Analysis of Eyewitness ranking data

Kym McCormick

15 May 2019

## Demographics

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 2016 37.17 11.41 35 35.93 10.38 18 81 63 0.92 0.26  
## se  
## X1 0.25

## $demographics\_gender  
## .  
## female male other   
## 0.000000000 0.509424603 0.484126984 0.006448413

## Counts of correct identifications within each rank position

### Across different levels of memory

## Create a vector of rank counts ---  
Rank\_dvector <- din %>%   
   
## Only include target present lineup data ---   
 filter(target == "P") %>%  
   
## Separate into levels of memory strength (High and Low) ---   
 group\_by(memory)%>%  
   
## Sum the correct selection of the target across each rank postion ---   
 summarise(r1 = sum(R1\_Corr),  
 r2 = sum(R2\_Corr),  
 r3 = sum(R3\_Corr),  
 r4 = sum(R4\_Corr),  
 r5 = sum(R5\_Corr),  
 r6 = sum(R6\_Corr),  
 r7 = sum(R7\_Corr),  
 r8 = sum(R8\_Corr)  
 ) %>%   
  
## Return as a structured vector   
 select(-memory) %>%  
 as.matrix() %>%   
 as.vector %>%  
 structure(.Dim= c(2L,8L))

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]  
## [1,] 365 118 55 51 46 49 28 13  
## [2,] 265 123 56 65 52 46 28 19

### Collapsed

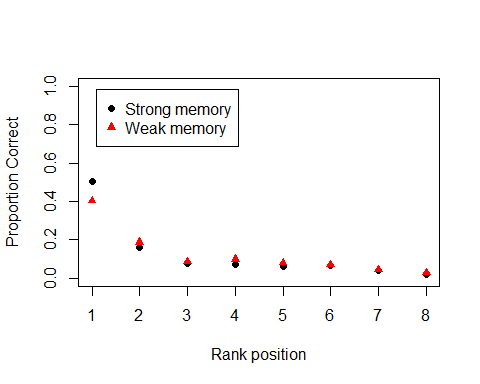
## Create a vector of rank counts ---  
Rank\_dvector\_collapsed <- din %>%   
   
## Only include target present lineup data ---   
 filter(target == "P") %>%  
   
## Sum the correct selection of the target across each rank postion ---   
 summarise(r1 = sum(R1\_Corr),  
 r2 = sum(R2\_Corr),  
 r3 = sum(R3\_Corr),  
 r4 = sum(R4\_Corr),  
 r5 = sum(R5\_Corr),  
 r6 = sum(R6\_Corr),  
 r7 = sum(R7\_Corr),  
 r8 = sum(R8\_Corr)  
 ) %>%   
  
## Return as a structured vector   
 as.matrix() %>%   
 as.vector %>%  
 structure(.Dim= c(1L,8L))  
  
## Bind the resulting vector to Rank\_dvector  
Rank\_dvector <- rbind(Rank\_dvector,Rank\_dvector\_collapsed)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]  
## [1,] 365 118 55 51 46 49 28 13  
## [2,] 265 123 56 65 52 46 28 19  
## [3,] 630 241 111 116 98 95 56 32

## Proportional correct

Rank\_prop <- din %>%  
 filter(target == "P")%>%  
 group\_by(memory) %>%  
 summarise(Rank\_1 = sum(R1\_Corr)/sum(n),  
 Rank\_2 = sum(R2\_Corr)/sum(n),  
 Rank\_3 = sum(R3\_Corr)/sum(n),  
 Rank\_4 = sum(R4\_Corr)/sum(n),  
 Rank\_5 = sum(R5\_Corr)/sum(n),  
 Rank\_6 = sum(R6\_Corr)/sum(n),  
 Rank\_7 = sum(R7\_Corr)/sum(n),  
 Rank\_8 = sum(R8\_Corr)/sum(n),  
 n = n()  
 )  
 Rank\_prop

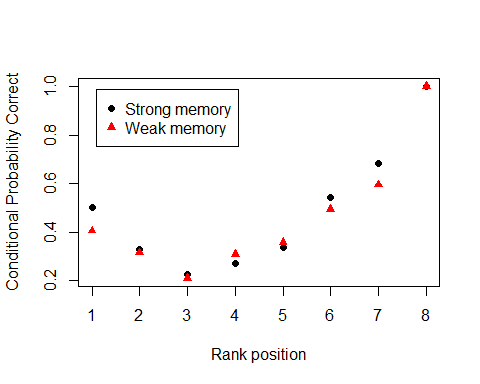
## # A tibble: 2 x 10  
## memory Rank\_1 Rank\_2 Rank\_3 Rank\_4 Rank\_5 Rank\_6 Rank\_7 Rank\_8 n  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>  
## 1 S 0.503 0.163 0.0758 0.0702 0.0634 0.0675 0.0386 0.0179 726  
## 2 W 0.405 0.188 0.0856 0.0994 0.0795 0.0703 0.0428 0.0291 654

 ##Conditional rank probabilities

## Create a vector of conditional rank probabilities ---  
Cond\_Cum\_Rank <- din %>%   
   
## Only include target present lineup data ---   
 filter(target == "P") %>%  
   
## Separate into levels of memory strength (High and Low) ---   
 group\_by(memory)%>%  
   
## Sum the correct selection of the target across each rank postion ---   
 summarise(r1 = sum(R1\_Corr),  
 r2 = sum(R2\_Corr),  
 r3 = sum(R3\_Corr),  
 r4 = sum(R4\_Corr),  
 r5 = sum(R5\_Corr),  
 r6 = sum(R6\_Corr),  
 r7 = sum(R7\_Corr),  
 r8 = sum(R8\_Corr),  
 n = r1+r2+r3+r4+r5+r6+r7+r8  
 ) %>%   
   
## Calcluate conditional rank probabilities ---  
 mutate(c1 = r1/n,  
 c2 = r2/(n-r1),  
 c3 = r3/(n-r1-r2),  
 c4 = r4/(n-r1-r2-r3),  
 c5 = r5/(n-r1-r2-r3-r4),  
 c6 = r6/(n-r1-r2-r3-r4-r5),  
 c7 = r7/(n-r1-r2-r3-r4-r5-r6),  
 c8 = r8/(n-r1-r2-r3-r4-r5-r6-r7),  
 ) %>%  
  
## Return as a structured vector   
 select(-memory,-n) %>%  
 as.matrix() %>%   
 as.vector %>%  
 structure(.Dim= c(2L,16L))  
  
## Remove ranking counts ---  
Cond\_Cum\_Rank <- Cond\_Cum\_Rank[,9:16]  
Cond\_Cum\_Rank

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.5034483 0.3277778 0.2272727 0.2727273 0.3382353 0.5444444 0.6829268  
## [2,] 0.4051988 0.3161954 0.2105263 0.3095238 0.3586207 0.4946237 0.5957447  
## [,8]  
## [1,] 1  
## [2,] 1

matplot(t(Cond\_Cum\_Rank), type = "p", bty="l", pch=c(19,17), ylab = "Conditional Probability Correct", xlab = "Rank position")  
 legend("topleft", legend = c("Strong memory","Weak memory"),pch =c(19,17), col = c("black","red"), inset = .05)  
 box("plot")



## Estimation of UV-SDT parameters

Model

expSDTrank <- function(Q, param.names, n.params, tmp.env){  
 n <- 8  
 e <- vector("numeric", n)  
 mu <- Q[1]  
 ss <- Q[2]  
 G <- function(x,i) {  
 (pnorm(x)^(n-i))\*dnorm(x, mean = mu, sd = ss)\*(1-pnorm(x))^(i-1)\*choose(n-1, i-1)  
 }  
   
 for (ii in 1:n) {  
 e[ii] <- integrate(G,-Inf,Inf,i = ii, rel.tol = .Machine$double.eps^0.5)$value  
 }  
 return(e)  
}

Fitting function

SDTrank <- function(Q, data, param.names, n.params, tmp.env, lower.bound, upper.bound){  
 e <- expSDTrank(Q, param.names, n.params, tmp.env)  
 LL <- -sum(data[data!=0]\*log(e[data!=0]))  
 return(LL)  
 }

### Results

Strong memory

## Presenting the best result out of 5 minimization runs.

## [1] "Model fitting begins at 2019-05-15 09:11:37"  
## [1] "Model fitting stopped at 2019-05-15 09:11:38"  
## Time difference of 0.866504 secs

## No function for computing Hessian Matrix specified or it failed. Hessian Matrix is estimated numerically. Validity of CIs is questionable.

## Note: CIs are based on the numerically estimated Hessian matrix

## Log.Likelihood G.Squared df p.value  
## 1 -1153.079 17.79353 5 0.003216612

## estimates lower.conf upper.conf  
## mu 1.335476 1.206000 1.464951  
## sigma 1.293614 1.164479 1.422749

Weak memory

## Presenting the best result out of 5 minimization runs.

## [1] "Model fitting begins at 2019-05-15 09:11:38"  
## [1] "Model fitting stopped at 2019-05-15 09:11:38"  
## Time difference of 0.3607981 secs

## No function for computing Hessian Matrix specified or it failed. Hessian Matrix is estimated numerically. Validity of CIs is questionable.

## Note: CIs are based on the numerically estimated Hessian matrix

## Log.Likelihood G.Squared df p.value  
## 1 -1147.708 11.73185 5 0.0386527

## estimates lower.conf upper.conf  
## mu 1.016965 0.9036574 1.130272  
## sigma 1.159893 1.0448971 1.274889

Combined

## Presenting the best result out of 5 minimization runs.

## [1] "Model fitting begins at 2019-05-15 09:11:38"  
## [1] "Model fitting stopped at 2019-05-15 09:11:39"  
## Time difference of 0.6706159 secs

## No function for computing Hessian Matrix specified or it failed. Hessian Matrix is estimated numerically. Validity of CIs is questionable.

## Note: CIs are based on the numerically estimated Hessian matrix

## Log.Likelihood G.Squared df p.value  
## 1 -2307.446 27.16114 5 5.306498e-05

## estimates lower.conf upper.conf  
## mu 1.177031 1.091317 1.262745  
## sigma 1.230494 1.144104 1.316883