seq_analysis

Avery Katko

July 28, 2014

```
rm(list=ls())
library(ggplot2)
library(reshape2)
library(plyr)
library(bootstrap)
setwd("~/github/local/pragmods")
#source("useful.R")
d <- read.csv("seq_data/pragmods_seq.anondata.csv")
#head(d)
```

Exclude participants that either failed manipulation check or were rejected.

```
exclude <- d$assignmentstatus == "Rejected" |
  d$Answer.name_check_correct == "\"FALSE\""
sum(exclude)
```

```
## [1] 9
```

```
mean(exclude)
```

```
## [1] 0.045
```

```
d <- subset(d, exclude == FALSE)

d$Answer.choice_correct_1 <- factor(as.logical(d$Answer.choice_correct_1))
d$Answer.choice_correct_2 <- factor(as.logical(d$Answer.choice_correct_2))
d$Answer.choice_correct_3 <- factor(as.logical(d$Answer.choice_correct_3))
```

Restructure data to have trial variable. (working around mturk data submission limitations; can't safely submit arrays, so we have to use separate variables for each trial)

```
d$level_1 <- factor(ifelse(d$Answer.sequence_condition == "\"1w0w1b\"",1,0))
d$level_2 <- factor(ifelse(d$Answer.sequence_condition == "\"1w0w1b\"",0,1))
d$level_3 <- factor(rep(1,nrow(d)))

trial.df <- function(tn){
  df <- data.frame(id = d$workerid,
                   seqCond = d$Answer.sequence_condition,
                   trial = factor(rep(tn,nrow(d))),
                   level = d[sprintf("level_%d",tn)],
                   item = d[sprintf("Answer.item_%d",tn)],
                   targetProp = d[sprintf("Answer.target_prop_%d",tn)],
                   distractorProp = d[sprintf("Answer.distractor_prop_%d",tn)],
                   targetPosition = d[sprintf("Answer.target_position_%d",tn)],
                   distractorPosition = d[sprintf("Answer.distractor_position_%d",tn)],
```

```
                  choice = d[sprintf("Answer.choice_%d",tn)],
                  choiceCorrect = d[sprintf("Answer.choice_correct_%d",tn)])
  names(df) <- c("id","seqCond","trial","level","item","targetProp","distractorProp",
                 "targetPosition","distractorPosition","choice","choiceCorrect")
  return(df)
}

d2 <- rbind(trial.df(1),trial.df(2),trial.df(3))
summary(d2)
```

```
##        id           seqCond    trial   level                 item
##  Min.   :  1   "0w1w1b":240   1:191   0:191   "boat"          : 85
##  1st Qu.: 50   "1w0w1b":333   2:191   1:382   "Christmas tree":100
##  Median :100                  3:191           "friend"        : 93
##  Mean   :100                                  "pizza"         : 86
##  3rd Qu.:151                                  "snowman"       :105
##  Max.   :200                                  "sundae"        :104
##
##        targetProp      distractorProp  targetPosition distractorPosition
##  "hat"      : 71   "hat"      : 73    "left"  :196    "left"  :194
##  "chocolate": 43   "cherry"   : 45    "middle":195    "middle":185
##  "ornaments": 38   "mittens"  : 38    "right" :182    "right" :194
##  "scarf"    : 35   "star"     : 35
##  "mittens"  : 34   "glasses"  : 34
##  "glasses"  : 33   "ornaments": 33
##  (Other)    :319   (Other)    :315
##        choice    choiceCorrect
##  "foil"   : 20   FALSE: 88
##  "logical": 68   TRUE :485
##  "target" :485
##
##
##
##
```

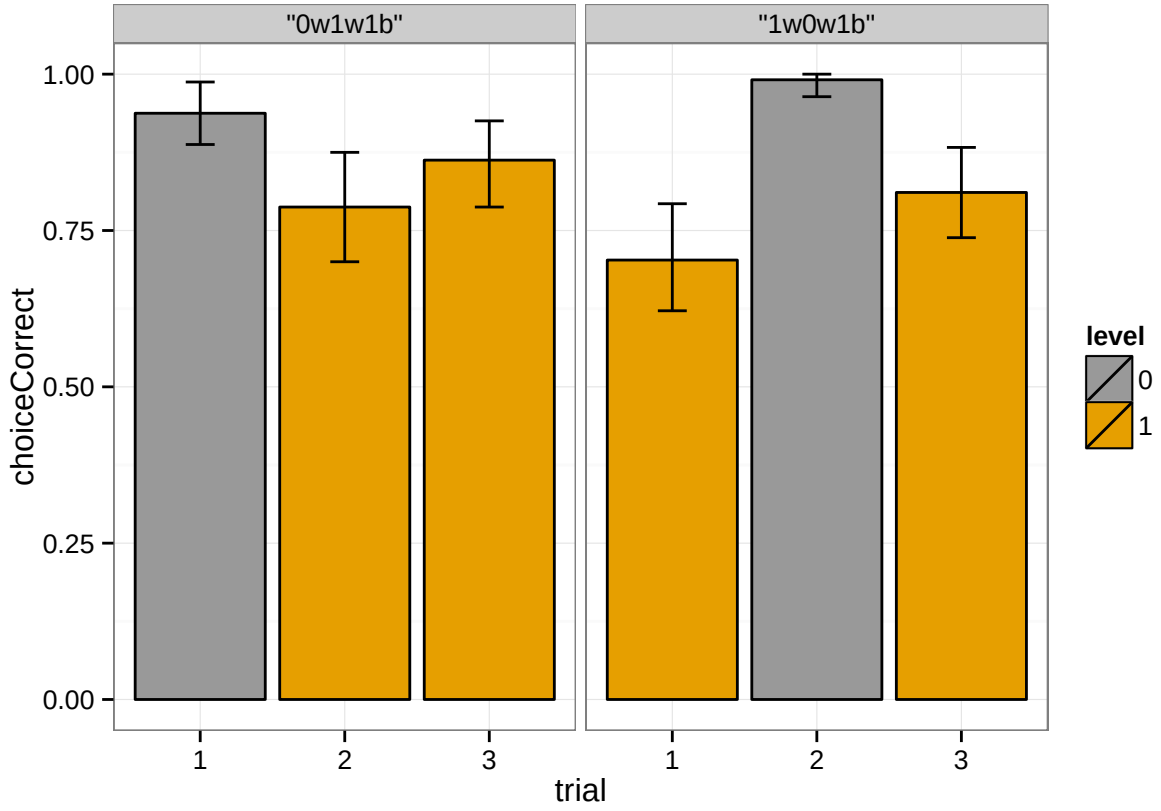
```
#statistics for boolean factors; copied from useful.R, with a slightly different mean function to work
l.mean <- function(...){mean(as.logical(...))}
l.theta <- function(x,xdata,na.rm=T) {l.mean(xdata[x],na.rm=na.rm)}
l.ci.low <- function(x,na.rm=T) {
  l.mean(x,na.rm=na.rm) - quantile(bootstrap(1:length(x),1000,l.theta,x,na.rm=na.rm)$thetastar,.025,na.
l.ci.high <- function(x,na.rm=T) {
  quantile(bootstrap(1:length(x),1000,l.theta,x,na.rm=na.rm)$thetastar,.975,na.rm=na.rm) - l.mean(x,na.

ms <- aggregate(choiceCorrect ~ seqCond + trial + level,data = d2,l.mean)
ms$cil <- aggregate(choiceCorrect ~ seqCond + trial + level, data = d2, l.ci.low)$choiceCorrect
ms$cih <- aggregate(choiceCorrect ~ seqCond + trial + level, data = d2, l.ci.high)$choiceCorrect

#colorblind-friendly color palettes
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
```

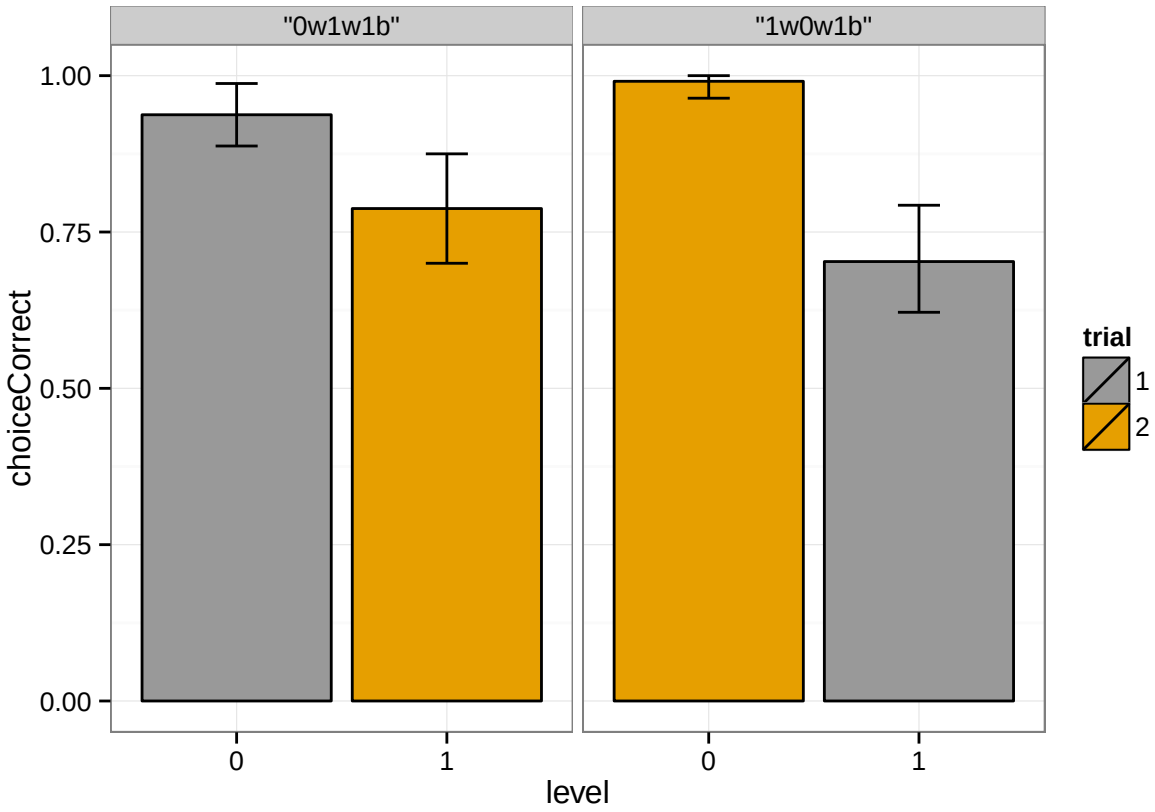
Compare performance on trials, in chronological order; facet by ordering condition.

```
ggplot(data = ms, aes(x = trial, y = choiceCorrect, fill = level)) +
  geom_bar(stat = "identity", color = "black") +
  geom_errorbar(aes(ymin = choiceCorrect - cil, ymax = choiceCorrect + cih), width = 0.2) +
  facet_grid(. ~ seqCond) +
  theme_bw() +
  scale_fill_manual(values=cbPalette)
```



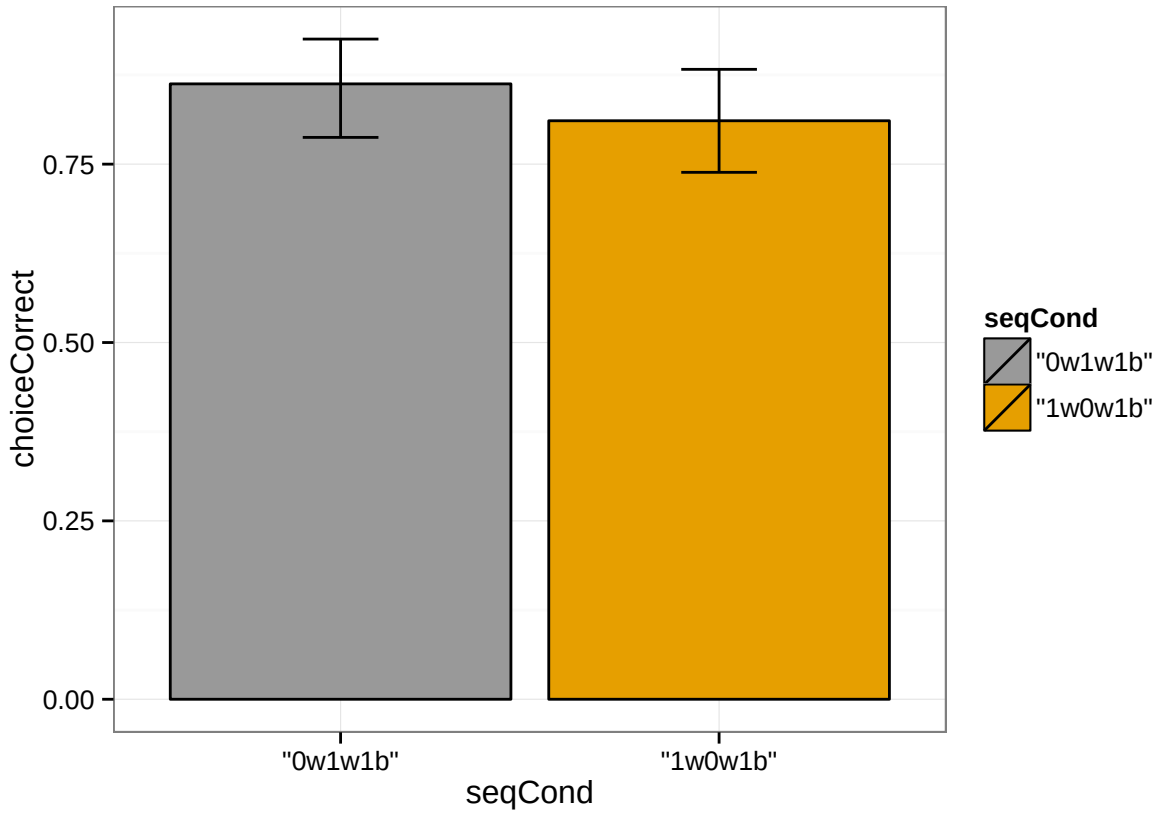
Compare performance on level-0 vs level-1 trials in the two conditions.

```
ggplot(data = subset(ms,trial != "3"), aes(x = level, y = choiceCorrect, fill = trial)) +
  geom_bar(stat = "identity", color = "black") +
  geom_errorbar(aes(ymin = choiceCorrect - cil, ymax = choiceCorrect + cih), width = 0.2) +
  facet_grid(. ~ seqCond) +
  theme_bw() +
  scale_fill_manual(values=cbPalette)
```



Compare performance on third trial (level-1, different stimuli)

```
ggplot(data = subset(ms,trial=="3"), aes(x = seqCond, y = choiceCorrect, fill = seqCond)) +
  geom_bar(stat = "identity", color = "black") +
  geom_errorbar(aes(ymin = choiceCorrect - cil, ymax = choiceCorrect + cih), width = 0.2) +
  theme_bw() +
  scale_fill_manual(values=cbPalette)
```



Chi-squared test between level-1 inferences in within trials

```
col_foil <- aggregate(choice ~ seqCond, data = d2[d2$level=="1" & d2$trial!="3",],
          function(...){sum(... == "\"foil\"")})

col_target <- aggregate(choice ~ seqCond, data = d2[d2$level=="1" & d2$trial!="3",],
          function(...){sum(... == "\"target\"")})

col_logical <- aggregate(choice ~ seqCond, data = d2[d2$level=="1" & d2$trial!="3",],
          function(...){sum(... == "\"logical\"")})

cont_table <- data.frame(cbind(col_foil,col_target[2],col_logical[2]))
names(cont_table) <- c("seqCond","foil","target","logical")

cont_table
```

```
##    seqCond foil target logical
## 1 "0w1w1b"    3     63      14
## 2 "1w0w1b"    6     78      27
```

```
chisq.test(cont_table[1:2,2:4])
```

```
## Warning: Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
```

```
## data:  cont_table[1:2, 2:4]
## X-squared = 1.732, df = 2, p-value = 0.4207
```