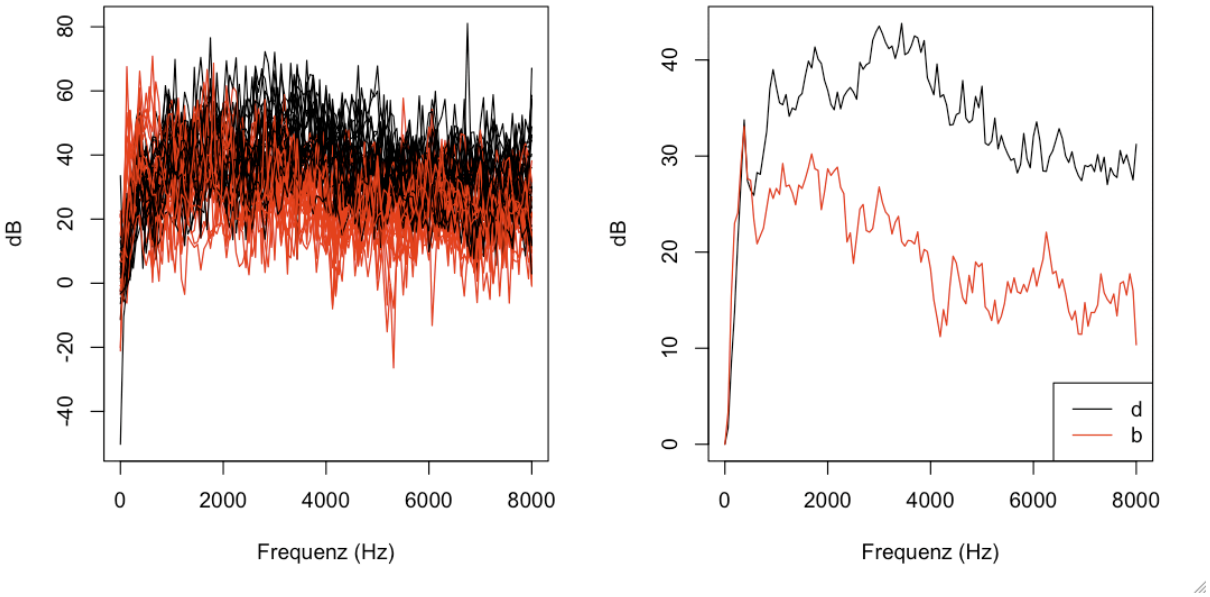
Welcher Laut müsste den größeren Differenz aufweisen ([b] oder [d]) und warum?

Daten

| | |
|---|---|
| **plos**: | Segmentliste, "b" und "d" in KV Silben, Verschluss + Lösung. |
| **plos.sam** | Trackdatei (audio-Signal) |
| **plos.dft** | Trackdatei (Spektra) |
| **plos.asp** | Vektor (numerisch) vom Zeitpunkt, zu dem die Lösung vorkommt |

```
# zB Abbildung der wav-Datei, Segment 1
plot(

# Zeitpunkt der Lösung überlagern
abline(v=plos.asp[1])

# (a) Spektra (Spektrale-Matrix)  20 ms vor der Lösung
davor  = dcut(

# (b) Spectra 10 ms nach der Lösung
danach  = dcut(

# Differenz-Spektra: (b) - (a)
d =
```

# Abbildung der Differenz-Spektra nach Kategorie kodiert (unten links)




# Wie oben, aber ensemble-averaged (Mittelwert pro Kategorie)

# Einen Frequenz-Bereich auswählen, in dem [b, d] unterschiedlich sind, und
# die Energie in diesem Bereich summieren.

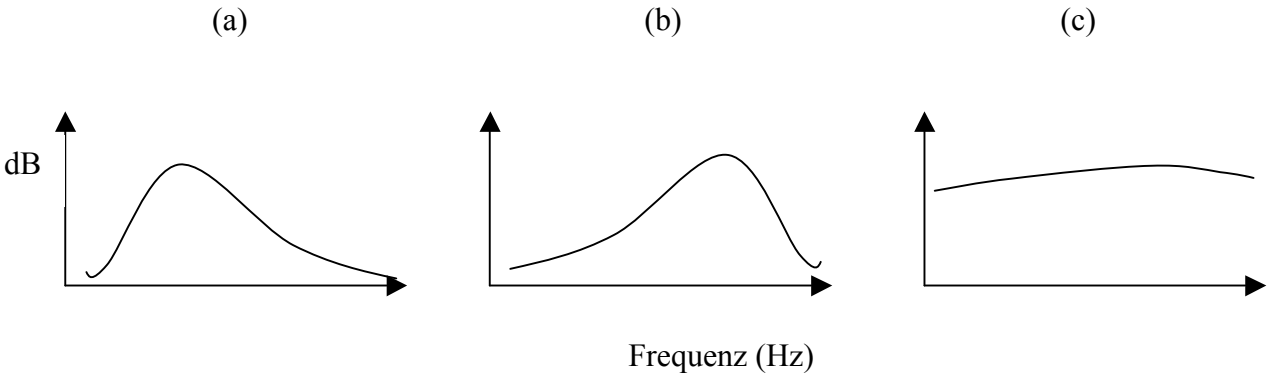
```
summe =
```

# Boxplot-Abbildung davon

### 3. Spektrale Momente

Ein ganzes Spektrum wird auf 4 Werte reduziert.
**$m_1$**: Gewichtschwerpunkt vom Spektrum
**$m_2$**: Spektrale Varianz

(Es gibt auch **$m_3$**, **$m_4$** womit wir uns jetzt nicht befassen werden).




(a) $m_1$ niedrig, $m_2$ niedrig
(b) $m_2$ hoch, $m_2$ niedrig
(c) $m_2$ hoch

Funktion: **moments()**

Anwendung auf einem Spektrum. (Die Einheit von $m_1$ ist Hz, von $m_2$ $Hz^2$).

```
m = moments(dor.dft5[1,], minval=T)
m
3.886044e+03  4.587377e+06  1.050328e-01 -9.651707e-01
```

# wie oben, zwei Komma-Stellen
```
round(m, 2)
3886.04     4587377.19       0.11      -0.97
```

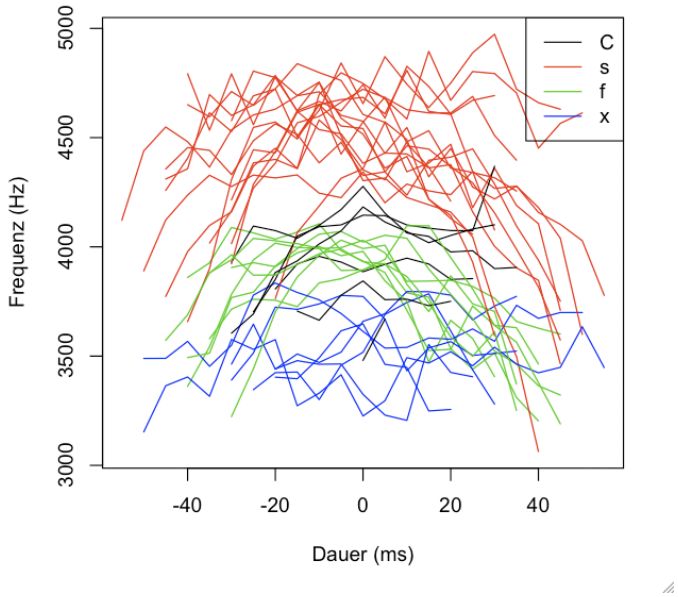
# Anwendung auf alle Spektra der spektralen Matrix (res ist eine 4-spaltige Matrix).
```
res = fapply(dor.dft5, moments, minval=T)
```

# Anwendung auf alle Spektra der spektralen Trackdatei (res ist eine 4-spaltige Trackdatei).
```
res = fapply(dor.dft, moments, minval=T)
```

**Frage.**
Welcher Frikativ von [f, s, C, x] müsste den höchsten $m_1$ haben? den niedrigsten $m_1$? den höchsten $m_2$? Warum?

# Segmentliste aus der timetable-Datenbank, "s", "f", "C", "x"
```
seg = emu.query("timetable", "*", "Phonetic=f | s | C | x")
```
# label-Vektor
```
seg.l = label(seg)
```
# Spektrale-Trackdatei
```
seg.dft = emu.track(seg, "dft")
```
# Wie berechne ich 'moments' für die gesamte Trackdatei?
```
m =
```

# Ich will eine Abbildung von $m_1$ als Funktion der Zeit haben, synchronisiert zum zeitlichen
# Mittelpunkt, und getrennt pro Kategorie abbilden
```
dplot(
```




# Das gleich für $m_2$
```
dplot(
```

# Den Ensemble-Average für $m_2$ abbilden

# Ich will eine Ellipse-Abbildung machen, $m_1$ x $m_2$ zum zeitlichen Mittelpunkt, getrennt pro
# Frikativ-Kategorie (um festzustellen, inwiefern $m_1$ x $m_2$ zwischen den Kategorien trennen).

```
m5 = dcut(
eplot(
```

## 4. DCT-Koeffiziente und DCT-Glättung von Spektra

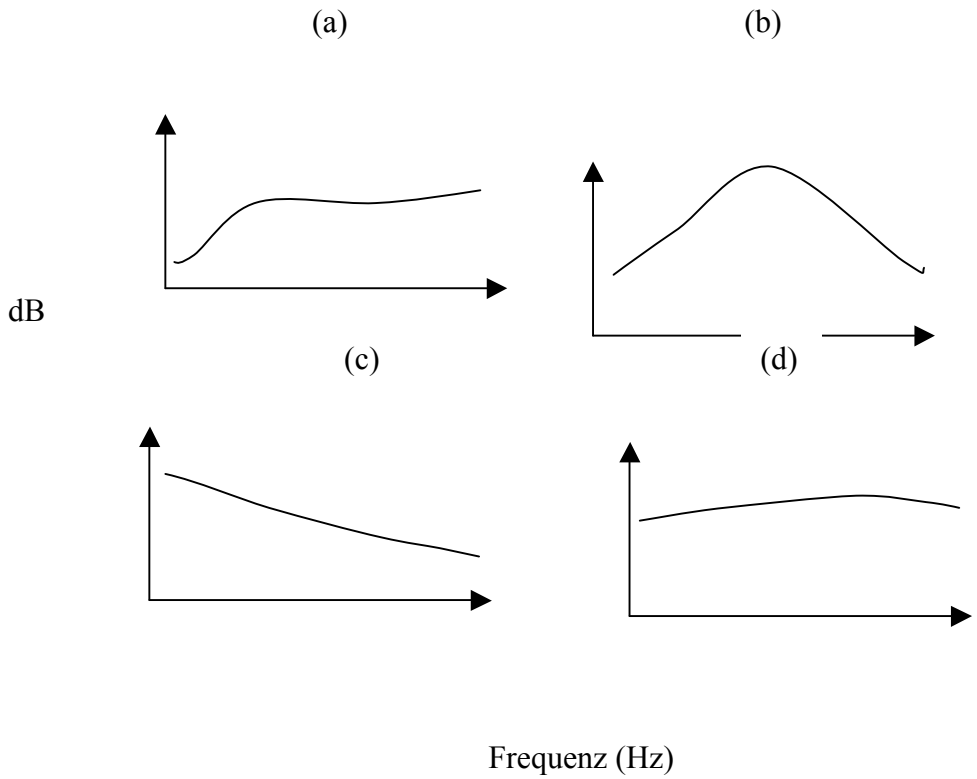
```
seg = emu.query("timetable", "*", "Phonetic=f | s | C | x")
# label-Vektor
seg.l = label(seg)
# Spektrale-Trackdatei
seg.dft = emu.track(seg, "dft")

# Spektra zum zeitlichen Mittelpunkt
d5 = dcut(seg.dft, .5, prop=T)
```

### 4.1. DCT-Koeffiziente berechnen (die kepstral Koeffiziente). zB die ersten 5

DCT-Koeffiziente von einem Spektrum sind **Kepstra**. Man kann DCT(Kepstral) Koeffiziente 0, 1, 2, ..N-1 berechnen, in der N-1 die Anzahl der spektralen Komponente bis zur Faltung ist (zB. 129 Komponente für eine 256 Punkt DFT).
Die ersten paar Koeffiziente verschlüsseln wichtige Eigenschaften der Gestaltung vom Spektrum (0: Mittelwert, 1: Neigung, 2: Krümmung).

```
k = fapply(d5[,500:7500], dct, 5)
```

- Kepstral Coeffizient 1 (K1, Spalte 2) ist im Verhältnis zur spektralen Neigung
- Kepstral Coeffizient 2 (K2, Spalte 3) ist im Verhältnis zur spektralen Krümmung




K1 unterscheidet (a, b, c)
K2 unterscheidet (b) von (a, c, d)

Inwiefern können diese 4 Frikative durch K1 und K2 getrennt werden?

```
eplot(k[,2:3], seg.l, dopoints=T)
```

### 4.2. DCT-Glättung

Wie wir in der Analyse von Vokalen gesehen haben, sind DCT-Koeffiziente nichts mehr als die Amplituden von Sinusoiden mit Frequenzen 0, ½ 1, 1.5, 2...

Wenn diese Sinsuoiden mit den niedrigsten Frequenzen summiert werden, dann bekommt man ein geglättetes Signal (daher zB geglättete Formanten). Man kann genau das gleiche Verfahren auf Spektra anwenden, um die Spektra zu glätten. Solche Spektra nennt man manchmal **DCT-** oder (öfter) **Kepstral-geglättete Spektra**. Wie in der Analyse von Formanten: je mehr Sinusoiden summiert werden, umso mehr nähert das glatte Signal (Spektrum) dem ursprünglichen Signal (rauhen Spektrum).

In stimmhaften Lauten: Der DCT-Koeffizient, der f0 verschlüsselt, ist ein hoher Koeffizient (nennen wir ihn Kf0), weil sich f0 im Spektrum 'schnell' ändert. (Siehe zB plot(e.dft[0:4000] von einem "E"). Daher entfernt man im Spektrum den Einfluss von f0, wenn die Summierung K0, K1, K2.... den Koeffizienten Kf0 *nicht* einschließt).

```
kglatt = fapply(d5[,500:7500], dct, 5, fit=T)
```

```
par(mfrow=c(1,2))
plot(d5[,500:7500], seg.l)
plot(kglatt, seg.l)
```

Rauh auf glatt überlagern, für irgendeinen Segment, hier den 10en.

```
j = 10
ylim = range(d5[j,500:7500], kglatt[j,])
plot(d5[j,500:7500], ylim=ylim)
par(new=T)
plot(kglatt[j,], ylim=ylim, col="red")
```

### 4.3 Bark- oder Mel-Frequenz Kepstral-Koeffiziente

Die Anwendung einer DCT auf ein Mel- oder Bark-transformiertes Spektrum. Wird eher in der automatischen Spracherkennung verwendet, weil nach einer Mel- oder Bark-Skalierung weniger DCT-Koeffiziente benötigt werden, um zwischen Lauten zu trennen.

**Die Mel- und Bark-Skalen**

Sind logarithmische Skalen ab ca. 1000 Hz. Mel: eine Verdoppelung der wahrgenommenen Tonhöhe entspricht ca. eine Mel-Verdopperlung. Bark-Skala ist auch im Verhältnis zu Eigenschaften vom Basilar-Membrane (die Energie soll in der Basilar-Membrane in Abständen von 1 Bark summiert werden).

```
par(mfrow=c(1,2))
plot(0:10000, mel(0:10000), type="l")
plot(0:10000, bark(0:10000), type="l")
```

```
# Spektrale Werte in Mel
plot(d5[1,], type="l", xlab="Frequenz (Hz)")
plot(mel(d5[1,]), type="l", xlab="Frequenz (Mel)")
```

**Mel-skalierte DCTs, Bark-skalierte DCTs**

```
# Spektrum 500-7500 Hz in Mel
d5m = mel(d5[,500:7500])

# Mel-skalierte DCTs
m = fapply(d5m, dct, 5)
par(mfrow=c(1,2))
eplot(k[,2:3], seg.l, dopoints=T, xlab="K1", ylab="K2", main="Hz-skaliert")
eplot(m[,2:3], seg.l, dopoints=T, xlab="K1", ylab="K2", main="Mel-skaliert")

boxplot(m[,4] ~ seg.l)


# Bark-skalierte DCTs für die Vokale in vowlax.fdat.5 berechnen
Sprecher-Etikettierungen: vowlax.spkr
Vokal-Etikettierungen: vowlax.l

# Hier wird nur 200 - 5000 Hz berücksichtigt, da kaum phonetische Informationen
# für die Vokal-Trennung außerhalb diesem Frequenz-Bereich vorhanden ist

b = fapply(bark(vowlax.dft.5[,200:5000]), dct, 3)

# Sprecherin 68
temp = vowlax.spkr == "68"
# aehnliche Darstellung wie bei den Formanten (siehe auch Aufgabe 4)
par(mfrow=c(1,2))
eplot(b[temp,2:3], vowlax.l[temp], centroid=T, xlab="Bark-DCT-1", ylab="Bark-DCT-2")
eplot(vowlax.fdat.5[temp,1:2], vowlax.l[temp], centroid=T, form=T, xlab="F2 (Hz)",
ylab="F1 (Hz)")

# aber nicht immer (Sprecher 67)
eplot(b[!temp,2:3], vowlax.l[!temp], centroid=T)
# jedoch
eplot(b[!temp,c(2,4)], vowlax.l[!temp], centroid=T)
```

(Also: meistens werden mehr Bark-skalierte DCTs also Formanten benötigt, um Vokale voneinander zu differenzieren).

**Fragen**

1. Laut Halle, Hughes & Radley (1957) lassen sich die zwei Hauptallophone von /k/, die vor vorderen und hinteren Vokalen auftreten, durch *a* - *b* differenzieren, wo *a* und *b* auf diese Weise definiert werden:

*a*: die Summe der Energie im Bereich 700 - 9000 Hz
*b*: die Summe der Energie im Bereich 2700 - 9000 Hz.

Bestätigen Sie ob dies der Fall ist (oder nicht) an hand von einem Boxplot für die folgendenen Daten:

keng
Segmentliste der Aspiration von /k/ vor vorderen /ɪ, ɛ/ und hinteren /ɔ, ʊ/ Vokalen

keng.dft.5
Eine der Segmentliste zum zeitlichen Mittelpunkt abgeleteiten spektralen Matrix

keng.l
Etikettierungen: `front` oder `back`, je nach dem ob /k/ vor vorderen oder hinteren Vokalen auftrat.

2. Wenn die Lippenrundung eine antizipatorische koartikulatorische Wirkung auf /z/ ausübt, (z.B. *Suppe* vs. *Sippe*) welche Unterschiede würden Sie dann in den /z/-Spektra vor gerundeten und nicht gerundeten Vokalen erwarten? Machen Sie eine Abbildung der Spektra zum zeitlichen Mittelpunkt von /z/ vor gerundeten und nicht gerundeten Vokalen, um Ihre Hypothese zu prüfen. Die benötigten Daten sind wie folgt:

| | |
|---|---|
| `sib` | Segmentliste, silbeninitialer [z] vor [i:, ɪ, u:, ʊ], |
| `sib.l` | Etikettierungen: `f`, für [z] vor [i:, ɪ], `b` für [z] vor [u:, ʊ] |
| `sib.w` | Wortetikettierungen |
| `sib.dft` | Spektrale Trackdatei |

3. Die Merkmale 'diffuse' und 'compact' werden in der Phonologie manchmal verwendet, um Segmente zu bezeichnen, in denen die Energie im Spektrum verbreitet (diffuse) oder sich eher in einem Bereich konzentriert ('compact').

3.1 Wie könnten Spektralmomente eingesetzt werden, um eventuell Segmente mit den Eigenschaften 'diffuse' und 'compact' voneinander zu trennen?

3.2 Laut Blumstein & Stevens (1979) lassen sich /b, d/ von /g/ durch dieses Merkmal trennen: in /b, d/ ist das Spektrum 'diffuse' und in /g/ 'compact' im Bereich 0-4 kHz. Prüfen Sie, inwiefern diese Annahme bestätigt werden kann, indem Sie eine Abbildungen machen von diesen Daten:

| | |
|---|---|
| `stops10` | Spektrale Matrix, zeitlicher Mittelpunkt vom Burst |
| `stops10.lab` | Vektor von Plosiv-Etikettierungen |

3.3 Berechnen Sie im 0-4 kHz der Spektralmoment, der am deutlichsten /b, d/ von /g/ trennen könnte, und erstellen Sie die Ergebnisse in einem Boxplot getrennt für /b, d, g/. Gibt es irgend welche Beweise, dass der Unterschied 'diffuse' vs 'compact', wie von Blumstein & Stevens

(1979) behauptet, zutrifft?

3.4 /b, d/ sollen sich laut Blumstein & Stevens (1979) ferner durch die spektrale Neigung trennen. Insbesondere soll in /b/ das Spektrum mit zunehmender Frequenz fallend, und in /d/ steigend sein. Gibt es irgendwelche Beweisungen dafür aus Ihrer Abbildung in 3.2? In welchem Frequenzbereich trifft dies zu (wenn überhaupt)?

3.5 Das erste Kepstralkoeffizient (K1) soll die spektrale Neigung verschlüsseln. Berechen Sie die ersten 3 Kepstralkoeffiziente (also K0, K1, K2) in dem von Ihnen in 3.4 gewählten Frequenzbereich, und erstellen Sie von K1 einen Boxplot für die drei Plosive. Wird dadurch /b/ von /d/ getrennt?

3.6 K2 soll die Krümmung verschlüsseln. Machen Sie eine Ellipse-Abbildung von /b, d, g/ im Raum K1 x K2. Inwiefern werden die Plosive in diesem Raum voneinander getrennt? Und wird /g/ hauptsächlich wegen der Krümmung von den anderen Plosiven durch K2 getrennt?

3.7 Die Neigung von einem Spektrum lässt sich auch erkennen, wenn Sie eine lineare Regression auf einem Spektrum überlagern, z.B. hier für den ersten Segment:

```
neigung = lm(stops10[1,] ~ trackfreq(stops10[1,]))
plot(stops10[1,])
abline(neigung)
```

Der Neigungskoeffizient selbst wird durch
`neigung$coeff[2]`

gegeben. Das ganze kann in einer Funktion gepackt werden, um den Neigungskoeffizient zu geben:

```
nfun <- function(spectrum)
{
neigung = lm(spectrum ~ trackfreq(spectrum))
neigung$coeff[2]
}
```

Wie könnten Sie diese Funktion auf die spektrale Matrix stops10 anwenden, um einen Neigungskoeffizient pro Segment zu bekommen? Bestätigen Sie anhand eines Boxplots, dass Sie eine ähnliche Trennung mit den von Ihnen berechneten Neigungskoeffizienten bekommen, wie durch K1 in 3.5.

4.
(a) Machen Sie eine grobe Skizze der Spektra der Vokale [ɪ, ʊ, a] (ein Spektrum pro Vokal) im Bereich 0 - 3000 Hz. Wie müssten sich die Vokale im Raum K1 x K2 unterscheiden?

(b) Prüfen Sie Ihre Hypothese aus (a) in dem Sie eine Ellipse-Abbildung im K1 x F2 Raum für die Vokale der Segmentliste timevow erstellen. Die Segmentliste ist aus der `timetable` Sprachdatenbank und die K1 x K2 Werte sollen zum zeitlichen Mittelpunkt der Vokale erstellt werden.

(c) Machen Sie eine überlagerte Abbildung von einem rauhen und ein geglättetem Spektrum von irgend einem [I] Vokal zum zeitlichen Mittelpunkt aus der Segmentliste in timevow im Bereich 0 - 4000 Hz an hand von einer Berechnung der ersten 5 (0, 1, ..5) Koeffiziente.