```
library(emu)
library(lattice)
# to intsall some trackdata objects
library(RCurl)
source(file.path(pfadu, "readtrackfromurl.R"))
# for ellipses
library(latticeExtra)
# three-point central differencing
source(file.path(pfadu, "cendiff.R"))

################################################################## Daten aus Woche 1
############################ Either:
son.lab = emu.requery(k.s, "Segment", "Segment", seq=2, j=T)
word.lab = emu.requery(k.s, "Segment", "Word", j=T)
tip.tt = emu.track(tip.s, "tt_posz")
body.tb = emu.track(body.s, "tb_posz")

############################## Or:
son.lab = scan(file.path(paste(pfadu, "physch5", sep="/"), "son.lab.txt"), what="")
word.lab = scan(file.path(paste(pfadu, "physch5", sep="/"), "word.lab.txt"), what="")
tip.tt = readtrackfromurl("tip.tt.txt",file.path(paste(pfadu, "physch5", sep="/")) ,
pfad)
body.tb = readtrackfromurl("body.tb.txt",file.path(paste(pfadu, "physch5", sep="/"))
, pfad)

########################################################## Part 1, trackapply():
# If you need to apply a function to a trackdata object
# and the function is not listed in
# Arith, Compare, Math, Math2 Funktionen of help(Ops)

o = trapply(vowlax.fdat[,1], mean, simplify=T)
o = trapply(vowlax.fdat[,1], range, simplify=T)

o = trapply(vowlax.fdat[,1], cendiff, returntrack=T)

# writing your function
mfun = function(dat, k=3, fun=mean)
{
fun(dat[1:k])
}

mfun(frames(vowlax.fdat[1,1]))
o = trapply(vowlax.fdat[,1], mfun, simplify=T)

plot(tip.tt[1,], xlim=c(1100, 1400))
par(new=T)
plot(body.tb[1,], xlim=c(1100, 1400), col=2)

afun = function(dat, fun=max)
{
zeiten = tracktimes(dat)
temp = dat == fun(dat)
zeiten[temp][1]
```

```
}

afun(frames(tip.tt[1,]))

spitze = trapply(tip.tt, afun, simplify=T)
body = trapply(body.tb, afun, simplify=T)
bwplot(spitze - body ~ son.lab)

plot(tip.tt[1,], xlim=c(1100, 1400))
par(new=T)
plot(body.tb[1,], xlim=c(1100, 1400), col=2)

########################################################## Part 2, scatter plots and
ellipses
dim(vowlax.fdat)
m = dcut(vowlax.fdat, .5, prop=T)
vowlax.l = label(vowlax)
vowlax.spkr = substring(utt(vowlax), 2, 3)
xyplot(m[,1] ~ m[,2] | vowlax.l * vowlax.spkr)
xyplot(m[,1] ~ m[,2] | vowlax.spkr, groups=vowlax.l, auto.key=T)
# See below for how to modify axis labels
xyplot(-m[,1] ~ -m[,2] | vowlax.spkr, groups=vowlax.l, auto.key=T)

# Ellipse superimposition
# the default (without level = .95) is for 1.5 standard-deviation ellipses
# they enclose 68% of the data points. Why?
# pchisq(1.5^2, 2)
# the above encloses 95% of data points
# remove center.pch=NULL to plot the ellipse-centroid

efun = function(x, y, ...) {
panel.xyplot(x, y, ...)
           panel.ellipse(x, y, level = .95, center.pch=NULL, ...)
       }

xyplot(-m[,1] ~ -m[,2] | vowlax.spkr, groups=vowlax.l, auto.key=T, panel=efun)

########################################################## Part 3, Euclidean distances
# Segment-list
v = read.emusegs(file.path(paste(pfadu, "dfg", sep="/"), "v.txt"))
# Label-vector: consonantal context
c.l = scan(file.path(paste(pfadu, "dfg", sep="/"), "c.l.txt"), what="")
# Label-vector: ±tense vowel
t.l = scan(file.path(paste(pfadu, "dfg", sep="/"), "t.l.txt"), what="")
# Formant-trackdata for the segment list v
# i.e. derived from emu.track(v, "fm")
v.fm = readtrackfromurl("v.fm.txt",file.path(paste(pfadu, "dfg", sep="/")) , pfad)

# get the vowel label
v.l = label(v)
# get the speaker labels from the utterance names
vpn = substring(utt(v), 1, 2)
# get the rate label (a=slow, b = fast from the utterance name)
```

```
rate = substring(utt(v), 3, 3)

# plot the spaces for overlaid vowels, separately for ± tense,
# data at the vowel temporal midpoint
f = dcut(v.fm, .5, prop=T)

xyplot(-f[,1] ~ -f[,2] | t.l, groups=v.l, auto.key=T)
xyplot(-f[,1] ~ -f[,2] | t.l, groups=v.l, auto.key=T, panel=efun)
xyplot(-f[,1] ~ -f[,2] | t.l * c.l, groups=v.l, auto.key=T, panel=efun)

# calculate the distance to the centre of the vowel space
# separately for tense and lax vowels
euc = function(a, b)
{
sqrt(sum((a - b)^2))
}

d = rep(0, nrow(f))
for(s in unique(vpn)){
for(k in unique(c.l)){
for(j in unique(t.l)){
temp = t.l == j & vpn == s & c.l == k
m = apply(f[temp,], 2, mean)
d[temp] = apply(f[temp,], 1, euc, m)
}
}
}

bwplot(d ~ t.l | v.l)

d.df = data.frame(d = d, Tn = factor(t.l), V = factor(v.l), Vpn = factor(vpn), K =
factor(c.l))
a = with(d.df, aggregate(d, list(Tn, K, Vpn), mean))
names(a) = c("Tn", "K", "Vpn", "d")
bwplot(d ~ Tn | K, data = a)
library(ez)
o = ezANOVA(a, .(d), .(Vpn), .(Tn, K))
source(file.path(pfadu, "phoc.txt"))
p = phoc(a, .(d), .(Vpn), .(Tn, K))
round(phsel(p$res), 3)
round(phsel(p$res, 2), 3)

# is there a greater u-y overlap in a /t/ context?
dv = rep(0, nrow(f))
for(s in unique(vpn)){
for(k in unique(c.l)){
for(j in unique(t.l)){
tempu = v.l == "u" & t.l == j & c.l == k & vpn == s
mu = apply(f[tempu,], 2, mean)
tempy = v.l == "y" & t.l == j & c.l == k & vpn == s
my = apply(f[tempy,], 2, mean)
dv[tempy] = apply(f[tempy,], 1, euc, mu)
dv[tempu] = apply(f[tempu,], 1, euc, my)
```

```
}
}
}
d.df = cbind(d.df, dv)
temp = d.df$V %in% c("u", "y")
u.df = d.df[temp,]
u.df$V = factor(u.df$V)
bwplot(dv ~ K | Tn * V, data = u.df)
a = with(u.df, aggregate(dv, list(K, Tn, Vpn), mean))
names(a) = c("K", "Tn", "Vpn", "dv")
bwplot(dv ~ K | Tn, data = a)
o = ezANOVA(a, .(dv), .(Vpn), .(Tn, K))
p = phoc(a, .(dv), .(Vpn), .(Tn, K))
round(phsel(p$res), 3)
round(phsel(p$res, 2), 3)


################################################## Modifying axis labels in xyplot

axfun =
function(side, ...) {

if(side == "bottom")
{
w = seq(-2500, -1000, by = 500)
wlab = as.character(-w)
panel.axis(side=side, at = w, labels = wlab, outside=T, rot=0)
}

if(panel.number()==1)
{
if(side == "left")
{
w = seq(-1000, -200, by = 200)
wlab = as.character(-w)
panel.axis(side=side, at = w, labels = wlab, outside=T, rot=0)

}
}
# else
# axis.default(side = side, ...)
}

xyplot(-m[,1] ~ -m[,2] | vowlax.spkr, groups=vowlax.l, auto.key=T, panel=efun, axis=
axfun)
```