```
library(emu)
library(lattice)
library(latticeExtra)
library(RCurl)
source(file.path(pfadu, "readtrackfromurl.R"))
source(file.path(pfadu, "ellipsefun.R"))

source(file.path(pfadu, "tracktodf.R"))
source(file.path(pfadu, "tracklinear.R"))

# aggregate plots as a function of time
dim(vowlax)
dim(vowlax.fdat)
table(vowlax.l)
table(vowlax.spkr)

dplot(vowlax.fdat[,2], vowlax.l)
# mit Lattice
# 1. Konvertierung Trackdatei in Data-Frame
vowlax.df = tracktodf(vowlax.fdat)
# ggf neue Namen vergeben
names(vowlax.df)[1:4] = paste("F", 1:4, sep="")

# 2. ggf. weitere Label-Dateien einbinden
vowlax.df.l = rep(vowlax.l, table(vowlax.df$segno))
vowlax.df.spkr = rep(vowlax.spkr, table(vowlax.df$segno))
vowlax.df = data.frame(vowlax.df, V = factor(vowlax.df.l), Vpn = factor(vowlax.df.
spkr))

# 3. Verschlüsselung für die Farb-Kodierung
# Für die Farben im xyplot
vowlax.col = as.numeric(factor(vowlax.l))
# Für die Beschriftung
vowlax.leg = list(
text = list(unique(vowlax.l)),
lines = list(col = unique(vowlax.col)))

# xyplot()
# Parameter1 ~ Paramter2
# segno: gruppiert die Werte pro Segment
# type = c("l", "g"): Line-Plot mit Grid
# key = für die Beschriftung
# data = Data-Frame
# par.settings: um jeder Linie die richtige Farbe zuzuordnen

xyplot(F2 ~ times, group = segno, type = c("l", "g"), key = vowlax.leg, data = vowlax
.df,
par.settings = list(
superpose.line = list(col = vowlax.col)
))

xyplot(F2 ~ times | Vpn, group = segno, type = c("l", "g"), key = vowlax.leg, data =
vowlax.df,
```

```
par.settings = list(
superpose.line = list(col = vowlax.col)
))

xyplot(-F1 ~ -F2 | Vpn, group = segno, type = c("l", "g"), key = vowlax.leg, data =
vowlax.df,
par.settings = list(
superpose.line = list(col = vowlax.col)
))

# synchronisiert zum zeitlichen Mittelpunkt
# Mittelpunkt
m = (start(vowlax) + end(vowlax))/2
mdf = rep(m, table(vowlax.df$segno))
mdf = vowlax.df$otimes - mdf
# aufrunden auf 5 ms
mdf = 5 * round(mdf/5)
vowlax.df = cbind(vowlax.df, midtimes= mdf)
xyplot(F2 ~ midtimes | Vpn, group = segno, type = c("l", "g"), key = vowlax.leg, data
= vowlax.df,
par.settings = list(
superpose.line = list(col = vowlax.col)
))
# entspricht:
# temp = vowlax.spkr == "67"
# par(mfrow=c(1,2))
# dplot(vowlax.fdat[temp,2], vowlax.l[temp], offset=.5)
# dplot(vowlax.fdat[!temp,2], vowlax.l[!temp], offset=.5)

# Mittelwerte
vowlax.dfm = with(vowlax.df, aggregate(vowlax.df[,1:4], list(V, Vpn, midtimes), mean)
)
names(vowlax.dfm)[1:3] = c("V", "Vpn", "midtimes")

xyplot(F2 ~ midtimes | Vpn, groups = V, type = c("l", "g"), data = vowlax.dfm, auto.
key=T)
# entspricht (fast)
# par(mfrow=c(1,2))
# dplot(vowlax.fdat[temp,2], vowlax.l[temp], offset=.5, av=T)
# dplot(vowlax.fdat[!temp,2], vowlax.l[!temp], offset=.5, av=T)

# 0. lineare Zeitnormalisierung
# trackdatei
vowlax.lin = tracklinear(vowlax.fdat)
# 1. in Data-Frame umsetzen
vowlax.lin.df = tracktodf(vowlax.lin)
names(vowlax.lin.df)[1:4] = paste("F", 1:4, sep="")
#  2. ggf. weitere Label-Dateien einbinden
vowlax.lin.df.l = rep(vowlax.l, table(vowlax.lin.df$segno))
vowlax.lin.df.spkr = rep(vowlax.spkr, table(vowlax.lin.df$segno))
vowlax.lin.df = data.frame(vowlax.lin.df, V = factor(vowlax.lin.df.l), Vpn = factor
(vowlax.lin.df.spkr))
```

```
# 3. Verschlüsselung für die Farb-Kodierung
# Das gleiche wie für 3. nicht linear zeitnormalisiert

xyplot(F2 ~ times | Vpn, group = segno, type = c("l", "g"), key = vowlax.leg, data =
vowlax.lin.df,
par.settings = list(
superpose.line = list(col = vowlax.col)
))
# entspricht fast (würde genau entsprechen bei: vowlax.lin = tracklinear(vowlax.fdat,
20))
# par(mfrow=c(1,2))
# dplot(vowlax.fdat[temp,2], vowlax.l[temp], norm=T)
# dplot(vowlax.fdat[!temp,2], vowlax.l[!temp], norm=T)

xyplot(-F1 ~ -F2 | Vpn, group = segno, type = c("l", "g"), key = vowlax.leg, data =
vowlax.lin.df,
par.settings = list(
superpose.line = list(col = vowlax.col)
))

# lineare Zeitnormalisierung und gemittelt
vowlax.lin.dfm = with(vowlax.lin.df, aggregate(vowlax.lin.df[,1:4], list(times, V,
Vpn), mean))
names(vowlax.lin.dfm)[1:3] = c("times", "V", "Vpn")
xyplot(F2 ~ times | Vpn, group = V, type = c("l", "g"), auto.key=T, data = vowlax.lin
.dfm)

# entspricht
# par(mfrow=c(1,2))
# dplot(vowlax.fdat[temp,2], vowlax.l[temp], norm=T, av = T)
# dplot(vowlax.fdat[!temp,2], vowlax.l[!temp], norm=T, av = T)

# Siehe Physiologische unten für ein Beispiel, wie der Transitions-Onset
# identifiziert werden kann
xyplot(-F1 ~ -F2 | Vpn, group = V, type = c("l", "g"), auto.key=T, data = vowlax.lin.
dfm)




############################################################
############################################################
## Physiologie
# Segmentlisten und Label-Dateien
fric.s = read.emusegs(file.path(paste(pfadu, "polnischphys", sep="/"), "fric.s.txt"))
fric.l = label(fric.s)
# Danach kommender Vokal
vow.s = read.emusegs(file.path(paste(pfadu, "polnischphys", sep="/"), "vow.s.txt"))
vow.l = label(vow.s)
# Promptliste einlesen (Um Akzentuierung zu identifizieren)
p.df = read.table(file.path(paste(pfadu, "polnischphys", sep="/"), "p.df.txt"))
all(utt(fric.s) == p.df$U)
# [1] TRUE
```

```
# Gaumendaten einlesen
gaumen = read.table(file.path(paste(pfadu, "polnischphys", sep="/"), "gaumen.txt"))

# Physiologische Track-Datein
# tty, ttx: Zungenspitze Y, X; tby, tbx: Zungendorsum Y, X
tty = readtrackfromurl("tty.txt",file.path(paste(pfadu, "polnischphys", sep="/")) ,
pfad)
ttx = readtrackfromurl("ttx.txt",file.path(paste(pfadu, "polnischphys", sep="/")) ,
pfad)
tby = readtrackfromurl("tby.txt",file.path(paste(pfadu, "polnischphys", sep="/")) ,
pfad)
tbx = readtrackfromurl("tbx.txt",file.path(paste(pfadu, "polnischphys", sep="/")) ,
pfad)

# In eine Trackdatei einbinden
phys = cbind(tty, ttx, tby, tbx)

# 1. Trackdatei in ein Data-Frame umsetzen
phys.df = tracktodf(phys)
names(phys.df)[1:4] = c("TTY", "TTX", "TBY", "TBX")

# 2. ggf. weitere Label-Dateien einbinden
fric.df.l = rep(fric.l, table(phys.df$segno))
vow.df.l = rep(vow.l, table(phys.df$segno))
acc.df.l = rep(p.df$Acc, table(phys.df$segno))
phys.df = cbind(phys.df, K = factor(fric.df.l), V = factor(vow.df.l), Acc = factor
(acc.df.l))

# 3. Verschlüsselung für die Farb-Kodierung
# Farbkodierung für den Frikativ
fric.col = as.numeric(factor(fric.l))
fric.leg = list(
text = list(unique(fric.l)),
lines = list(col = unique(fric.col)))

# Farbkodierung für den Vokal
vow.col = as.numeric(factor(vow.l))
vow.leg = list(
text = list(unique(vow.l)),
lines = list(col = unique(vow.col)))

# TTY zwischen Onset und Offset des Frikativs
# s, si, sz = [s, ɕ, ʂ]
xyplot(TTY ~ times | V, group = segno, data = phys.df, type=c("l", "g"),
key = fric.leg,
par.settings = list(
superpose.line = list(col = fric.col)
))

xyplot(TTY ~ times | V * Acc, group = segno, data = phys.df, type=c("l", "g"),
key = fric.leg, main = "TTY",
par.settings = list(
superpose.line = list(col = fric.col)
```

```
))

xyplot(TTY ~ TTX | V * Acc, group = segno, data = phys.df, type=c("l", "g"),
key = fric.leg, main = "TTX x TTY",
par.settings = list(
superpose.line = list(col = fric.col)
))

# Es sind dieselben Daten wie oben (zwischen akustischem Onset und Offset
# des Frikativs) aber diesmal farbkodiert nach folgendem Vokal und
# mit einem Bild pro Frikativ-Kategorie
xyplot(TTY ~ TTX | K * Acc, group = segno, data = phys.df, type=c("l", "g"),
key = vow.leg, main = "TTX x TTY",
par.settings = list(
superpose.line = list(col = vow.col)
))


#####
# Mit überlagertem Gaumen
xlim = c(0, 60); ylim = c(-10, 22)

p = function(x,y,...) {
panel.xyplot(x, y, ...)
panel.points(gaumen[,2], gaumen[,3])
}

xyplot(TTY ~ TTX | V * Acc, group = segno, data = phys.df, type=c("l", "g"),
key = fric.leg, main = "TTX x TTY", panel = p, xlim=xlim, ylim = ylim,
par.settings = list(
superpose.line = list(col = fric.col)
))


### Mittelwerte
phys.dfm = with(phys.df, aggregate(phys.df[,1:4], list(times, K, V, Acc), mean))
names(phys.dfm)[1:4] = c("times", "K", "V", "Acc")
xyplot(TTY ~ times | V * Acc, groups = K, data = phys.dfm, type = c("l", "g"), auto.
key=T)

xyplot(TTY ~ TTX | V * Acc, groups = K, data = phys.dfm, type = c("l", "g"),
auto.key=T, panel=p, xlim=xlim, ylim=ylim)

# Wie oben aber hier identifizieren wir
# für die untere Reihe
# den Onset des Frikativs durch einen schwarzen Kreis

pneu = function(x,y, ...) {
panel.xyplot(x, y, ...)
if(panel.number()==1)
{
temp = phys.dfm$times == 0 & phys.dfm$V == "a" & phys.dfm$Acc == "ACC"
panel.points(gaumen[,2], gaumen[,3])
```

```
panel.points(phys.dfm$TTX[temp], phys.dfm$TTY[temp], col = "black")
}
if(panel.number()==2)
{
temp = phys.dfm$times == 0 & phys.dfm$V == "e" & phys.dfm$Acc == "ACC"
panel.points(gaumen[,2], gaumen[,3])
panel.points(phys.dfm$TTX[temp], phys.dfm$TTY[temp], col = "black")
}
if(panel.number()==3)
{
temp = phys.dfm$times == 0 & phys.dfm$V == "o" & phys.dfm$Acc == "ACC"
panel.points(gaumen[,2], gaumen[,3])
panel.points(phys.dfm$TTX[temp], phys.dfm$TTY[temp], col = "black")
}
}
xyplot(TTY ~ TTX | V * Acc, groups = K, data = phys.dfm, type = c("l", "g"),
auto.key=T, panel=pneu, xlim=xlim, ylim=ylim)
```