```
# the effect of digits recalled on hit rates and of condition on
digits recalled

setwd('/home/allgoodguys/Documents/Studying/Lund_PhD/epistles/
005_with-Lima/analysis')
library(lme4)
require(brms)
require(shinystan)
library (ggplot2)

df = read.csv ('data/data_preprocessed.csv')[,-1] # df = read.csv
('data_preprocessed_48sounds.csv')[,-1]
df = droplevels (df[!is.na(df$memTask),])
table(df$condition) # only load 1 and load 2 conditions
df$log_time = log(df$time)

# hist(df$memTask)
# table(df$memTask, df$condition)
df$memTask_max = ifelse(df$condition=='Load 1', 6, 8)
df$memTask_prop = df$memTask/df$memTask_max
# hist(df$memTask_prop)

aggregate (memTask~condition, df, mean)
aggregate (memTask_prop~condition, df, mean)

# memory under load 1 vs load 2
mod0 = glmer(cbind(memTask, (memTask_max-memTask)) ~ condition + (1|
subject)+(1|item), family='binomial', data=df, nAGQ=0)
summary(mod0)
drop1(mod0, test = 'Chisq')

### Does performance on the memory task predict accuracy of emotion
recognition?
df = df[df$type=='target',] # targets only
plot (aggregate (correct~memTask_prop, df, mean)) # very weakly, if
at all

mod0 = glmer(correct ~ memTask_prop + (1|subject)+(1|item),
family='binomial', data=df, nAGQ=0)
summary(mod0)
drop1(mod0, test='Chisq') # only for hits

# same for load 1 and load 2?
mod1 = glmer(correct ~ memTask_prop*condition + (1|subject)+(1|
item), family='binomial', data=df, nAGQ=0)
summary(mod1)
drop1(mod1, test='Chisq') # no interaction, main effect only of
condition

# response latencies?
mod_speed1 = glmer(log_time ~ memTask_prop*condition + (1|subject)
+(1|item), family='gaussian', data=df)
summary(mod_speed1)
drop1(mod_speed1, test='Chisq') # no interaction
```

```
mod_speed2 = glmer(log_time ~ memTask_prop*emotionBlock + (1|
subject)+(1|item), family='gaussian', data=df)
summary(mod_speed2)
drop1(mod_speed2, test='Chisq') # no interaction

mod_speed3 = glmer(log_time ~ memTask_prop + (1|subject)+(1|item),
family='gaussian', data=df)
summary(mod_speed3)
drop1(mod_speed3, test='Chisq') # yes: a negative effect of
memTask_prop on log_time

mod_speed4 = glmer(log_time ~ memTask + (1|subject)+(1|item),
family='gaussian', data=df)
summary(mod_speed4)
drop1(mod_speed4, test='Chisq')

plot(aggregate (log_time~memTask, df, mean), type='b')
plot(aggregate (log_time~memTask_prop, df, mean), type='b')

plot(aggregate (correct~memTask, df, mean), type='b')
plot(aggregate (correct~memTask_prop, df, mean), type='b')

## emotion?
mod2 = glmer(correct ~ memTask_prop*emotionBlock + (1|subject)+(1|
item), family='binomial', data=df, nAGQ=0)
summary(mod2)
drop1(mod2, test='Chisq', scope =
~memTask_prop+emotionBlock+memTask_prop:emotionBlock) # yes

mod_brm_em = brm(correct ~ memTask_prop*emotionBlock + (1|subject)
+(1|item), family='bernoulli', data=df, warmup=100, iter=500,
chains=4, cores=4, prior=set_prior("normal(0,10)"))
# saveRDS(mod_brm_em, 'models/mod_hit~mem*em.RDS')
# mod_brm_em = readRDS('models/mod_hit~mem*em.RDS')
summary(mod_brm_em)
pred_df = expand.grid(memTask_prop=seq(0,1,by=.01),
emotionBlock=levels(df$emotionBlock))
pred_df = cbind( pred_df, fitted (mod_brm_em, newdata=pred_df,
re_formula=NA, scale='response', robust=T) )
colnames(pred_df)[3:6] = c('fit','se','lwr','upr')

library(ggplot2)
ggplot (pred_df, aes(x=memTask_prop, y=fit, color=emotionBlock,
linetype=emotionBlock, fill=emotionBlock)) +
  geom_line() +
  geom_ribbon(aes(ymin=lwr, ymax=upr), linetype='blank', alpha=0.2)

# betas per emotion
coda = posterior_samples(mod_brm_em)[,1:18]
colnames(coda)
betas = data.frame(
  achievement = coda[,2],
  amusement = coda[,2] + coda[,10],
```

```
  anger = coda[,2] + coda[,11],
  disgust = coda[,2] + coda[,12],
  fear = coda[,2] + coda[,13],
  pleasure = coda[,2] + coda[,14],
  relief = coda[,2] + coda[,15],
  sadness = coda[,2] + coda[,16]
)
t(apply (betas, 2, function(x)quantile(x, probs=c(.5, .025, .975))))
# positive for achievement and pleasure, negative for fear


### does valence have anything to do with the effect of memTask?
df_val = df[!is.na(df$subjValence),] # only targets left!!!
mod3 = glmer(correct ~ memTask_prop*subjValence + (1|subject)+(1|
item), family='binomial', data=df_val, nAGQ=0)
summary(mod3)
drop1(mod3, test='Chisq') # marginal positive interaction with
subjValence

mod_brm_val = brm(correct ~ memTask_prop*subjValence + (1|subject)
+(1|item), family='bernoulli', data=df_val, warmup=100, iter=500,
chains=4, cores=4, prior=set_prior("normal(0,10)"))
# saveRDS(mod_brm_val, 'models/mod_hit~mem*val.RDS')
# mod_brm_val = readRDS('models/mod_hit~mem*val.RDS')

summary(mod_brm_val)

memTask_prop_seq = seq(min(df_val$memTask_prop),
max(df_val$memTask_prop), length.out=100)
subjValence_seq =
seq(min(df_val$subjValence),max(df_val$subjValence), length.out=100)
newdata = expand.grid (memTask_prop = memTask_prop_seq, subjValence
= subjValence_seq)
f = cbind (newdata, fitted (mod_brm_val, newdata=newdata, re=NA))
colnames(f)[3:6] = c('fit','se','lwr','upr')

output = matrix(ncol=100,nrow=100)
for (i in 1:100) {
  for (j in 1:100) {
    output [i,j] = f$fit[f$memTask_prop==memTask_prop_seq[i] &
f$subjValence==subjValence_seq[j]]
  }
}

z = output
nrz <- nrow(z)   # some stuff to get a color gradient in persp - see
help for persp (base)
ncz <- ncol(z)
jet.colors <- colorRampPalette( c("lightblue", "yellow") )
# Generate the desired number of colors from this palette
nbcol <- 50
color <- jet.colors(nbcol)
# Compute the z-value at the facet centres
zfacet <- z[-1, -1] + z[-1, -ncz] + z[-nrz, -1] + z[-nrz, -ncz]
```

```
# Recode facet z-values into color indices
facetcol <- cut(zfacet, nbcol)
p = persp (memTask_prop_seq, subjValence_seq, output*100, theta=-30,
phi=30, expand=1, col = color[facetcol], zlab='Accuracy, %',
xlab='memTask_prop', ylab='subjValence', cex.lab=1.4, d=1,
ticktype='detailed', zlim=c(0,100))

# how to show CI with perspective plots???
output_lwr = matrix(ncol=100,nrow=100)
for (i in 1:100) {
  for (j in 1:100) {
    output_lwr [i,j] = f$lwr[f$memTask_prop==memTask_prop_seq[i] &
f$subjValence==subjValence_seq[j]]
  }
}
output_upr = matrix(ncol=100,nrow=100)
for (i in 1:100) {
  for (j in 1:100) {
    output_upr [i,j] = f$upr[f$memTask_prop==memTask_prop_seq[i] &
f$subjValence==subjValence_seq[j]]
  }
}

library(rgl)
persp3d(memTask_prop_seq, subjValence_seq, output*100, col='yellow',
zlim=c(50,100), alpha=.75, zlab='Hit rate, %', xlab='Proportion of
digits recalled', ylab='Valence of sound', ticktype='detailed')
persp3d(memTask_prop_seq, subjValence_seq, output_lwr*100, add=T,
col='blue', alpha=.25)
persp3d(memTask_prop_seq, subjValence_seq, output_upr*100, add=T,
col='red', alpha=.25)



##  Is there an effect of Emotion Category on the performance on the
memory task?
aggregate (memTask_prop~emotionBlock, df, mean)

mod0 = glmer(cbind(memTask, (memTask_max-memTask)) ~
emotionBlock*condition + (1|subject)+(1|item), family='binomial',
data=df, nAGQ=0)
summary(mod0)
drop1(mod0, test='Chisq') # yes: L = 80.6, df = 7, p < .001

mod_load1 = glmer(cbind(memTask, (memTask_max-memTask)) ~
emotionBlock + (1|subject)+(1|item), family='binomial',
data=df[df$condition=='Load 1',], nAGQ=0)
summary(mod_load1)
drop1(mod_load1, test='Chisq')

mod_load2 = glmer(cbind(memTask, (memTask_max-memTask)) ~
emotionBlock + (1|subject)+(1|item), family='binomial', data=df,
nAGQ=0)
summary(mod_load2)
```

```
drop1(mod_load2, test='Chisq')

# mod_brm = brm( memTask|trials(memTask_max) ~
emotionBlock*condition + (1|subject)+(1|item), family='binomial',
data=df, warmup=500, iter=1000, chains=4, cores=4,
prior=set_prior("normal(0,10)"))
# saveRDS(mod_brm, 'models/mod_mem~cond*em.RDS')
# mod_brm = readRDS('models/mod_mem~cond*em.RDS')

summary(mod_brm)
# coda = posterior_samples(mod_brm)
# colnames(coda)

df_plot = expand.grid(emotionBlock=levels(df$emotionBlock),
condition=levels(df$condition))
df_plot$memTask_max = ifelse(df_plot$condition=="Load 1", 6, 8)
df_plot = cbind (df_plot, fitted (mod_brm, newdata=df_plot,
re_formula = NA))
colnames(df_plot)[(ncol(df_plot)-3) : ncol(df_plot)] =
c('fit','se','lwr','upr')
df_plot[(ncol(df_plot)-3) : ncol(df_plot)] =
df_plot[(ncol(df_plot)-3) : ncol(df_plot)]/df_plot$memTask_max*100

# overall
# mod0 = brm( memTask|trials(memTask_max) ~ condition + (1|subject)+
(1|item), family='binomial', data=df, warmup=500, iter=1000,
chains=4, cores=4, prior=set_prior("normal(0,10)"))
# saveRDS(mod0, 'models/mod_mem~cond.RDS')
# mod0 = readRDS('models/mod_mem~cond.RDS')

summary(mod0)
df_plot0 = data.frame(emotionBlock='Overall',
condition=levels(df$condition))
df_plot0$memTask_max = ifelse(df_plot0$condition=="Load 1", 6, 8)
df_plot0 = cbind (df_plot0, fitted (mod0, newdata=df_plot0,
re_formula = NA))
colnames(df_plot0)[(ncol(df_plot0)-3) : ncol(df_plot0)] =
c('fit','se','lwr','upr')
df_plot0[(ncol(df_plot0)-3) : ncol(df_plot0)] =
df_plot0[(ncol(df_plot0)-3) : ncol(df_plot0)]/
df_plot0$memTask_max*100

df_plot = rbind(df_plot, df_plot0)
df_plot$emotion = factor(df_plot$emotion,
levels=c('Achievement','Amusement','Pleasure','Relief','Anger','Disg
ust','Fear','Sadness','Overall'))

ggplot (df_plot, aes(x=emotionBlock, y=fit, fill=condition)) +
  geom_bar(stat='identity', position=position_dodge(.7), width=.7) +
  geom_errorbar(aes(ymin=lwr, ymax=upr), position=position_dodge(.
7), width=.5) +
  geom_vline(xintercept=4.5, linetype=2) +
  geom_vline(xintercept=8.5, linetype=2) +
  scale_fill_manual(name='',values=c('gray50','gray20')) +
```

```
  xlab("") +
  ylab('Recall in memory task (%)') +
  theme_bw () +
  theme(legend.position="top", panel.grid=element_blank())

# ggsave ('pix/mem~cond*em.jpg', width=10, height=5, units='cm',
dpi=300, scale=2)


# contrasts between emotions for each condition
coda = posterior_samples(mod_brm)
colnames(coda)[1:16]

load1_gr = data.frame(
  ach_load1 = antilogit(coda[,1]),
  amu_load1 = antilogit(coda[,1]+coda[,2]),
  ang_load1 = antilogit(coda[,1]+coda[,3]),
  disg_load1 = antilogit(coda[,1]+coda[,4]),
  fear_load1 = antilogit(coda[,1]+coda[,5]),
  pls_load1 = antilogit(coda[,1]+coda[,6]),
  rel_load1 = antilogit(coda[,1]+coda[,7]),
  sad_load1 = antilogit(coda[,1]+coda[,8])
)

load1_gr$mean = apply(load1_gr, 1, mean)
for (i in 1:8){
  em = levels(df$emotionBlock)[i]
  contrast = load1_gr[,i]-load1_gr$mean
  q = round (quantile(contrast, probs=c(.5,.025,.975)), 2)
  print ( paste0 (em, ': ', q[1], ' [', q[2], ', ', q[3], ']') )
}  # signif different from average only for anger


load2_gr = data.frame(
  ach_load2 = antilogit(coda[,1]+coda[,9]),
  amu_load2 = antilogit(coda[,1]+coda[,9]+coda[,2]+coda[,10]),
  ang_load2 = antilogit(coda[,1]+coda[,9]+coda[,3]+coda[,11]),
  disg_load2 = antilogit(coda[,1]+coda[,9]+coda[,4]+coda[,12]),
  fear_load2 = antilogit(coda[,1]+coda[,9]+coda[,5]+coda[,13]),
  pls_load2 = antilogit(coda[,1]+coda[,9]+coda[,6]+coda[,14]),
  rel_load2 = antilogit(coda[,1]+coda[,9]+coda[,7]+coda[,15]),
  sad_load2 = antilogit(coda[,1]+coda[,9]+coda[,8]+coda[,16])
)

load2_gr$mean = apply(load2_gr, 1, mean)
for (i in 1:8){
  em = levels(df$emotionBlock)[i]
  contrast = load2_gr[,i]-load2_gr$mean
  q = round (quantile(contrast, probs=c(.5,.025,.975)), 2)
  print ( paste0 (em, ': ', q[1], ' [', q[2], ', ', q[3], ']') )
}  # signif different from average for achievement, anger, and
disgust
```