Ranging_Analyses_by_Clusters_on_Penlevel

Y. Gomez

create table with Cluster Label and Variables

1. Read in Files

datafile of cluster label per hen based on distance matrices calculations using dtwclust for details see Codes “RCode.penXX.R”:

2. merge files
3. exclude cluster 5, which is one HenID only

Penlevel Analysis

code given for pen 11 only #### PEN 11

```
library(lmerTest)
library(pbkrtest)
  clus11.max.lmer <- lmer((Total_transitions) ~ mycl + (1|HenID), data=allhen11.df,REML=F)
  summary(clus11.max.lmer)
  anova(clus11.max.lmer)
  clus11.min.lmer <- lmer((Total_transitions) ~  (1|HenID), data=allhen11.df, REML=F)
  anova(clus11.max.lmer, clus11.min.lmer, test="F")
```

```
library(multcomp)
summary(glht(clus11.max.lmer, linfct = mcp(mycl="Tukey")),test=adjusted("fdr"))
```

% fdr as p-value correction = forced discovery rate (Benjamin and Hochberg,1995)

```
# Residuenanalysen #
  par (mfrow= c (2, 2))
  qqnorm (resid (clus11.max.lmer))
  qqnorm (ranef (clus11.max.lmer) [['HenID']] [, 1])
  scatter.smooth (fitted (clus11.max.lmer), resid (clus11.max.lmer))

# Random Analyse #
  for (ranEF in names (ranef (clus11.max.lmer))) {
    qqnorm (ranef (clus11.max.lmer) [[ranEF]] [, 1])
  }
```

% strong effect of heteroscedasticity ==> overall variance is forced to homos. but it should not affect the slope. % therefore the effect size won’t change due to the strong heteroscedasticity % however, the p-values will be affected by the hetero ==> % therefore I tried not to use normal lmer, but robust lmer (Package: robustlmm) large parameter k values

(tuning parameter) yield more efficient, but less robust estimates, whereas smaller values yield more robust but less efficient estimates. A popular choice is to fix the asymptotic efficiency at 95% of the classic estimates (k = 1.345 for Huber function) (aus Paper: Koller 2016)

```
library(robustlmm)
#install.packages("robustlmm")
#library(robustlmm)
clus11.max.rlmer <- rlmer(Total_transitions ~ mycl + (1|HenID), data=allhen11.df, REML=F)
# clus11.min.rlmer <- rlmer(Total_transitions ~ (1|HenID), data=allhen11.df, REML=F)
summary(clus11.max.rlmer)
getME(clus11.max.rlmer,"w_e") # getting all robustness weights for HenID

plot(clus11.max.rlmer)
```

to get higher efficiency for the estimates will be achieved by tuning the parameter k

```
clus11.max.rlmer_tuned1 <- update(clus11.max.rlmer, rho.sigma.e=psi2propII(smoothPsi, k=2.28))
plot(clus11.max.rlmer_tuned1)
# with more than one random term:
# rsb <- list(psi2propII(smoothPsi),psi2propII(smoothPsi, k = 2.28))
# clus11.max.rlmer_tuned2 <- update(clus11.max.rlmer, rho.sigma.b=rsb)
```

```
# comparison of robustness of different model fits, including non-robust lmer
compare(clus11.max.lmer, clus11.max.rlmer, clus11.max.rlmer_tuned1,show.rho.functions = F)
```

```
#effect size of model
1-var(residuals(clus11.max.rlmer))/(var(model.response(model.frame(clus11.max.rlmer))))
```

If understood correctly, based on the weighing there does not seem to be much of a difference despite of the heteroscedasticity in terms of the robustness of model fits, so I will stick to non-robust lmer. I did not manage to extract p-values with robust lmm, somehow not implemented and not extractable as with lmer. Therefore, I want to apply a bootstrap with 1000 Simulations using lmer: by parametric bootstrapping- that is, by simulating data from the fitted model, refitting the model with subset of data, and extracting the new estimated parameters (or any other quantities of interest). Therefore, I keep the results extracted from non-robust lmer mentioned above.

POISSON for Total_Transition

Because Total Transition refers to count data ==> POISSON

```
clus11.max.lmer <- glmer((Total_transitions) ~ mycl + (1|HenID),family=poisson,
data=allhen11.df)

clus11.min.lmer <- glmer((Total_transitions) ~  (1|HenID), family=poisson, data=allhen11.df)
```

```
# Check for Overdispersion #
overdisp_fun <- function(model) {
  rdf <- df.residual(model)
  rp <- residuals(model,type="pearson")
  Pearson.chisq <- sum(rp^2)
  prat <- Pearson.chisq/rdf
  pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=F)
  c(chisq=Pearson.chisq,ratio=prat, rdf=rdf, p=pval)
}
overdisp_fun(clus11.max.lmer)
```

Data is very overdispersed, therefore: negative binomial

```
clus11.max.lmer <- glmer.nb((Total_transitions) ~ mycl + (1|HenID), data=allhen11.df)
overdisp_fun(clus11.max.lmer)
```

with this approach: ratio is close to 1, so no overdispersion left

```
clus11.max.lmer <- glmer.nb((Total_transitions) ~ mycl +  (1|HenID),  data=allhen11.df)
clus11.min.lmer <- glmer.nb((Total_transitions) ~  (1|HenID),  data=allhen11.df)
anova(clus11.max.lmer, clus11.min.lmer, test="F")
```

Modelling for Plotting

```
library(boot)
pred.df <- expand.grid (mycl= c ("1","2","3","4","5"))
estim.Mod <- function (x) {
  predict (x, newdat= pred.df, re.form= NA)}
extract.ci <- function (x) {
    out <- data.frame (numeric (0), numeric (0), numeric (0))
    for (i in 1:length (x [['t0']])) {
      out <- rbind (out, c (x [['t0']] [i],
                            boot.ci (x, index= i, type= 'perc') [['percent']] [, 4:5]))
    }
    names (out) <- c ('estim', 'lo.ci', 'up.ci')
    out
  }

# Boxplot with model estimations #
mod <-glmer.nb(Total_transitions ~ mycl +
             +(1 | HenID),
           REML= FALSE, data= allhen11.df)

mod.boot <- bootMer (mod, estim.Mod, nsim= 100, .progress= 'win')

##CI calculations #
mod.estim <- exp(extract.ci(mod.boot))
mod.estim
```

Plotting

```
library(Hmisc)
allhen11.df$mycl <- factor(allhen11.df$mycl,levels=c("4","3","2","1"))
par(mar=c(4,4,2,4))
boxplot((allhen11.df$Total_transitions)~allhen11.df$mycl
        , axes=F,las=1,ylab="number of transitions",xlab="",cex.lab=1.0, outline=F)
axis (2, seq(0, 120, 20), c ('0','20', '40', '60','80','100','120'), line= 0, las= 1,
cex.axis=1.0)
axis (1, at= c(1.0,2.0, 3.0, 4.0),
      labels = c("","","",""),
      cex.axis = 1.3)

mtext (side =1, at = 0.2, "cluster", line = 1.2, cex=1.0)
mtext (side =1, at = 1, "1", line = 1.2, cex=1.0)
mtext (side =1, at = 2, "2", line = 1.2, cex=1.0)
mtext (side =1, at = 3, "3", line = 1.2, cex=1.0)
mtext (side =1, at = 4, "4", line = 1.2, cex=1.0)

segments(x0 = 0.56, x1 = 1.44, y0 = mod.estim [4:4, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 0.56, x1 = 1.44, y0 = mod.estim [4:4, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 0.56, x1 = 1.44, y0 = mod.estim [4:4, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")

segments(x0 = 1.56, x1 = 2.44, y0 = mod.estim [3:3, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 1.56, x1 = 2.44, y0 = mod.estim [3:3, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 1.56, x1 = 2.44, y0 = mod.estim [3:3, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")

segments(x0 = 2.56, x1 = 3.44, y0 = mod.estim [2:2, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 2.56, x1 = 3.44, y0 = mod.estim [2:2, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 2.56, x1 = 3.44, y0 = mod.estim [2:2, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")

segments(x0 = 3.56, x1 = 4.44, y0 = mod.estim [1:1, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 3.56, x1 = 4.44, y0 = mod.estim [1:1, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 3.56, x1 = 4.44, y0 = mod.estim [1:1, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")
```

```
  clus11.max.lmer <- lmer(total_time_IN ~ mycl + (1|HenID), data=allhen11.df, REML=F)
  clus11_min.lmer <- lmer(total_time_IN  ~  (1|HenID), data=allhen11.df, REML=F)
  anova(clus11.max.lmer, clus11_min.lmer, test="F")
```

```
  clus11.totIN.lmer <- PBmodcomp(clus11.max.lmer,clus11_min.lmer)
  summary(clus11.totIN.lmer)
```

```
library(multcomp)
summary(glht(clus11.max.lmer, linfct = mcp(mycl="Tukey")),test=adjusted("fdr"))
```

```
# Residuenanalysen #
par (mfrow= c (2, 2))
qqnorm (resid (clus11.max.lmer))
qqnorm (ranef (clus11.max.lmer) [['HenID']] [, 1])
scatter.smooth (fitted (clus11.max.lmer), resid (clus11.max.lmer))

# Random Analyse #
for (ranEF in names (ranef (clus11.max.lmer))) {
  qqnorm (ranef (clus11.max.lmer) [[ranEF]] [, 1])
}
```

Again issue with heteroscedasticity ==> check with robust rlmer how it is weighed.

```
#install.packages("robustlmm")
#library(robustlmm)
clus11.max.rlmer <- rlmer(total_time_IN ~ mycl + (1|HenID), data=allhen11.df, REML=F)
summary(clus11.max.rlmer)
effect size of model
1-var(residuals(clus11.max.rlmer))/(var(model.response(model.frame(clus11.max.rlmer))))
```

Modelling for Plotting

```
library(boot)
pred.df <- expand.grid (mycl= c ("1","2","3","4","5"))
estim.Mod <- function (x) {
  predict (x, newdat= pred.df, re.form= NA)}
extract.ci <- function (x) {
    out <- data.frame (numeric (0), numeric (0), numeric (0))
    for (i in 1:length (x [['t0']])) {
      out <- rbind (out, c (x [['t0']] [i],
                            boot.ci (x, index= i, type= 'perc') [['percent']] [, 4:5]))
    }
    names (out) <- c ('estim', 'lo.ci', 'up.ci')
    out
  }

mod <-lmer(total_time_IN ~ mycl +
             +(1 | HenID),
           REML= FALSE, data= allhen11.df)
```

```
mod.boot <- bootMer (mod, estim.Mod, nsim= 100, .progress= 'win')

mod.estim <- (extract.ci(mod.boot))
mod.estim
```

Plotting

```
allhen11.df$mycl <- factor(allhen11.df$mycl,levels=c("4","3","2","1"))
par(mar=c(4,4,2,4))
boxplot((allhen11.df$total_time_IN)~allhen11.df$mycl
        , axes=F,las=1,ylab="duration spent indoors [min]",xlab="",cex.lab=1.0, outline=F)
axis (2, seq(0, 30240, 5040), c ('0','84', '168', '252','336','420','504'), line= 0, las= 1,
cex.axis=1.0)
axis (1, at= c(1.0,2.0, 3.0, 4.0),
      labels = c("","","",""),
      cex.axis = 1.3)

mtext (side =1, at = 0.2, "cluster", line = 1.2, cex=1.0)

mtext (side =1, at = 1, "1", line = 1.2, cex=1.0)
mtext (side =1, at = 2, "2", line = 1.2, cex=1.0)
mtext (side =1, at = 3, "3", line = 1.2, cex=1.0)
mtext (side =1, at = 4, "4", line = 1.2, cex=1.0)

segments(x0 = 0.56, x1 = 1.44, y0 = mod.estim [4:4, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 0.56, x1 = 1.44, y0 = mod.estim [4:4, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 0.56, x1 = 1.44, y0 = mod.estim [4:4, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")

segments(x0 = 1.56, x1 = 2.44, y0 = mod.estim [3:3, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 1.56, x1 = 2.44, y0 = mod.estim [3:3, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 1.56, x1 = 2.44, y0 = mod.estim [3:3, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")

segments(x0 = 2.56, x1 = 3.44, y0 = mod.estim [2:2, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 2.56, x1 = 3.44, y0 = mod.estim [2:2, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 2.56, x1 = 3.44, y0 = mod.estim [2:2, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")

segments(x0 = 3.56, x1 = 4.44, y0 = mod.estim [1:1, 'estim'], lwd = 3, lty= 1, col =
"firebrick")
segments(x0 = 3.56, x1 = 4.44, y0 = mod.estim [1:1, 'lo.ci'], lwd = 2, lty= 2, col =
"firebrick")
segments(x0 = 3.56, x1 = 4.44, y0 = mod.estim [1:1, 'up.ci'], lwd = 2, lty= 2, col =
"firebrick")
```