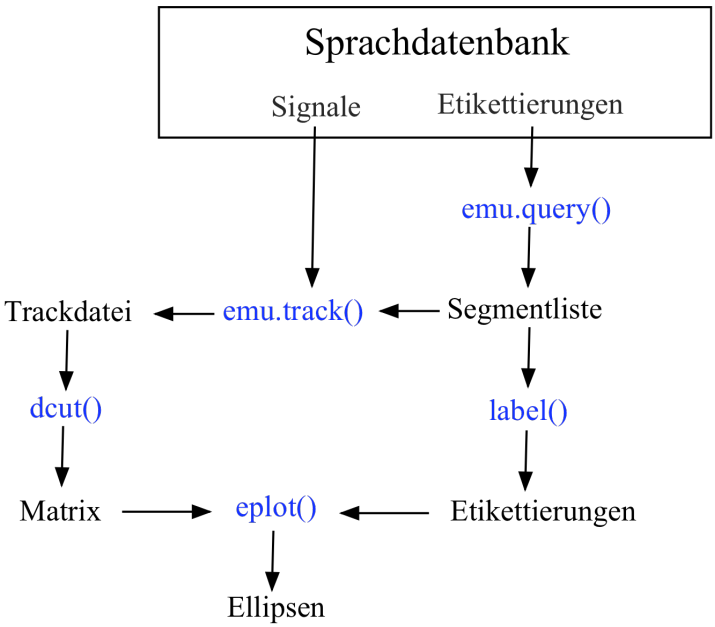
Ellipse-Abbildungen von Formanten in R



`second`: Name der Sprachdatenbank
`gam*`: Nur in den Äußerungen suchen, die mit *gam* beginnen
`Phonetic`: Ebene, aus der die Segmente entnommen werden
`i:|e:|a:|o:|u:` : "i:", oder "e:", oder "a:", oder "o:", oder "u:" Vokale dieser Ebene entnehmen
vok.s = emu.query("second", "gam*", "Phonetic=i:|e:|a:|o:|u:")

Ein Vektor nur mit den Vokaletikettierungen
vok.l = label(vok.s)

Eine sogenannte Trackdatei. vok.fm enthält alle Formantwerte, F1-F4, zwischen den Start- und Endzeiten aller Segmente in vok.s
vok.fm = emu.track(vok.s, "fm")

Mit dem Befehl dcut() entnehmen wir der Trackdatei F1 und F2 zum zeitlichen Mittelpunkt von jedem Vokal (prop bedeutet: die Zeiten werden *proportional* in der Zeit entnommen)

vok.fm5 = dcut(vok.fm[,1:2], .5, prop=T)

Eine Ellipse-Abbildung im Raum F1 und F2. form=T dreht die Achsen, damit F2 auf der *x*-Achse, F1 auf der *y*-Achse erscheinen, und sodass die Werte von links nach rechts (F2) und von unten nach oben (F1) fallen.

eplot(vok.fm5, vok.l, form=T, dopoints=T)

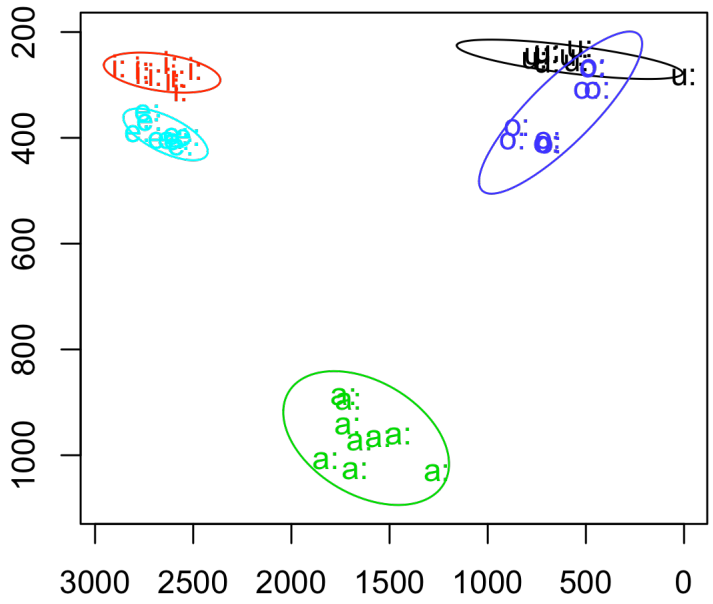
Nehmen wir an, wir möchten die Äußerung(en) mit /a:/ finden, in denen F1 höher als 650 Hz ist. (Diese Methode werden wir ggf. später benötigen, um Äußerungen zu finden, in denen Formantfehler auftreten)

Ein logischer Vektor. `T` wenn F1 > 650 Hz und die Etikettierung ist "a:"

```
temp = vok.fm5[,1] > 650 & vok.l == "a:"
vok.s[temp,]
segment  list from database:  second
query was:  Phonetic=i:|e:|a:|o:|u:
   labels   start     end   utts
14     a: 473.912 647.072 gam023
```

Aufgabe
1. Bitte jetzt das gleiche für die Sprecherin **agr** wiederholen. In tkassp Nominal F1 auf 600 Hz setzen. Dann z.B.

```
# Segmentliste
agr.s = emu.query("second", "agr*", "Phonetic=i:|e:|a:|o:|u:")
# Label-Vektor
agr.l = label(agr.s)
# Formantrackdatei
agr.fm = emu.track(agr.s, "fm")
# Formanten zum zeitlichen Mittelpunkt
agr.fm5 = dcut(agr.fm[,1:2], .5, prop=T)
# Ellipsen
eplot(agr.fm5, agr.l, form=T, dopoints=T)
```



Es gibt einen Formantfehler in /u:/, der noch korrigiert werden muss. In Emu korrigieren (nachdem Sie in R den Vokal gefunden haben), dann die obigen Befehle wiederholen.

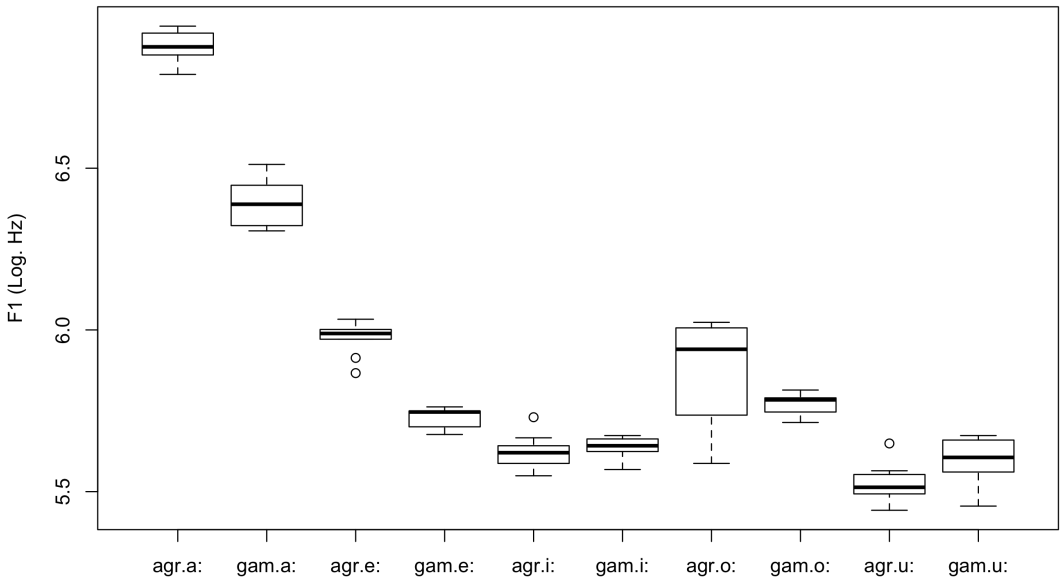
2. Erstellen Sie einen Data-Frame mit kontinuierlichen Variablen F1 zum zeitlichen Mittelpunkt, F2 zum zeitlichen Mittelpunkt, der Vokaldauer, und mit Faktoren

Versuchsperson und Vokal. Der Eigenschaften des Data-Frames (genannt hier d.df) müssten wie folgt sein:

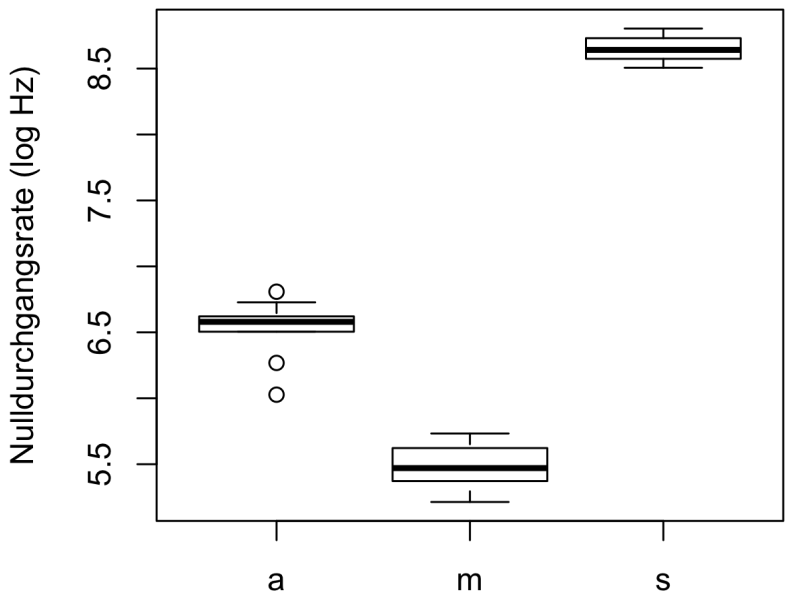
```
head(d.df)
   F1   F2   dauer Vpn Vokal
1 260  889 101.030 gam    u:
2 234  539 134.132 gam    u:
3 287  732 157.796 gam    u:
4 291 1994 104.372 gam    i:
5 282 1961  78.757 gam    i:
6 291  765 108.879 gam    u:

dim(d.df)
[1] 90  5

with(d.df, table(Vpn))
# oder
table(d.df$Vpn)
Vpn
agr gam
 45  45

with(d.df, table(Vokal))
# oder
table(d.df$Vokal)
Vokal
a: e: i: o: u:
18 18 18 18 18
```

3. Verwenden Sie den Data-Frame um einen Boxplot der log. F1-Werte als Funktion von Vokal und Sprecher zu erstellen, wie unten:

(man bekommt Logarithmen mit log(x), wo x ein numerischer Vektor ist)



4. Für die Sprachdatenbank `timetable` erstellen Sie einen Boxplot der log. zcr-Werte (Nulldurchgangsrate in log. Hz) zum zeitlichen Mittelpunkt in den Segmenten "a", "m", und "s" wie unten:



Weitere Befehle, um Trackdateien zu manipulieren
(N.B.: diese funktionieren nur nach library(emu))

Allgemein
# Bestätigen, dass es eine Trackdatei ist
class(vok.fm)
[1] "trackdata"

Eigenschaften der Trackdatei
summary(vok.fm)

dim(vok.fm)

nrow(vok.fm)

ncol(vok.fm)

Indizierung (wie für eine Matrix)
F1 und F2 zwischen den Start- und Endzeiten von Segment 1
neu.fm = vok.fm[1,1:2]

F2 aller Segmenten
neu.fm = vok.fm[,2]

F1-F3 von Segmenten 10, 11, 13-15
neu.fm = vok.fm[c(10, 11, 13:14),1:3]

Die Indizierung erfolgt auch durch logische Vektoren. z.B. Formanten der "a:" Vokale
temp = vok.l == "a:"
neu.fm = vok.fm[temp,]

F1 und F2 der "a:" und "e:" Vokale
temp = vok.l %in% c("a:", "e:")
neu.fm = vok.fm[temp,1:2]

Abbildungen als Funktion der Zeit
plot(): um einen Track, oder Tracks **einzelner Segmente** abzubilden
dplot() um Track(s) **mehrerer Segmente aufeinander zu überlagern**

F1-F4 von 10en Segment
plot(vok.fm[10,])

oder

dplot(vok.fm[10,])

F2 von "e:" und "o:" Segmenten kodiert nach Farbe
temp = vok.l %in% c("e:", "o:")
dplot(vok.fm[temp,2], vok.l[temp])

Vertiefung
siehe *Zusätzliche Informationen zu Trackdateien* in der Webseite.