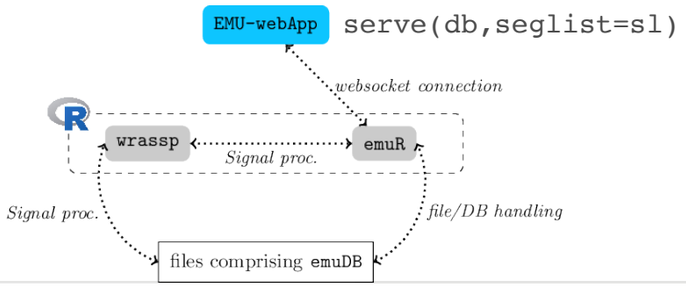
emuR-seminar CHEAT SHEET



Prerequisites

You should have:

```
mypath = "/PATH/WHERE/YOUR/DATABASES/ARE"
pfadu = "http://www.phonetik.uni-muenchen.de/~jmh/lehre/Rdf"
```

Installation of packages:

tidyverse:

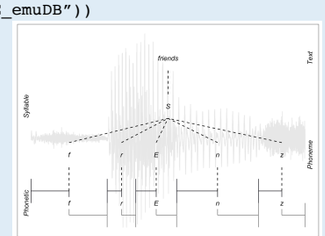
```
install.packages("tidyverse")
```

Newest version of emuR:

```
install.packages("gridExtra")
install.packages("devtools")
devtools::install_github("IPS-LMU/emuR")
```

Basics

```
library(emuR)
library(tidyverse)

db=load_emuDB(file.path(mypath,"DBNAME_emuDB"))

summary(db)
#or
list_levelDefinitions(db)
list_linkDefinitions(db)
list_ssffTrackDefinitions(db)
```



QUERIES

```
sl = query(db, "Phonetic == m")
```

Queries using regular expressions, e.g. ANY phonetic symbol

```
sl = query(db, "Phonetic =~ .*")
```

NOT "H*" on the "Tone" level

```
sl = query(db, "Tone != H*")
```

sequences

```
sl = query(db, "[Phonetic == @ -> Phonetic == n]")
```

first element of a sequence only

```
sl = query(db, "[#Phonetic == @ -> Phonetic == n]")
```

second element of a sequence only

```
sl = query(db, "[Phonetic == @ -> #Phonetic == n]")
```

conjunction queries

```
sl = query(db, "[Text == always & Accent == S]")
```

domination

```
sl = query(db, "[Phoneme == p ^ Syllable == S]")
```

query all phoneme items that occur
at the start of a syllable (see also `Mid` or `End`)

```
sl = query(db, "[Start(Syllable, Phoneme) == TRUE]")
```

See also the EQL chapter in the EMU-SDMS manual.

REQUERIES

sequential, e.g. the segment before

```
requery_seq(db, seglist = sl, offset = -1, calcTimes=TRUE)
```

hierarchical

```
requery_hier(db, seglist = sl_s,level = "Phonetic",
calcTimes=FALSE)
```

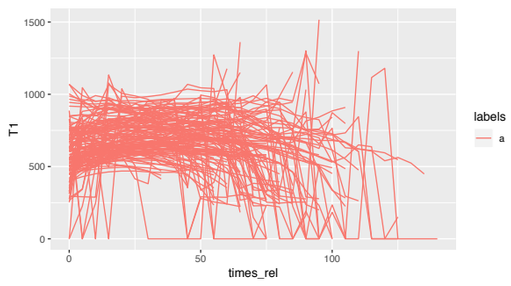
Formants

List currently available tracks

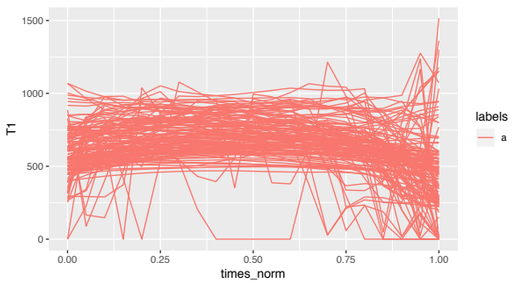
```
list_ssffTrackDefinitions(db)
```

Formants (T1…T4)

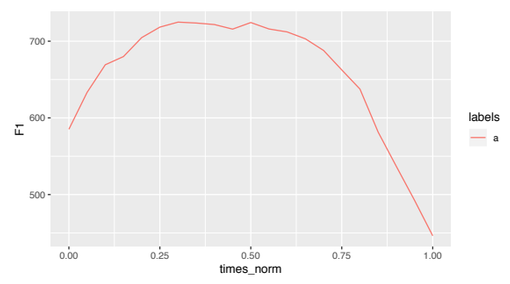
```
sl.fm = get_trackdata(db,
   seglist = sl,
   ssffTrackName = "FORMANTS",
   resultType = "tibble")
ggplot(sl.fm)+geom_line(
    aes(x=times_rel,y=T1,
    group=sl_rowIdx,col=labels))
```




```
sl.fm_norm = normalize_length(sl.fm)
ggplot(sl.fm_norm)+
    geom_line(aes(x=times_norm,y=T1,
    group=sl_rowIdx,col=labels))
```

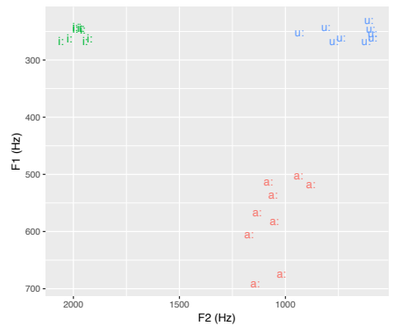



```
sl.fm_norm_average = sl.fm_norm %>%
group_by(labels,times_norm) %>%
summarise(F1=mean(T1))
ggplot(sl.fm_norm_average)+
 geom_line(aes(x=times_norm,y=F1,col=labels
```



extract at segments temporal midpoint

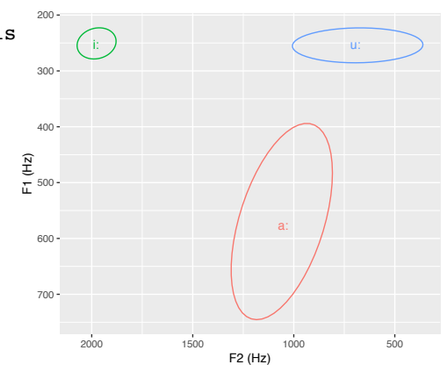
```
sl.fm.05 = get_trackdata(second,
        seglist = sl,
        ssffTrackName = "FORMANTS",
        resultType = "tibble",cut=0.5)

ggplot(sl.fm.05) +
aes(x=T2,y=T1, col=labels, label=labels)
+ geom_text() +
scale_y_reverse() + scale_x_reverse()+
labs(x = "F2 (Hz)", y = "F1 (Hz)") +
theme(legend.position="none")
```




```
sl.fm.05.centroid = sl.fm.05 %>%
  group_by(labels) %>%
  summarize(F1=mean(T1), F2=mean(T2))
```

plot both the centroids and ellipses
for all data

```
ggplot(sl.fm.05) +
aes(x=T2,y=T1, col=labels,label=labels)+
stat_ellipse() +geom_text(data=sl.fm.
05.centroid,aes(x=F2,y=F1)) +
scale_y_reverse() + scale_x_reverse() +
labs(x = "F2 (Hz)", y = "F1 (Hz)") +
theme(legend.position="none")
```



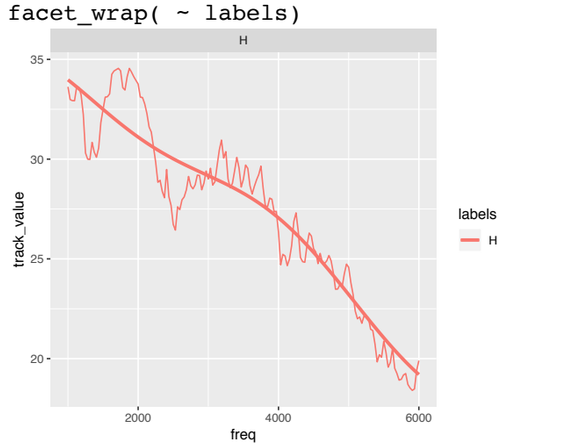
Spectral analyses

extract at segments temporal midpoint

```
sl.dft = get_trackdata(db,seglist =
sl,ssffTrackName = "dft",resultType
= "tibble",cut=0.5)
sl.dft.long =
convert_wideToLong(sl.dft,
calcFreqs = TRUE)
```

calculate smoothed spectra and plot both
for frequency range 1000-6000 Hz

```
sl.dft.long = sl.dft.long %>%
  group_by(sl_rowIdx) %>%
  mutate(smoothed
  =emuR::dct(track_value,m=4,fit=T))

sl.dft.long.mean = sl.dft.long %>%
  filter(freq >=1000 & freq <= 6000) %>%
  group_by(labels,freq) %>%
  mutate(track_value=mean(track_value),
  smoothed=mean(smoothed))
ggplot(sl.dft.long.mean) +
  aes(x = freq,col=labels) +
  geom_line(aes(y = track_value)) +
  geom_line(aes(y = smoothed),lwd = 1.2) +
  facet_wrap( ~ labels)
```



calculate spectral moments

```
sl.moments=sl.dft.long %>%
group_by(labels,sl_rowIdx) %>%
do(tibble(Moments = emuR::moments(.$track_value,
.$freq,minval=TRUE))) %>%mutate(Momentnumbers =
paste0("Moment",1:(table(sl_rowIdx)))) %>%
tidyr::spread(Momentnumbers, Moments)
```