```
# hit ~ condition*emotionBlock

setwd('/home/allgoodguys/Documents/Studying/Lund_PhD/epistles/
005_with-Lima/analysis')
library(lme4)
require(brms)
require(shinystan)

df = read.csv ('data/data_preprocessed.csv')[,-1]
# or: df = read.csv('data_preprocessed_40sounds.csv')[,-1]

df_target = df[df$type=='target',] # targets only

## sanity check
df_plot_ctrl = aggregate(hit~condition+emotionBlock, df_target,
mean)
colnames(df_plot_ctrl)[3] = 'outcome'
ggplot (df_plot_ctrl, aes(x=emotionBlock, y=outcome*100,
fill=condition)) +
  geom_bar(stat='identity', position=position_dodge(.7)) +
  xlab("Hit") +
  ylab('%') +
  theme_bw ()

# NB: very few non-hits in some emotion-condition combinations:
myt = aggregate(hit~emotion+condition, df_target, function(x)
(length(x)-sum(x)))
hist(myt[,3])
range(myt[,3]) # 3 to 58 out of 280 possible

mod0 = glmer(hit ~ condition*emotionBlock + (1|subject)+(1|item),
family='binomial', data=df_target, nAGQ=0)
summary(mod0)
drop1(mod0, test='Chisq')

# with intensity and pretest accuracy as covariates
mod1 = glmer(hit ~ condition*emotionBlock + subjIntensity +
pretestAccuracy + (1|subject)+(1|item), family='binomial',
data=df_target, nAGQ=0)
summary(mod1)
drop1(mod1, test='Chisq')

mod = brm(hit ~ condition*emotionBlock + (1|subject)+(1|item), data
= df_target, warmup=100, iter=500, chains=4, cores=4,
family='bernoulli', ranef=F, prior=set_prior("horseshoe(1)"),
control = list(adapt_delta = 0.95)) # set_prior for all beta-coef at
once (shrinkage): see https://cran.r-project.org/web/packages/brms/
vignettes/brms.pdf
# saveRDS(mod, 'models/mod_hit~cond*em_shrinkage.RDS') # or
saveRDS(mod, 'models/mod_hit~cond*em_shrinkage_40sounds.RDS')
# mod = readRDS('models/mod_hit~cond*em_shrinkage.RDS') # or mod =
readRDS('models/mod_hit~cond*em_shrinkage_40sounds.RDS')
# stancode(mod)
# plot(mod)
summary(mod)
# launch_shiny(mod)
# pairs(mod, pars = c("b_Intercept", "b_emotionBlockFear", "lp__"),
las = 1)

# pred_df = expand.grid(condition=levels(df_target$condition),
emotionBlock=levels(df_target$emotionBlock))
# myfit = fitted (mod, newdata=pred_df, re_formula=NA,
scale='response', robust=T) # should be identical to manual below

coda = posterior_samples(mod)
colnames(coda)

a = data.frame(achievement_delib=coda[,1],
               achievement_fast=coda[,1]+coda[,2],
               achievement_load6=coda[,1]+coda[,3],
               achievement_load8=coda[,1]+coda[,4],

               amusement_delib=coda[,1]+coda[,5],
               amusement_fast=coda[,1]+coda[,2]+coda[,5]+coda[,12],
               amusement_load6=coda[,1]+coda[,3]+coda[,5]+coda[,13],
               amusement_load8=coda[,1]+coda[,4]+coda[,5]+coda[,14],

               anger_delib=coda[,1]+coda[,6],
               anger_fast=coda[,1]+coda[,2]+coda[,6]+coda[,15],
               anger_load6=coda[,1]+coda[,3]+coda[,6]+coda[,16],
               anger_load8=coda[,1]+coda[,4]+coda[,6]+coda[,17],

               disgust_delib=coda[,1]+coda[,7],
               disgust_fast=coda[,1]+coda[,2]+coda[,7]+coda[,18],
               disgust_load6=coda[,1]+coda[,3]+coda[,7]+coda[,19],
               disgust_load8=coda[,1]+coda[,4]+coda[,7]+coda[,20],

               fear_delib=coda[,1]+coda[,8],
               fear_fast=coda[,1]+coda[,2]+coda[,8]+coda[,21],
               fear_load6=coda[,1]+coda[,3]+coda[,8]+coda[,22],
               fear_load8=coda[,1]+coda[,4]+coda[,8]+coda[,23],

               pleasure_delib=coda[,1]+coda[,9],
               pleasure_fast=coda[,1]+coda[,2]+coda[,9]+coda[,24],
               pleasure_load6=coda[,1]+coda[,3]+coda[,9]+coda[,25],
               pleasure_load8=coda[,1]+coda[,4]+coda[,9]+coda[,26],

               relief_delib=coda[,1]+coda[,10],
               relief_fast=coda[,1]+coda[,2]+coda[,10]+coda[,27],
               relief_load6=coda[,1]+coda[,3]+coda[,10]+coda[,28],
               relief_load8=coda[,1]+coda[,4]+coda[,10]+coda[,29],

               sadness_delib=coda[,1]+coda[,11],
               sadness_fast=coda[,1]+coda[,2]+coda[,11]+coda[,30],
               sadness_load6=coda[,1]+coda[,3]+coda[,11]+coda[,31],
               sadness_load8=coda[,1]+coda[,4]+coda[,11]+coda[,32])
a = apply (a, 2, function(x)1/(1+exp(-x)))
```

```
df_plot = data.frame (emotion =
rep(c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure',
'Relief','Sadness'),each=4),
condition=rep(c('Deliberated','Fast','Load 1','Load 2'),8), lwr=NA,
fit=NA, upr=NA)
df_plot[,3:5] = t ( apply (a, 2, function(x)quantile(x,probs=c(.
025,.5,.975))*100) )
# df_plot$fit==myfit[,1]*100 # should be identical

# add overall:
df_plot_overall = read.csv('output/hit~cond.csv')[,-1]
# or df_plot_overall = read.csv ('output/hit~cond_40sounds.csv')
[,-1]
df_plot_overall$emotion = 'Overall'
df_plot_overall = df_plot_overall[,c(1,5,2,3,4)]
df_plot = rbind (df_plot, df_plot_overall)

# re-arrange the order of emotions:
df_plot$emotion = factor(df_plot$emotion,
levels=c('Achievement','Amusement','Pleasure','Relief','Anger','Disg
ust','Fear','Sadness','Overall'))

# from accuracy to %errors:
# df_plot[,3:5] = 100 - df_plot[,3:5]
library(scales)
ggplot (df_plot, aes(x=emotion, y=fit, fill=condition)) +
  geom_bar(stat='identity', position=position_dodge(.7)) +
  geom_errorbar(aes(ymin=lwr, ymax=upr), position=position_dodge(.
7)) +
  geom_hline(yintercept=50, linetype=2) +
  geom_vline(xintercept=4.5, linetype=2) +
  geom_vline(xintercept=8.5, linetype=2) +
  scale_fill_discrete(name='') +
  xlab("") +
  ylab('Hit rate, %') +
  scale_y_continuous(limits=c(50,100),oob = rescale_none) +
  theme_bw () +
  theme(legend.position="top")

# ggsave ('pix/hit~cond*em.jpg', width=10, height=4, units='cm',
dpi=300, scale=2)
# ggsave ('pix/hit~cond*em_40sounds.jpg', width=10, height=4,
units='cm', dpi=300, scale=2)

## contrasts
mcmc0 = read.csv('output/mcmc_hit~cond.csv')[,-1] # or mcmc0 =
read.csv('mcmc_hit~cond_40sounds.csv')[,-1]
colnames(mcmc0)
# Is "deliberated" overall better than the other three conditions?
colnames(a)
delib_vs_rest = data.frame (ems =
c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure','Rel
ief','Sadness'), fit=NA, lwr=NA, upr=NA, stringsAsFactors=F)
```

```
for (em in 1:nrow(delib_vs_rest)){
  r = 4*em-3
  delib_vs_rest[em,2:4] = quantile( a[,r]-(a[,r+1]+a[,r+2]+a[,r+3])/
3, probs=c(.5,.025,.975)) * 100
}
delib_vs_rest = rbind (delib_vs_rest,
c('Overall',quantile( mcmc0[,'delib']-(mcmc0[,'load1']
+mcmc0[,'load2']+mcmc0[,'fast'])/3, probs=c(.5,.025,.975))*100)) #
overall a modest effect

delib_vs_fast = data.frame(ems =
c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure','Rel
ief','Sadness'), fit=NA, lwr=NA, upr=NA, stringsAsFactors=F)
for (em in 1:nrow(delib_vs_fast)){
  r = 4*em-3
  delib_vs_fast[em,2:4] = quantile( a[,r]-a[,r+1], probs=c(.
5,.025,.975)) * 100
}
delib_vs_fast = rbind(delib_vs_fast, c('Overall',
quantile(mcmc0[,'delib']-mcmc0[,'fast'], probs=c(.
5,.025,.975))*100))

delib_vs_load1 = data.frame(ems =
c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure','Rel
ief','Sadness'), fit=NA, lwr=NA, upr=NA, stringsAsFactors=F)
for (em in 1:nrow(delib_vs_load1)){
  r = 4*em-3
  delib_vs_load1[em,2:4] = quantile( a[,r]-a[,r+2], probs=c(.
5,.025,.975)) * 100
}
delib_vs_load1 = rbind(delib_vs_load1, c('Overall',
quantile(mcmc0[,'delib']-mcmc0[,'load1'], probs=c(.
5,.025,.975))*100))

delib_vs_load2 = data.frame(ems =
c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure','Rel
ief','Sadness'), fit=NA, lwr=NA, upr=NA, stringsAsFactors=F)
for (em in 1:nrow(delib_vs_load2)){
  r = 4*em-3
  delib_vs_load2[em,2:4] = quantile( a[,r]-a[,r+3], probs=c(.
5,.025,.975)) * 100
}
delib_vs_load2 = rbind(delib_vs_load2, c('Overall',
quantile(mcmc0[,'delib']-mcmc0[,'load2'], probs=c(.
5,.025,.975))*100))


# Is load6 better than load8?
load6_vs_8 = data.frame (ems =
c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure','Rel
ief','Sadness'), fit=NA, lwr=NA, upr=NA, stringsAsFactors=F)
for (em in 1:nrow(load6_vs_8)){
  r = 4*em
  load6_vs_8[em,2:4] = quantile( a[,r-1]-a[,r], probs=c(.
5,.025,.975)) * 100
}
load6_vs_8 = rbind (load6_vs_8,
c('Overall',quantile( mcmc0[,'load1']-mcmc0[,'load2'], probs=c(.
5,.025,.975))*100)) # overall a small effect


# Is deliberated better than fast?
delib_vs_fast = data.frame (ems =
c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure','Rel
ief','Sadness'), fit=NA, lwr=NA, upr=NA, stringsAsFactors=F)
for (em in 1:nrow(delib_vs_fast)){
  r = 4*em-3
  delib_vs_fast[em,2:4] = quantile( a[,r]-a[,r+1], probs=c(.
5,.025,.975)) * 100
}
delib_vs_fast = rbind (delib_vs_fast,
c('Overall',quantile( mcmc0[,'delib']-mcmc0[,'fast'], probs=c(.
5,.025,.975))*100))  # overall a small effect


# Is fast better than mean(load6,load8)?
fast_vs_load = data.frame (ems =
c('Achievement','Amusement','Anger','Disgust','Fear','Pleasure','Rel
ief','Sadness'), fit=NA, lwr=NA, upr=NA, stringsAsFactors=F)
for (em in 1:nrow(fast_vs_load)){
  r = 4*em-2
  fast_vs_load[em,2:4] = quantile( a[,r]-(a[,r+1]+a[,r+2])/2,
probs=c(.5,.025,.975)) * 100
}
fast_vs_load = rbind (fast_vs_load,
c('Overall',quantile( mcmc0[,'fast']-(mcmc0[,'load1']
+mcmc0[,'load2'])/2, probs=c(.5,.025,.975))*100))  # overall no
effect


df_plot = rbind (delib_vs_rest, delib_vs_fast, fast_vs_load,
load6_vs_8)
df_plot$contrast = rep(c('Deliberated vs.\nmean(fast, low load, high
load)', 'Deliberated vs. fast', 'Fast vs.\nmean(low load, high
load)', 'Low load vs. high load'), each=9)
df_plot$ems = factor(df_plot$ems,
levels=c('Achievement','Amusement','Anger','Disgust','Fear','Pleasur
e','Relief','Sadness','Overall'))
df_plot[,2:4] = apply (df_plot[,2:4], 2, function(x)as.numeric(x))

# re-arrange the order of emotions:
df_plot$ems = factor(df_plot$ems,
levels=c('Achievement','Amusement','Pleasure','Relief','Anger','Disg
ust','Fear','Sadness','Overall'))

ggplot (df_plot, aes(x=contrast, y=fit, color=ems, shape=ems)) +
  geom_point(position=position_dodge(.8), size=4) +
  geom_errorbar(aes(ymin=lwr, ymax=upr), position=position_dodge(.
8), width=.5, show.legend=F) + # NB: show.legend=F for error bars
removes horizontal slashes in the legend!
  geom_hline(yintercept=0, linetype=2) +
  geom_vline(xintercept=1.5, linetype=2) +
  geom_vline(xintercept=2.5, linetype=2) +
  geom_vline(xintercept=3.5, linetype=2) +
  scale_color_manual(name='',
values=c(paste0('gray',floor(seq(50,0,length.out=9))))) +
  scale_shape_manual(name='', values=c(0,1,2,5,7,11,12,13,19)) +
  xlab("") +
  ylab('Difference in hit rates, %') +
  theme_bw () +
  theme(legend.position="top",
legend.key=element_rect(fill='white'), panel.grid=element_blank())

# ggsave ('pix/hit~cond*em_contrasts_points.jpg', width=12,
height=4, units='cm', dpi=300, scale=2)
# ggsave ('pix/hit~cond*em_40sounds_contrasts_points.jpg', width=12,
height=4, units='cm', dpi=300, scale=2)




### Non-Bayesian testing of the significance of delib vs all other 3
conditions for each emotion separately (8 comparisons)
alpha = .05 / 8  # Bonferroni  # set alpha = .05 to see all the
output
df$delib = df$condition == 'Deliberated'
for (em in levels(df$emotionBlock)) {
  temp = df[df$emotionBlock == em, ]
  mod = glmer(hit ~ delib + (1|subject)+(1|item), family='binomial',
data=temp, nAGQ=0)
  s = summary(mod)$coef[2, 3:4, drop = FALSE]
  print(em)
  print(s[s[, 2] < alpha, ])
}

### Same but with dur + int + acc
alpha = .05 / 8  # Bonferroni  # set alpha = .05 to see all the
output
df$delib = df$condition == 'Deliberated'
for (em in levels(df$emotionBlock)) {
  temp = df[df$emotionBlock == em, ]
  mod = glmer(hit ~ delib + duration + subjIntensity +
pretestAccuracy + (1|subject)+(1|item), family='binomial',
data=temp, nAGQ=0)
  s = summary(mod)$coef[2, 3:4, drop = FALSE]
  print(em)
  print(s[s[, 2] < alpha, ])
}


### Non-Bayesian testing of the significance of delib vs the other 3
conditions for each emotion separately (3 x 8 = 24 comparisons)
```

```
alpha = .05 / 24  # Bonferroni  # set alpha = .05 to see all the
output
for (em in levels(df$emotionBlock)) {
  temp = df[df$emotionBlock == em, ]
  mod = glmer(hit ~ condition + (1|subject)+(1|item),
family='binomial', data=temp, nAGQ=0)
  s = summary(mod)$coef[2:4, 3:4]
  print(em)
  print(s[s[, 2] < alpha, , drop = FALSE])
}


### Non-Bayesian testing of the significance of low vs high load for
each emotion separately (8 comparisons)
alpha = .05 / 8  # Bonferroni
df_load = droplevels(df[df$condition == 'Load 1' | df$condition ==
'Load 2', ])
for (em in levels(df$emotionBlock)) {
  temp = df_load[df_load$emotionBlock == em, ]
  mod = glmer(hit ~ condition + (1|subject)+(1|item),
family='binomial', data=temp, nAGQ=0)
  s = summary(mod)$coef[2, 3:4]
  print(em)
  print(s[s[2] < alpha])
}

### Non-Bayesian testing of the significance of fast vs low/high
load for each emotion separately (8 comparisons)
alpha = .05 / 8  # Bonferroni
df1 = droplevels(df[df$condition %in% c('Fast', 'Load 1' , 'Load
2'), ])
df1$fast = df1$condition == 'Fast'
for (em in levels(df1$emotionBlock)) {
  temp = df1[df1$emotionBlock == em, ]
  mod = glmer(hit ~ fast + (1|subject)+(1|item), family='binomial',
data=temp, nAGQ=0)
  s = summary(mod)$coef[2, 3:4]
  print(em)
  print(s[s[2] < alpha])
}
# overall:
mod = glmer(hit ~ fast + (1|subject)+(1|item), family='binomial',
data=df1, nAGQ=0)
summary(mod)
drop1(mod, test = "Chisq")
```