Eyewitness identification data analysis

Kym McCormick

15 May 2019

Analysis of experiment 3 Eyewitness identification data.

# Define lineup size and number of confidence level bins required  
n <- 8 #lineup size  
cbins <- 5  
  
# Extract required data from raw dataset  
din <- data %>%  
 # Select required variables ---  
 select(uid,  
 condition,   
 Test\_T1\_suspectIdentified,   
 confidence\_rating,   
 justification\_justification,  
 demographics\_age,   
 demographics\_gender,   
 demographics\_country)%>%  
 # Only include data from USA and includes a confidence rating ----  
 filter(demographics\_country == "USA", !is.na(confidence\_rating)) %>%  
 # Separate the confounded condition variable into separate variables ----  
 separate(condition,c("memory","expectation","target")) %>%  
 # Create boolean variables for each identification outcome ---  
 mutate(CID = if\_else(target == "P" & Test\_T1\_suspectIdentified == "F68", 1, 0),  
 Miss = if\_else(target == "P" & Test\_T1\_suspectIdentified == "Silhouette", 1, 0),  
 TPFoilID = if\_else(target == "P" & Test\_T1\_suspectIdentified != "F68" & Test\_T1\_suspectIdentified != "Silhouette", 1, 0),  
 CR = if\_else(target == "A" & Test\_T1\_suspectIdentified == "Silhouette", 1, 0),  
 TAFoilID = if\_else(target == "A" & Test\_T1\_suspectIdentified != "Silhouette", 1, 0)  
 )  
  
# There are twice the number of target present lineups than target absent lineups. To rebalance this in a way that preserves the groupings of confidence intervals, I have chosen to double the Target Absent data before grouping into equal sized confidence rating bins. The following function does this and returns the data matrix with a new variable: decile.  
  
Con\_Perc <- function(data) {  
 # Separate data into target present and absent dataframes ---  
 Ratings\_Absent <- data %>% filter(target == "A")  
 Ratings\_Present <- data %>% filter(target == "P")  
 # Bind back together, doubling up on the target absent group ---  
 # NOTE THAT DUPLICATES MAY BE REMOVED USING THE UID NUMBERS ---  
 Ratings <- rbind(Ratings\_Absent,Ratings\_Absent,Ratings\_Present)  
 # Group chooser confidence into decile confidence rating bins  
 din\_mod\_choose <- Ratings %>%  
 filter(Test\_T1\_suspectIdentified != "Silhouette") %>%  
 mutate(c = ntile(confidence\_rating,cbins)  
 )  
 # Group non-chooser confidence into a single zero confidence rating bin ---  
 din\_mod\_nochoose <- Ratings %>%  
 filter(Test\_T1\_suspectIdentified == "Silhouette") %>%  
 mutate(c = 0)  
 # Bind chooser and non-chooser data back together ---  
 din\_mod <- rbind(din\_mod\_choose, din\_mod\_nochoose)  
 return(din\_mod)  
}  
  
# Apply the above function to the cleaned dataset  
din\_mod <- Con\_Perc(din)  
  
# Check the grouping of the confidence rating deciles to ensure evenness  
din\_mod %>%  
 select(c) %>%  
 map(~prop.table(table(.)))

## $c  
## .  
## 0 1 2 3 4 5   
## 0.2444277 0.1511145 0.1511145 0.1511145 0.1511145 0.1511145

## Demographics

age <- describe(din$demographics\_age)  
age

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 2012 37.15 11.42 35 35.91 10.38 18 81 63 0.92 0.26  
## se  
## X1 0.25

gender <- din %>%   
 select(demographics\_gender) %>%   
 map(~prop.table(table(.)))  
gender

## $demographics\_gender  
## .  
## female male other   
## 0.000000000 0.509443340 0.484095427 0.006461233

## Identification counts

# Counts within each confidence level and accross manipulation groups ---  
obsData <- din\_mod %>%  
 group\_by(c) %>%  
 summarise(  
 CID = sum(CID),  
 TA = sum(CID)+sum(TPFoilID),  
 FA = sum(TPFoilID)  
 ) %>%  
 select (-c)  
  
# Reverse the order of the confidence level groups (from highest to lowest) ---  
obsData <- t(obsData[6:2,])  
  
# Calculate sums for non-identifications  
cZero <- din\_mod %>%  
 filter(c == "0")%>%  
 summarise(  
 CID = 0,  
 TA = sum(Miss),  
 FA = sum(CR)  
 ) %>%  
 t()  
  
# Bind together into a matrix ---  
obsData <- as.matrix(cbind(obsData,cZero[,1]))  
obsData

## [,1] [,2] [,3] [,4] [,5] [,6]  
## CID 191 120 104 81 76 0  
## TA 263 248 217 186 202 261  
## FA 72 128 113 105 126 386

#obsData

# Strong memory observed counts  
obsData\_S <- din\_mod %>%  
 filter(memory == "S")%>%  
 group\_by(c) %>%  
 summarise(  
 CID = sum(CID),  
 TA = sum(CID)+sum(TPFoilID),  
 FA = sum(TPFoilID)  
 ) %>%  
 select (-c)  
   
obsData\_S <- t(obsData\_S[6:2,])  
  
cZero\_S <- din\_mod %>%  
 filter(c == "0", memory == "S") %>%  
 summarise(CID = 0, TA = sum(Miss), FA = sum(CR)) %>%  
 t()  
  
obsData\_S <- as.matrix(cbind(obsData\_S,cZero\_S[,1]))  
obsData\_S

## [,1] [,2] [,3] [,4] [,5] [,6]  
## CID 109 74 62 54 37 0  
## TA 152 141 120 111 92 109  
## FA 43 67 58 57 55 198

# Weak memory observed counts  
obsData\_W <- din\_mod %>%  
 filter(memory == "W")%>%  
 group\_by(c) %>%  
 summarise(  
 CID = sum(CID),  
 TA = sum(CID)+sum(TPFoilID),  
 FA = sum(TPFoilID)  
 ) %>%  
 select (-c)  
   
obsData\_W <- t(obsData\_W[6:2,])  
  
cZero\_W <- din\_mod %>%  
 filter(c == "0", memory == "W")%>%  
 summarise(CID = 0, TA = sum(Miss), FA = sum(CR)) %>%  
 t()  
  
obsData\_W <- as.matrix(cbind(obsData\_W,cZero\_W[,1]))  
obsData\_W

## [,1] [,2] [,3] [,4] [,5] [,6]  
## CID 82 46 42 27 39 0  
## TA 111 107 97 75 110 152  
## FA 29 61 55 48 71 188

#Likelihood functions generate predicted data.  
#Given a particular set of parameters that define the likelihood surface, they give the most likely data  
  
#Definitions:  
#CID - target identification, i.e. selection of the target when the target is present  
#TA - target detection, i.e. selection of either a target or a foil when a target is present  
#FA - false alarm, i.e. selection of a foil when no target is present  
  
  
#Predicted proportion of Correct IDs according to MAX model  
QT <- function(c,d,s,n){  
 m <- function(x) dnorm(x,mean = d, sd = s)\*pnorm(x)^(n-1)  
 p <- vector(mode = "integer", length = length(c))  
 for (i in 1:length(c)){  
 a <- integrate(m,c[i],15)   
 p[i] <- a$value  
 }  
 return(p)  
}  
  
#predicted proportion of total detections on TP trials MAX model  
TP <- function(c,d,s,n){  
 p <- vector(mode = "integer", length = length(c))  
 for (i in 1:length(c)){  
 p[i] <- pnorm(((c[i])-d)/s)\*pnorm(c[i])^(n-1)  
 }  
 p <- 1 - p  
 return(p)  
}  
  
#predicted proportion of total detections on TA trials MAX model  
TA <- function(c,n){  
 p = vector(mode = "integer", length = length(c))  
 for (i in 1:length(c)){  
 p[i] = pnorm(c[i])^n  
 }  
 p = 1 - p  
 return(p)  
}  
  
genpred <- function(pars, obs.data, n){  
 ## The variables below (c,d & s) are defined by pars, which are passed into the chisq function via "theta = x0" in the constrOptim function (see below)  
 c <- pars[1:(length(pars)-2)]  
 d <- pars[length(pars)-1]  
 s <- tail(pars,1)  
   
 total.TP <- sum(obs.data[2,])  
 total.TA <- sum(obs.data[3,])   
   
 CID <- QT(c(c, -15),d,s,n)  
 CID <- c(CID[1],diff(CID))  
   
 TDTP <- c(TP(c,d,s,n),1)  
 TDTP <- c(TDTP[1],diff(TDTP))  
   
 TDTA <- c(TA(c,n),1)  
 TDTA <- c(TDTA[1],diff(TDTA))  
   
 CID <- CID\*total.TP  
 TDTP <- TDTP\*total.TP  
 TDTA <- TDTA\*total.TA  
   
 pred.data <- rbind(CID,TDTP,TDTA)  
 rownames(pred.data) <- c()  
   
 return(pred.data)  
}  
  
#Chi-squared   
  
chisq <- function(pars,obs.data,n){  
   
 pred.data <- genpred(pars,obs.data,n)  
 lastcell <- ncol(obs.data)   
 nc <- ncol(obs.data)-1  
 f <- vector(mode = "integer", length = nrow(obs.data)\*ncol(obs.data)) #for storing and summing chi-sq fit value  
   
 for (i in 1:nc){  
   
 a <- pred.data[1,i] #Correct ID   
 b <- obs.data[1,i]  
 f[1] <- f[1] + (b-a)^2/a  
   
 a <- pred.data[2,i]-pred.data[1,i] #Foil ID on TP lineup   
 b <- obs.data[2,i]-obs.data[1,i]   
 f[2] = f[2] + (b-a)^2/a  
   
 a <- pred.data[3,i] #False Alarm   
 b <- obs.data[3,i]  
 f[3] <- f[3] + (b-a)^2/a  
 }  
   
 a <- pred.data[2,lastcell] #Rejection TP  
 b <- obs.data[2,lastcell]  
 f[4] <- (b-a)^2/a  
   
 a <- pred.data[3,lastcell] #Rejection TA  
 b <- obs.data[3,lastcell]  
 f[5] <- (b-a)^2/a  
   
 f <- sum(f)  
   
 return(f)   
}  
  
#optimisation  
  
## Define starting values (also used as pars variable)  
x0 = c(5,4,3,2,1,1,1) #c1, c2, c3, c4, c5, d, s  
  
## Define contstraint matrix ---  
A <- cbind(  
 c(1,0,0,0),  
 c(-1,1,0,0),  
 c(0,-1,1,0),  
 c(0,0,-1,1),  
 c(0,0,0,-1),  
 c(0,0,0,0),  
 c(0,0,0,0) #added extra column for s parameter  
 )   
  
## Define constraint vector   
b <- c(0,0,0,0)

# Estimate parameters from observed data

## Collapsed identification data

out <- constrOptim(  
 theta = x0, ## passed into the chisq function as "pars"  
 f = chisq,   
 grad = NULL,   
 ui = A,   
 ci = b,   
 mu = 1e-04,   
 method = "Nelder-Mead",   
 outer.iterations = 100,   
 obs.data = obsData,   
 n = n  
 )  
  
#get fit statistic and parameters from model fit  
  
chisq.modelfit <- out$value  
c.modelfit <- out$par[1:(length(out$par)-2)]  
d.modelfit <- out$par[length(out$par)-1]  
s.modelfit <- tail(out$par,1)  
  
pred.data <- genpred(out$par, obsData, n)  
chisq.modelfit

## [1] 19.94592

c.modelfit

## [1] 2.338194 1.924590 1.667831 1.461845 1.220607

d.modelfit

## [1] 1.291592

s.modelfit

## [1] 0.9549149

pred.data

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 183.07251 142.4595 98.62367 74.12414 71.39007 94.92729  
## [2,] 266.34096 263.0469 208.63620 172.77632 182.22118 283.97842  
## [3,] 69.68551 114.0644 117.08338 117.16492 149.56026 362.44159

## Strong memory identification data

out <- constrOptim(  
 theta = x0,   
 f = chisq,   
 grad = NULL,   
 ui = A,   
 ci = b,   
 mu = 1e-04,   
 method = "Nelder-Mead",   
 outer.iterations = 100,   
 obs.data = obsData\_S,   
 n = n  
 )  
  
chisq.modelfit <- out$value  
c.modelfit <- out$par[1:(length(out$par)-2)]  
d.modelfit <- out$par[length(out$par)-1]  
s.modelfit <- tail(out$par,1)  
  
pred.data <- genpred(out$par, obsData\_S, n)  
chisq.modelfit

## [1] 14.8581

c.modelfit

## [1] 2.287253 1.895171 1.644995 1.420571 1.206155

d.modelfit

## [1] 1.416162

s.modelfit

## [1] 0.8369076

pred.data

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 104.50166 85.10261 59.73211 48.58640 36.61049 49.35084  
## [2,] 154.34086 148.03404 114.89789 101.10968 81.91875 124.69879  
## [3,] 40.79996 59.57829 60.46789 66.93067 68.51571 181.70748

## Weak memory identification data

out <- constrOptim(  
 theta = x0,   
 f = chisq,   
 grad = NULL,   
 ui = A,   
 ci = b,   
 mu = 1e-04,   
 method = "Nelder-Mead",   
 outer.iterations = 100,   
 obs.data = obsData\_W,   
 n = n  
 )  
  
#get fit statistic and parameters from model fit  
  
chisq.modelfit <- out$value  
c.modelfit <- out$par[1:(length(out$par)-2)]  
d.modelfit <- out$par[length(out$par)-1]  
s.modelfit <- tail(out$par,1)  
  
pred.data\_W <- genpred(out$par, obsData\_W, n)  
chisq.modelfit

## [1] 8.259845

c.modelfit

## [1] 2.398589 1.958411 1.694406 1.507982 1.235278

d.modelfit

## [1] 1.109326

s.modelfit

## [1] 1.107116

pred.data\_W

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 77.98870 58.00463 39.29333 26.69390 33.25408 43.04941  
## [2,] 111.78203 115.38185 93.74607 71.68073 100.11944 159.28988  
## [3,] 28.91364 54.23496 56.37080 50.21985 81.69274 180.56800