```
# analysis of accuracy as a function of decision time

setwd('/home/allgoodguys/Documents/Studying/Lund_PhD/epistles/
005_with-Lima/analysis')

library (ggplot2)
library (brms)


cutoff_low = 250
# cutoff_high is 3*SD


df = read.csv ('data/data_preprocessed.csv')[,-1]
x_range = range(df$time2) # before we separate conditions

df$log_time2 = log(df$time2)
df = droplevels(df[df$condition!='Deliberated',])
# or df = droplevels(df[df$condition=='Deliberated',])

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


## exclude everything beyond ±3 SD relative to the mean for each
participant
df_noOutliers = df
excl_lwr = excl_upr = 0
for (p in 1:length(unique(df$subject))){
  temp = df[df$subject==unique(df$subject)[p],]
  mean_per_subject = mean(temp$time2, na.rm=T)
  sd_per_subject = sd(temp$time2, na.rm=T)
  lwr = mean_per_subject - 3*sd_per_subject
  upr = mean_per_subject + 3*sd_per_subject
  idx_lwr = which(df_noOutliers$subject==unique(df$subject)[p] &
df_noOutliers$time2<lwr)
  idx_upr = which(df_noOutliers$subject==unique(df$subject)[p] &
df_noOutliers$time2>upr)
  if (length(idx_lwr)>0) {df_noOutliers = df_noOutliers[-idx_lwr,]}
  if (length(idx_upr)>0) {df_noOutliers = df_noOutliers[-idx_upr,]}
  excl_lwr = excl_lwr + length(idx_lwr)
  excl_upr = excl_upr + length(idx_upr)
}

range(df$time2)
range(df_noOutliers$time2)

df_noOutliers = df_noOutliers[df_noOutliers$time2>cutoff_low,]
# quantile(df_noOutliers$time2, probs=c(.5,.025,.975))

# actual HIT rates
bin_ms = 25
out = data.frame (thresholds = sort(unique(df$time2)), accuracy=NA)
for (i in 1:nrow(out)){
  out$accuracy[i] = mean(df$correct[ (df$time2 >=
(out$thresholds[i]-bin_ms)) & (df$time2 <= (out$thresholds[i]
+bin_ms)) ])*100 # average over bins of ... ms
}

# density plot of latencies
d = density(df$time2)
d = d$y/max(d$y)*100
dens =
data.frame(latency=seq(out$thresholds[1],out$thresholds[nrow(out)],l
ength.out=length(d)), d=d)

# quantiles of latencies
quantiles = quantile(df$time2, probs=c(.
0005,.005,.01,.025,.25,.5,.75,.975,.99,.995,.9995))
dens$quant = factor(findInterval(dens$latency,quantiles))
qq = data.frame('quantile' = names(quantiles),
                'latency' = as.numeric(quantiles)) #
as.numeric(c(quantiles[1]/2, quantiles[1:(length(quantiles)-1)]
+diff(quantiles/2), (max(dens$latency)-
quantiles[length(quantiles)])/2+quantiles[length(quantiles)]))
qq$density = dens$d[sapply(1:nrow(qq),
function(x)which.min(abs(dens$latency-qq$latency[x])))]

r = c( floor(log2(min(df$time2))), ceiling(log2(max(df$time2))) )
mybreaks = round(2^(seq(r[1],r[2],by=.5)))



prop_low = paste0 (round((nrow(df)-nrow(df_noOutliers)-excl_upr) /
nrow(df) * 100, 2), '%')
prop_high = paste0 (round(excl_upr / nrow(df) * 100, 1), '%')


## Add a GAMM for overall accuracy in all conditions after removing
the outliers
# mod_brm = brm (correct~s(log_time2) + (1|subject)+(1|item),
data=df_noOutliers, warmup=500, iter=1000, chains=4, cores=4,
family='bernoulli', ranef=F, prior=set_prior("normal(0,1)"),
control=list(adapt_delta=0.95))

# saveRDS(mod_brm,
'mod_correct~time_noNA_gamm_allConditions_±3SD.RDS')
# mod_brm =
readRDS('mod_correct~time_noNA_gamm_allConditions_±3SD.RDS')

# or, for 3 conditions:
# saveRDS(mod_brm,
'mod_correct~time_noNA_gamm_3Conditions_±3SD.RDS')
# mod_brm =
readRDS('mod_correct~time_noNA_gamm_3Conditions_±3SD.RDS')

# or, for delib:
# saveRDS(mod_brm, 'mod_correct~time_noNA_gamm_delib_±3SD.RDS')
# mod_brm = readRDS('mod_correct~time_noNA_gamm_delib_±3SD.RDS')

# plot(mod_brm)
# summary(mod_brm)

# ps: quickly checking for diff. k's
# library (mgcv)
# mod_gam1 = gam (correct~s(log_time2), data=df_noOutliers) #
defaults to 10
# mod_gam2 = gam (correct~s(log_time2, k=5), data=df_noOutliers)
# mod_gam3 = gam (correct~s(log_time2, k=10), data=df_noOutliers)
# mod_gam4 = gam (correct~s(log_time2, k=20), data=df_noOutliers)
# mod_gam5 = gam (correct~s(log_time2, k=30), data=df_noOutliers)
# plot (mod_gam1)
# anova (mod_gam1, mod_gam2, mod_gam3, mod_gam4, mod_gam5,
test='Chisq')
# AIC (mod_gam1, mod_gam2, mod_gam3, mod_gam4, mod_gam5)

newdata = data.frame
(log_time2=seq(min(df_noOutliers$log_time2),max(df_noOutliers$log_ti
me2),length.out=100))
myfit = as.data.frame (fitted (mod_brm, newdata=newdata,
re_formula=NA, scale='response', robust=T))
colnames(myfit) = c('fit','se','lwr','upr')
myfit[,c('fit','lwr','upr')] = myfit[,c('fit','lwr','upr')] * 100
myfit$time = exp(newdata$log_time2)
# min(myfit$time[myfit$lwr>50]) # when is accuracy first above
chance level?
# myfit$fit[myfit$time >= 500][1] # accuracy at 500 ms

# plot predicted + observed
r = c( floor(log2(x_range[1])), ceiling(log2(x_range[2])) )
mybreaks = round(2^(seq(r[1],r[2],by=.5)))


ggplot(out, aes(x=thresholds, y=accuracy)) +
  # original data aggregated in bins + density plot with quantiles
marked
  geom_point(alpha=.5) +
  geom_ribbon(data=dens, aes(x=latency, ymin=0, ymax=d, fill=quant),
alpha=.5, inherit.aes=F) +
  scale_fill_manual(values=c(paste0('gray',floor(seq(90, 50,
length.out=length(unique(dens$quant))))))) +
  geom_text(data=qq, aes(x=latency, y=density, label=quantile),
size=3, inherit.aes=F) +

  # GAMM line and 95% CI
  geom_line (data=myfit, aes(x=time, y=fit), color='black',
linetype="solid",  size=1, alpha=1, inherit.aes=F) +
  geom_ribbon(data=myfit, aes(x=time, ymin=lwr,ymax=upr),
alpha=0.25, inherit.aes=F) +

  # cutoff points
  geom_vline(xintercept=cutoff_low, color="gray50",
```

```r
linetype="dashed", size=1) +
  geom_text(aes(x=160, y=40, label=paste0('<3 SD or\n<',cutoff_low,'
ms')), size=5, color='gray50') + # 140 for 3cond, 250 for delib
  geom_text(aes(x=160, y=20, label=prop_low), size=5,
color='gray50') +
  # geom_vline(xintercept=cutoff_high, color="blue",
linetype="dashed", size=1) +
  geom_text(aes(x=9000, y=40, label='>3 SD'), size=5,
color='gray50') + # 5800 for 3 cond, 9000 for delib
  geom_text(aes(x=9000, y=20, label=prop_high), size=5,
color='gray50') +

  # frills
  scale_x_continuous(trans='log2', breaks=mybreaks,
limits=c(128,16384)) + # limits for delib only
  ylab('Accuracy (hits and false alarms, %') +
  xlab('Latencies (ms)') +
  theme_bw() +
  theme(legend.position="none", panel.grid=element_blank())

# ggsave ('pix/time_excludingOutliers.jpg', width=17, height=8,
units='cm', dpi=300, scale=1.5)
# ggsave ('pix/time_excludingOutliers_3conds.jpg', width=17,
height=8, units='cm', dpi=300, scale=1.5)
# ggsave ('pix/time_excludingOutliers_delib.jpg', width=17,
height=8, units='cm', dpi=300, scale=1.5)


# at what latencies does accuracy climb above chance level?
myfit$time[min(which(myfit$lwr>50))] # 300 ms for 3 conds
# myfit$time[min(which(myfit$fit>50))]


### per emotion. NB: looks like this is w/o "interaction" - find out
how to get that! te(log_time2)+te(emotinBlock)?
# sanity check
bin_ms = 25
out = data.frame (thresholds =
seq(min(df$time2),max(df$time2),by=bin_ms), ach=NA, amu=NA, ang=NA,
disg=NA, fear=NA, plsr=NA, rel=NA, sad=NA)
for (i in 1:(nrow(out)-1)){
  temp = df[ (df$time2 >= (out$thresholds[i])) & (df$time2 <=
(out$thresholds[i+1])), ]
  out[i,2:9] = tapply(X=temp$correct, INDEX=temp$emotionBlock,
FUN=mean) # average over bins of ... ms
}
library(reshape2)
out_melt = melt(out, id='thresholds')
colnames(out_melt)=c('time','emotion','correct')
out_melt$correct = out_melt$correct * 100
r = c( floor(log2(min(out_melt$time))),
ceiling(log2(max(out_melt$time))) )
mybreaks = round(2^(seq(r[1],r[2],by=.5)))
```

```
ggplot(out_melt, aes(x=time, y=correct, group=emotion,
color=emotion, shape=emotion)) +
  geom_point(size=3) +
  scale_shape_manual(values=c(1,2,4,8,9,11,13,18)) +
  scale_x_continuous(trans='log2', breaks=mybreaks) +
  scale_y_continuous(limits=c(0,105)) +
  ylab('Accuracy, %') +
  xlab('Latency, ms') +
  theme_bw() +
  theme(legend.position="top", legend.title=element_blank())
# thus: not enough points to talk about emotions separately under
~500 ms, so we won't know if they differ in terms of fast
recognition!!!

mod_brm1 = brm (correct~s(log_time2,emotionBlock,bs="fs") + (1|
subject)+(1|item), data=df_noOutliers, warmup=50, iter=100,
chains=4, cores=4, family='bernoulli', ranef=F,
prior=set_prior("normal(0,10)"), control=list(adapt_delta=0.95))
# saveRDS(mod_brm1, 'mod_cor~time*em_GAMM_3conds.RDS') #
saveRDS(mod_brm1, 'mod_cor~time*em_GAMM-fs_3conds.RDS')
# mod_brm1 = readRDS('mod_cor~time*em_GAMM_3conds.RDS')

plot(mod_brm1)
summary(mod_brm1)

n_points = 100

newdata = data.frame
(log_time2=rep(seq(min(df_noOutliers$log_time2),max(df_noOutliers$lo
g_time2),length.out=n_points),8),

emotionBlock=rep(levels(df_noOutliers$emotionBlock),each=n_points))
myfit = as.data.frame (fitted (mod_brm1, newdata=newdata,
re_formula=NA, scale='response', robust=T))
colnames(myfit) = c('fit','se','lwr','upr')
myfit[,c('fit','lwr','upr')] = myfit[,c('fit','lwr','upr')] * 100
myfit$time = exp(newdata$log_time2)
myfit$emotionBlock = newdata$emotionBlock

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

# plot predicted + observed
r = c( floor(log2(min(df_noOutliers$time2))),
ceiling(log2(max(df_noOutliers$time2))) )
mybreaks = round(2^(seq(r[1],r[2],by=.5)))
ggplot(myfit, aes(x=time, y=fit, group=emotionBlock,
color=emotionBlock, shape=emotionBlock, fill=emotionBlock)) +
  geom_ribbon(aes(ymin=lwr,ymax=upr),alpha=0.05, linetype=0) +
  geom_point(size=1) +
  scale_shape_manual(values=c(1,2,4,8,9,11,13,18)) +
  scale_x_continuous(trans='log2', breaks=mybreaks) +
  scale_y_continuous(limits=c(0,105)) +
  ylab('Accuracy, %') +
  xlab('Latency, ms') +
  theme_bw() +
  theme(legend.position="top", legend.title=element_blank())

# ggsave ('pix/cor~time*em_3cond_fs.jpg', width=16, height=10,
units='cm', dpi=300, scale=1)
```