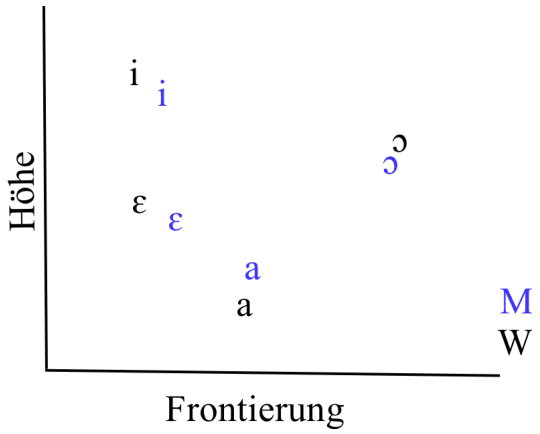
Formantanalyse und euklidsche Entfernungen

R-Objekte: vowlax, vowlax.l, vowlax.fdat, vowlax.spkr

0. Fragestellung und Vorgang
Sind die Vokale der Sprecherin 68 periphärer im Vergleich zu denjenigen vom Sprecher 67?

Hypothese:



Vorgang
1. Formantfehler im Vokal finden

2. Formanten glätten und der geglätteten Formanten die Werte zum zeitlichen Mittelpunkt entnehmen.

3. Getrennt pro Sprecher den **Zentroiden** berechnen

4. Getrennt pro Sprecher die **euklidsche Entfernung** jedes Vokals zum Zentroiden berechnen.

5. Boxplots erstellen dieser E-Entfernungen getrennt pro Sprecher und Vokal-Kategorie.

Literatur zu 3. und 4.: Siehe Wright (2003), *Laboratory Phonology* VI.

1: Formantfehler finden

```
# F2-Abbildung Trackdatei vowlax.fdat
# vowlax Segmentliste, vowlax.l, Etikettierungen
dplot(vowlax.fdat[,2])

nullfun <- function(frames, schwelle=50)
{
# T wenn die Werte unter 50 sind (Default)
any(frames < schwelle)
}

# Anwendung auf Frames möglich?
nullfun(vowlax.fdat[10,2]$data)


# Ja. Daher mit trapply() auf alle Segmente anwenden
# (Output = logischer Vektor der T ist, wenn ein Segment mindestens
# einen Wert hat mit F2 < 50 Hz
temp = trapply(                )
```

Frage: für welche Segmente (aus der Segmentlliste vowlax) ist F2 < 50?

Für welche Segmente ist F1 oder F2 < 50?

```
# Default ändern: die Schwelle ist jetzt < 130 Hz.
nullfun(vowlax.fdat[10,2]$data, 130)

# Segmente finden, die mindestens einen Wert haben mit F2 < 130 Hz.
temp = trapply(  )
```

2: Glättung mit der DCT (diskrete Cosinus Tranformation)

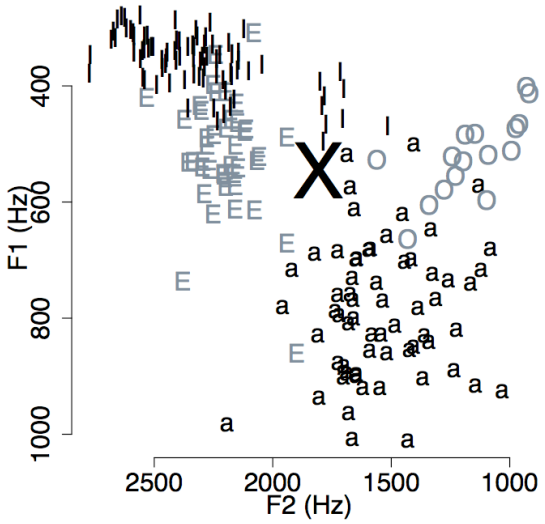
Eine DCT zerlegt ein Signal in Cosinuswellen zunehmender Frequenz (0, 0.5, 1, 1.5 ... Schwingungen) auf eine solche Weise, dass das Signal durch deren Summierung genau rekonstruiert wird - also eine Art von Fourier-Transformation in der die Phase = 0.

Das Signal wird geglättet indem nur die ersten 2-3 berechneten Cosinuswellen (z.B. mit 0, 0.5, 1 Schwingung) summiert werden.

```
# DCT-Anwendung auf ein Signal
vec = bridge[,2]
vec.dct = dct(vec)
# Die Amplituden der Cosinuswellen mit Schwingungen 0, 0.5, 1, 1.5... über
# denselben Zeitraum wie das ursprüngliche Signal

vec.dct
```

```
# Die ersten 3 Schwingungen:
cr(vec.dct[1], k=0, N=length(vec))
cr(vec.dct[2], k=0.5, N=length(vec))
cr(vec.dct[3], k=1, N=length(vec))

# Die Summierung davon
cr(vec.dct[1:3], k = c(0, .5, 1), N=length(vec))

# Einfacher und zum ursprünglichen Signal skaliert: fit=T Argument verwenden

vec.dctfit = dct(vec, 2, fit=T)

par(mfrow=c(1,2))
plot(vec, type="b"); plot(vec.dctfit, type="b")

# Die F2-Frames  vom 10en Segment glätten (3 Koeffiziente)
dct(vowlax.fdat[10,2]$data, 2, fit=T)

# F2 aller Segmente mit trapply() glätten. Wieso nicht simplify=T?
f2g = trapply(  )

# Als Trackdatei erstellen
f2g = trapply(   )
is.trackdata(f2g)

# F1 glätten
f1g = trapply(     )

# Trackdatei bestehend aus F1 und F2
g = cbind(f1g, f2g)

# Abbildung: Rauh (links), glatt (rechts)
par(mfrow=c(1,2))
plot(vowlax.fdat[10,1:2], type="b")
plot(g[10,1:2], type="b")

# F1 rauh und glatt vom 15en Segment überlagern
d = rbind(vowlax.fdat[15,1], g[15,1])
lab = c("r", "g")
dplot(d, lab)

# F1 rauh und glatt Segmente 15-20. Linker Mausklick nach jeder Abb.
lab = c("r", "g")
for(j in 15:20){
d = rbind(vowlax.fdat[j,1], g[j,1])
dplot(d, lab)
locator(1)
}
```

3. Den Mittelpunkt (Zentroid) im F2 x F1 Raum der Vokale berechnen



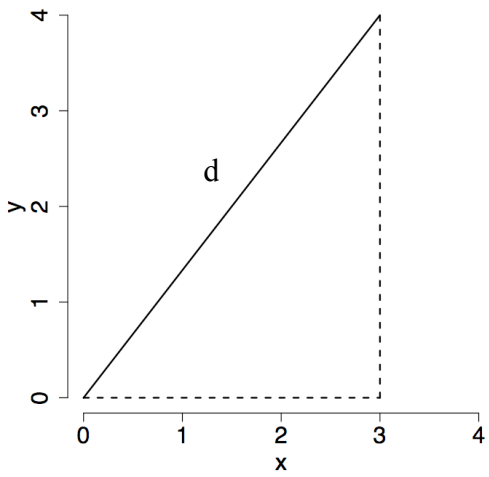
X ist [m2, m1] der Zentroid von F2 x der Mittelwert von F1

F1 x F2 den geglätteten Formanten zum zeitlichen Mittelpunkt entnehmen
g50 =

# Mittelwert pro Spalte (=Zentroid) der Matrix bridge
apply(bridge, 2, mean)

Mittelwert von F1 und F2 (=Zentroid) in g50?
m =

4. Euklidsche Entfernung zwischen zwei Punkten



Wir wollen die Entfernung, *d*, zwischen [0, 0] und [3,4] berechnen. Der Abstand entlang der geraden Linie ist die *euklidsche Entfernung*

```
a = c(0, 0)
b = c(3, 4)
# Länge entlang der x-Achse
lx = a[1] - b[1]
# Länge entlang der y-Achse
ly = a[2] - b[2]

# Pythagoras      d = sqrt(lx^2 + ly^2)
d = sqrt(lx^2  + ly^2)

# Einfacher (und auch in 3, 4, ... n Dimensionen gültig)
d = sqrt(sum((a - b)^2))
```

Wir wollen die euklidsche Entfernung zwischen jeder Reihe von g50 und m (dem Zentroiden) berechnen, also zwischen
g50[1,] und m, g50[2,] und m, g50[3,] und m...

Entweder eine for-Schleife oder eine Funktion schreiben und mit apply() auf jede Reihe der Matrix g50 anwenden. Die Funktion:

```
efun <- function(a, b)
{
sqrt(sum((a - b)^2))
}
```

Anwendung auf jede Reihe von g50:

```
apply(g50, 1, efun, m)
```

Die Daten sind aber von beiden Sprechern zusammen. Ich will jetzt die euklidschen Entfernungen getrennt pro Sprecher berechnen, d.h.:

- die E-Entfernungen für Sprecher 67 von seinen Vokalen zu seinem Zentroid
- die E-Entfernungen für Sprecherin 68 von ihren Vokalen zu ihrem Zentroid.

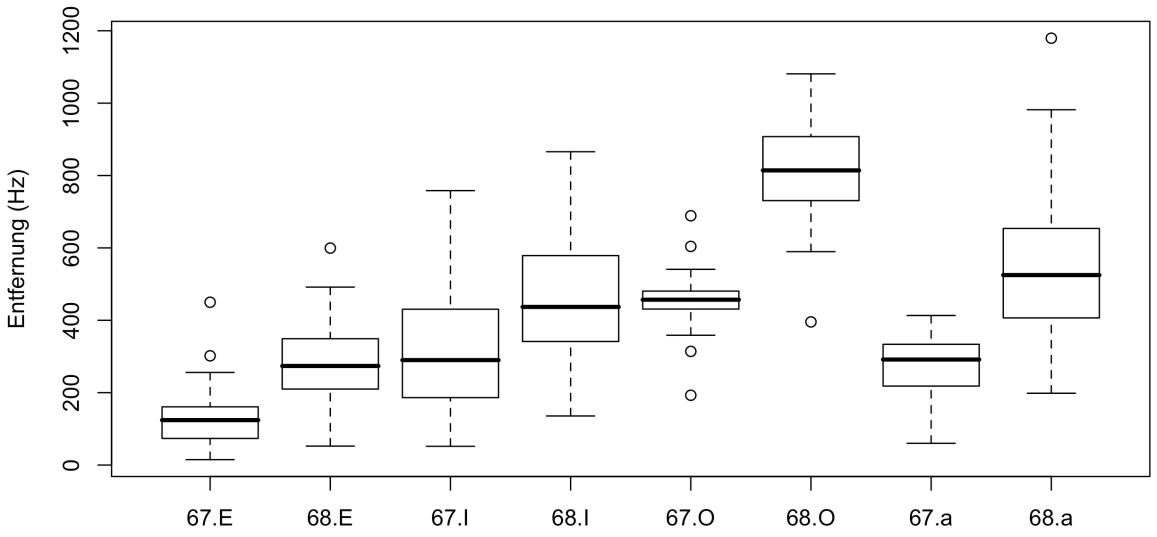
```
# Logischer Vektor: T für Sprecher 67
temp = vowlax.spkr == "67"

# [F1,F2] Mittelpunkt für Sprecher 67?
m67 =

# [F1,F2] Mittelpunkt für Sprecherin 68?
m68 =
```

```
# Vektor erstellen mit 0 derselben Länge wie g50
ent = rep(0, nrow(g50))

# E-Entfernungen für 67
ent[temp] =

# E-Entfernungen für 68
ent[!temp] =
```

5. Boxplots

Da ent parallel ist zu allen anderen vowlax Objekten, ist es einfach Boxplots zu erstellen, um die Sprecher miteinander zu vergleichen.

```
boxplot(ent ~ vowlax.spkr)

# getrennt pro Vokal und Sprecher
boxplot(              )
```



6. Das Ziel

Eine ähnliche Abbildung mit Euren Vokaldaten (Euklidsche Entfernungen im geglätteten F1 x F2 Raum) erstellen: auf der *x*-Achse in der Abbildung oben wäre dann zB p.i, s.i, p.a, s.a, p.o, s.o.

p.i = [I] Vokale mit primärer lexikalischen Betonung
s.o = [O] Vokale mit sekundärer lexikalischer Betonung
usw.