Analysis of Eyewitness ranking data

Kym McCormick

15 May 2019

## Warning: Expected 8 pieces. Missing pieces filled with `NA` in 1 rows  
## [1362].

## Demographics

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 2029 37.48 11.54 35 36.29 10.38 18 99 81 0.94 0.62  
## se  
## X1 0.26

## $demographics\_gender  
## .  
## female male other   
## 0.000000000 0.532281912 0.462789552 0.004928536

## 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)  
 ) %>%   
 ungroup()%>%  
select(-memory) %>%  
  
## Return as a structured vector   
 ## Return as a structured vector   
 as.matrix() %>%   
 as.vector %>%  
 structure(.Dim= c(3L,8L))

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]  
## [1,] 365 118 55 50 46 49 28 13  
## [2,] 224 95 63 56 56 75 39 43  
## [3,] 265 122 56 65 52 46 28 19

### Collapsed

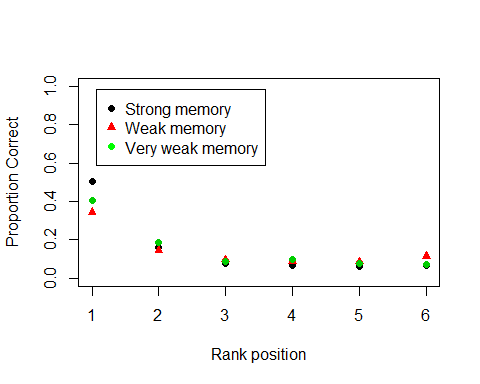
## Create a vector of rank counts ---  
Rank\_dvector\_collapsed <- din %>%   
   
## Only include target present lineup data ---   
 filter(target == "P") %>%  
   
## Sum the correct selection of the target across each rank postion ---   
 summarise(r1 = sum(R1\_Corr),  
 r2 = sum(R2\_Corr),  
 r3 = 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 50 46 49 28 13  
## [2,] 224 95 63 56 56 75 39 43  
## [3,] 265 122 56 65 52 46 28 19  
## [4,] 854 335 174 171 154 170 95 75

## 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: 3 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 0.0759 0.0690 0.0634 0.0676 0.0386 0.0179 725  
## 2 VW 0.344 0.146 0.0968 0.0860 0.0860 0.115 0.0599 0.0661 651  
## 3 W 0.406 0.187 0.0858 0.0995 0.0796 0.0704 0.0429 0.0291 653

 ##Conditional rank probabilities

## Create a vector of conditional rank probabilities ---  
Cond\_Cum\_R <- din %>%   
   
## Only include target present lineup data ---   
 filter(target == "P") %>%  
   
## Separate into levels of memory strength (High and Low) ---   
 group\_by(memory)%>%  
 print()%>%  
## 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  
 ) %>%   
 print()%>%  
   
## 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/(r8)  
 )

## # A tibble: 2,029 x 27  
## # Groups: memory [3]  
## uid memory expectation target Rank\_T1\_lineupO~ Rank\_1 Rank\_2 Rank\_3  
## <dbl> <chr> <chr> <chr> <fct> <chr> <chr> <chr>   
## 1 4.65e15 S H P F71:F34:F176:F1~ F147 F34 F176   
## 2 4.66e15 S H P F34:F50:F71:F68~ F71 F34 F50   
## 3 4.68e15 W H P F34:F68:F88:F71~ F34 F88 F50   
## 4 4.76e15 S L P F50:F34:F88:F14~ F147 F34 F68   
## 5 5.08e15 W L P F68:F88:F71:F17~ F147 F68 F34   
## 6 5.15e15 W L P F147:F71:F50:F1~ F34 F147 F88   
## 7 5.30e15 S L P F50:F71:F68:F14~ F68 F147 F50   
## 8 5.43e15 S L P F134:F34:F177:F~ F34 F147 F134   
## 9 5.44e15 W L P F34:F68:F134:F1~ F147 F68 F177   
## 10 5.44e15 W L P F50:F134:F88:F1~ F176 F147 F134   
## # ... with 2,019 more rows, and 19 more variables: Rank\_4 <chr>,  
## # Rank\_5 <chr>, Rank\_6 <chr>, Rank\_7 <chr>, Rank\_8 <chr>,  
## # Test\_T1\_suspectIdentified <fct>, demographics\_age <int>,  
## # demographics\_gender <fct>, demographics\_country <fct>, date <date>,  
## # R1\_Corr <dbl>, R2\_Corr <dbl>, R3\_Corr <dbl>, R4\_Corr <dbl>,  
## # R5\_Corr <dbl>, R6\_Corr <dbl>, R7\_Corr <dbl>, R8\_Corr <dbl>, n <dbl>  
## # A tibble: 3 x 10  
## memory r1 r2 r3 r4 r5 r6 r7 r8 n  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 S 365 118 55 50 46 49 28 13 724  
## 2 VW 224 95 63 56 56 75 39 43 651  
## 3 W 265 122 56 65 52 46 28 19 653

Cond\_Cum\_R

## # A tibble: 3 x 18  
## memory r1 r2 r3 r4 r5 r6 r7 r8 n c1 c2  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 S 365 118 55 50 46 49 28 13 724 0.504 0.329  
## 2 VW 224 95 63 56 56 75 39 43 651 0.344 0.222  
## 3 W 265 122 56 65 52 46 28 19 653 0.406 0.314  
## # ... with 6 more variables: c3 <dbl>, c4 <dbl>, c5 <dbl>, c6 <dbl>,  
## # c7 <dbl>, c8 <dbl>

## Create a structured vector  
Cond\_Cum\_RV <- Cond\_Cum\_R %>%  
 select(-memory,-n) %>%  
 as.matrix() %>%  
 as.vector() %>%  
 structure(.Dim = c(3L,16L))  
  
Cond\_Cum\_RV <- Cond\_Cum\_RV[,9:16]  
Cond\_Cum\_RV

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.5041436 0.3286908 0.2282158 0.2688172 0.3382353 0.5444444 0.6829268  
## [2,] 0.3440860 0.2224824 0.1897590 0.2081784 0.2629108 0.4777070 0.4756098  
## [3,] 0.4058193 0.3144330 0.2105263 0.3095238 0.3586207 0.4946237 0.5957447  
## [,8]  
## [1,] 1  
## [2,] 1  
## [3,] 1

Strong <- as.data.frame(t(Cond\_Cum\_R[1,11:18]))  
Strong

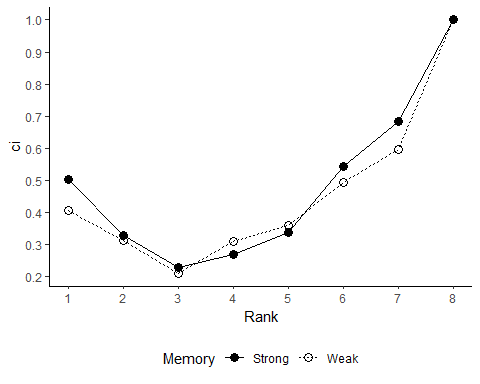
## V1  
## c1 0.5041436  
## c2 0.3286908  
## c3 0.2282158  
## c4 0.2688172  
## c5 0.3382353  
## c6 0.5444444  
## c7 0.6829268  
## c8 1.0000000

Strong$Rank <- c(1:8)  
Strong$Memory <- c(rep("Strong",2))  
  
Weak <- as.data.frame(t(Cond\_Cum\_R[3,11:18]))  
Weak$Rank <- c(1:8)  
Weak$Memory <- c(rep("Weak",2))  
  
VeryWeak <- as.data.frame(t(Cond\_Cum\_R[2,11:18]))  
VeryWeak$Rank <- c(1:8)  
VeryWeak$Memory <- c(rep("VeryWeak",2))  
  
ci <- rbind(Strong,Weak,VeryWeak)  
  
ci <- ci %>%  
 rename(ci = V1)

ciSVW <- ci %>%  
 filter(Memory != "VeryWeak")  
ciSVW

## ci Rank Memory  
## 1 0.5041436 1 Strong  
## 2 0.3286908 2 Strong  
## 3 0.2282158 3 Strong  
## 4 0.2688172 4 Strong  
## 5 0.3382353 5 Strong  
## 6 0.5444444 6 Strong  
## 7 0.6829268 7 Strong  
## 8 1.0000000 8 Strong  
## 9 0.4058193 1 Weak  
## 10 0.3144330 2 Weak  
## 11 0.2105263 3 Weak  
## 12 0.3095238 4 Weak  
## 13 0.3586207 5 Weak  
## 14 0.4946237 6 Weak  
## 15 0.5957447 7 Weak  
## 16 1.0000000 8 Weak

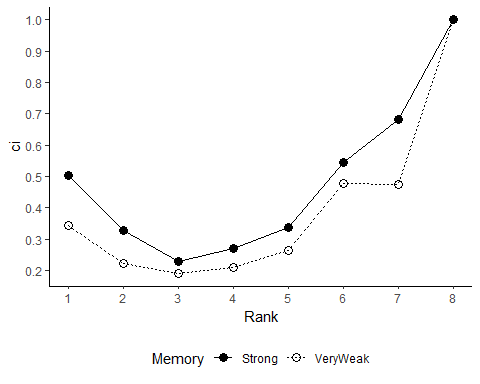
ciPlot1 <- ggplot(ciSVW,aes(x = Rank, y = ci))+  
 geom\_line(aes(linetype = Memory)) +  
 geom\_point(aes(shape = Memory), size = 3) +  
 scale\_shape\_manual(values = c(16, 1)) +  
 theme\_classic()+  
 theme(legend.position="bottom")+  
 scale\_x\_continuous(breaks = scales::pretty\_breaks(n = 8)) +   
 scale\_y\_continuous(breaks = scales::pretty\_breaks(n = 10))  
ciPlot1



ciSW <- ci %>%  
 filter(Memory != "Weak")  
ciSW

## ci Rank Memory  
## 1 0.5041436 1 Strong  
## 2 0.3286908 2 Strong  
## 3 0.2282158 3 Strong  
## 4 0.2688172 4 Strong  
## 5 0.3382353 5 Strong  
## 6 0.5444444 6 Strong  
## 7 0.6829268 7 Strong  
## 8 1.0000000 8 Strong  
## 9 0.3440860 1 VeryWeak  
## 10 0.2224824 2 VeryWeak  
## 11 0.1897590 3 VeryWeak  
## 12 0.2081784 4 VeryWeak  
## 13 0.2629108 5 VeryWeak  
## 14 0.4777070 6 VeryWeak  
## 15 0.4756098 7 VeryWeak  
## 16 1.0000000 8 VeryWeak

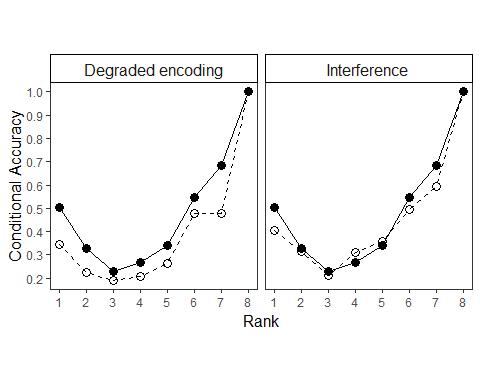
ciPlot2 <- ggplot(ciSW,aes(x = Rank, y = ci))+  
 geom\_line(aes(linetype = Memory)) +  
 geom\_point(aes(shape = Memory), size = 3) +  
 scale\_shape\_manual(values = c(16, 1)) +  
 theme\_classic() +  
 theme(legend.position="bottom") +  
 scale\_x\_continuous(breaks = scales::pretty\_breaks(n = 6)) +   
 scale\_y\_continuous(breaks = scales::pretty\_breaks(n = 10))  
ciPlot2



ciWVW <- ci %>%  
 filter(Memory != "Strong")  
ciWVW

## ci Rank Memory  
## 1 0.4058193 1 Weak  
## 2 0.3144330 2 Weak  
## 3 0.2105263 3 Weak  
## 4 0.3095238 4 Weak  
## 5 0.3586207 5 Weak  
## 6 0.4946237 6 Weak  
## 7 0.5957447 7 Weak  
## 8 1.0000000 8 Weak  
## 9 0.3440860 1 VeryWeak  
## 10 0.2224824 2 VeryWeak  
## 11 0.1897590 3 VeryWeak  
## 12 0.2081784 4 VeryWeak  
## 13 0.2629108 5 VeryWeak  
## 14 0.4777070 6 VeryWeak  
## 15 0.4756098 7 VeryWeak  
## 16 1.0000000 8 VeryWeak

ciS <- ci %>%  
 filter(Memory == "Strong")%>%  
 rename(Strong = Memory)  
  
labels <- c(VeryWeak = "Degraded encoding", Weak = "Interference")  
  
ciPlot <- ggplot(ciWVW,aes(x = Rank, y = ci))+  
 geom\_line(linetype = 2) +  
 geom\_point(shape = 1, size = 3) +  
 geom\_line(data = ciS, linetype = 1)+  
 geom\_point(data = ciS, shape = 16, size = 3)+  
 ylab("Conditional Accuracy")+  
 theme\_bw() +  
 theme(panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 axis.title.y = element\_text(size=12),  
 axis.title.x = element\_text(size=12),  
 strip.text.x = element\_text(size=12),  
 strip.background = element\_rect(colour="black", fill=("0"))  
 ) +  
  
 scale\_x\_continuous(breaks = scales::pretty\_breaks(n = 6)) +   
 scale\_y\_continuous(breaks = scales::pretty\_breaks(n = 10))+  
 facet\_grid(.~Memory, labeller=labeller(Memory = labels))+   
 theme(aspect.ratio = 1)  
  
ciPlot



svg("C:/Users/User/Documents/Kym/PhD-Thesis/analysis/Experiment 3/ciPlot.svg")  
print(ciPlot) # Plot 1 --> in the first page of PDF  
#print(myplot2) # Plot 2 ---> in the second page of the PDF  
dev.off()

## png   
## 2

pdf("C:/Users/User/Documents/Kym/PhD-Thesis/analysis/Experiment 3/ciPlot.pdf")  
print(ciPlot) # Plot 1 --> in the first page of PDF  
#print(myplot2) # Plot 2 ---> in the second page of the PDF  
dev.off()

## png   
## 2

Setting up ranking to look at between choosers and non-choosers

Choosing <- din %>%  
 select(memory, expectation, target, R1\_Corr, R2\_Corr, R3\_Corr, R4\_Corr, R5\_Corr, R6\_Corr, R7\_Corr, R8\_Corr, Test\_T1\_suspectIdentified)%>%  
 mutate(choose = if\_else(Test\_T1\_suspectIdentified != "Silhouette", 1, 0))  
   
## Create a vector of conditional rank probabilities ---  
Choosing <- Choosing %>%   
   
## Only include target present lineup data ---   
 filter(target == "P") %>%  
   
## Separate into levels of memory strength (High and Low) ---   
 group\_by(choose, 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/(r8)  
 )   
Choosing

## # A tibble: 6 x 19  
## # Groups: choose [2]  
## choose memory r1 r2 r3 r4 r5 r6 r7 r8 n c1  
## <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0 S 26 21 10 8 9 19 11 4 108 0.241  
## 2 0 VW 45 32 21 20 13 20 13 11 175 0.257  
## 3 0 W 31 30 17 26 17 13 9 9 152 0.204  
## 4 1 S 339 97 45 42 37 30 17 9 616 0.550  
## 5 1 VW 179 63 42 36 43 55 26 32 476 0.376  
## 6 1 W 234 92 39 39 35 33 19 10 501 0.467  
## # ... with 7 more variables: c2 <dbl>, c3 <dbl>, c4 <dbl>, c5 <dbl>,  
## # c6 <dbl>, c7 <dbl>, c8 <dbl>

## Create a dataframe from above for use in ggplot  
StrongChoose <- as.data.frame(t(Choosing[4,12:19]))  
StrongChoose$Rank <- c(1:8)  
StrongChoose$Memory <- c(rep("Strong",2))  
StrongChoose$Choose <- c(rep("Identify",2))  
  
StrongNoChoose <- as.data.frame(t(Choosing[1,12:19]))  
StrongNoChoose$Rank <- c(1:8)  
StrongNoChoose$Memory <- c(rep("Strong",2))  
StrongNoChoose$Choose <- c(rep("Reject",2))  
  
WeakChoose <- as.data.frame(t(Choosing[6,12:19]))  
WeakChoose$Rank <- c(1:8)  
WeakChoose$Memory <- c(rep("Weak",2))  
WeakChoose$Choose <- c(rep("Identify",2))  
  
WeakNoChoose <- as.data.frame(t(Choosing[3,12:19]))  
WeakNoChoose$Rank <- c(1:8)  
WeakNoChoose$Memory <- c(rep("Weak",2))  
WeakNoChoose$Choose <- c(rep("Reject",2))  
  
VeryWeakChoose <- as.data.frame(t(Choosing[4,12:19]))  
VeryWeakChoose$Rank <- c(1:8)  
VeryWeakChoose$Memory <- c(rep("Very Weak",2))  
VeryWeakChoose$Choose <- c(rep("Identify",2))  
  
VeryWeakNoChoose <- as.data.frame(t(Choosing[2,12:19]))  
VeryWeakNoChoose$Rank <- c(1:8)  
VeryWeakNoChoose$Memory <- c(rep("Very Weak",2))  
VeryWeakNoChoose$Choose <- c(rep("Reject",2))  
  
ciChoose <- rbind(StrongChoose,StrongNoChoose,WeakChoose,WeakNoChoose,VeryWeakChoose,VeryWeakNoChoose)  
  
ciChoose <- ciChoose %>%  
 rename(ci = V1)  
ciChoose

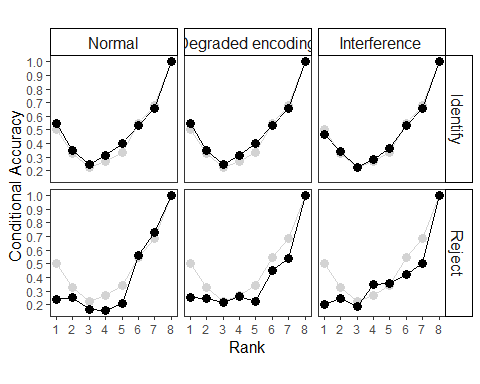
## ci Rank Memory Choose  
## c1 0.5503247 1 Strong Identify  
## c2 0.3501805 2 Strong Identify  
## c3 0.2500000 3 Strong Identify  
## c4 0.3111111 4 Strong Identify  
## c5 0.3978495 5 Strong Identify  
## c6 0.5357143 6 Strong Identify  
## c7 0.6538462 7 Strong Identify  
## c8 1.0000000 8 Strong Identify  
## c11 0.2407407 1 Strong Reject  
## c21 0.2560976 2 Strong Reject  
## c31 0.1639344 3 Strong Reject  
## c41 0.1568627 4 Strong Reject  
## c51 0.2093023 5 Strong Reject  
## c61 0.5588235 6 Strong Reject  
## c71 0.7333333 7 Strong Reject  
## c81 1.0000000 8 Strong Reject  
## c12 0.4670659 1 Weak Identify  
## c22 0.3445693 2 Weak Identify  
## c32 0.2228571 3 Weak Identify  
## c42 0.2867647 4 Weak Identify  
## c52 0.3608247 5 Weak Identify  
## c62 0.5322581 6 Weak Identify  
## c72 0.6551724 7 Weak Identify  
## c82 1.0000000 8 Weak Identify  
## c13 0.2039474 1 Weak Reject  
## c23 0.2479339 2 Weak Reject  
## c33 0.1868132 3 Weak Reject  
## c43 0.3513514 4 Weak Reject  
## c53 0.3541667 5 Weak Reject  
## c63 0.4193548 6 Weak Reject  
## c73 0.5000000 7 Weak Reject  
## c83 1.0000000 8 Weak Reject  
## c14 0.5503247 1 Very Weak Identify  
## c24 0.3501805 2 Very Weak Identify  
## c34 0.2500000 3 Very Weak Identify  
## c44 0.3111111 4 Very Weak Identify  
## c54 0.3978495 5 Very Weak Identify  
## c64 0.5357143 6 Very Weak Identify  
## c74 0.6538462 7 Very Weak Identify  
## c84 1.0000000 8 Very Weak Identify  
## c15 0.2571429 1 Very Weak Reject  
## c25 0.2461538 2 Very Weak Reject  
## c35 0.2142857 3 Very Weak Reject  
## c45 0.2597403 4 Very Weak Reject  
## c55 0.2280702 5 Very Weak Reject  
## c65 0.4545455 6 Very Weak Reject  
## c75 0.5416667 7 Very Weak Reject  
## c85 1.0000000 8 Very Weak Reject

## Create a structured vector  
Choosing <- Choosing %>%  
 ungroup()%>%  
 select(-choose,-memory,-n) %>%  
 as.matrix() %>%  
 as.vector() %>%  
 structure(.Dim = c(6L,16L))  
  
Choosing <- Choosing[,9:16]  
Choosing

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.2407407 0.2560976 0.1639344 0.1568627 0.2093023 0.5588235 0.7333333  
## [2,] 0.2571429 0.2461538 0.2142857 0.2597403 0.2280702 0.4545455 0.5416667  
## [3,] 0.2039474 0.2479339 0.1868132 0.3513514 0.3541667 0.4193548 0.5000000  
## [4,] 0.5503247 0.3501805 0.2500000 0.3111111 0.3978495 0.5357143 0.6538462  
## [5,] 0.3760504 0.2121212 0.1794872 0.1875000 0.2756410 0.4867257 0.4482759  
## [6,] 0.4670659 0.3445693 0.2228571 0.2867647 0.3608247 0.5322581 0.6551724  
## [,8]  
## [1,] 1  
## [2,] 1  
## [3,] 1  
## [4,] 1  
## [5,] 1  
## [6,] 1

Plot by choosers and memory

labels <- c(Strong = "Normal","Very Weak" = "Degraded encoding", Weak = "Interference")  
  
ciChoosePlot <- ggplot(ciChoose,aes(x = Rank, y = ci))+  
 geom\_line(data = ciS, linetype = 1, color = "light grey")+  
 geom\_point(data = ciS, shape = 16, size = 3, color = "light grey")+  
 geom\_line(linetype = 1) +  
 geom\_point(shape = 16, size = 3) +  
 ylab("Conditional Accuracy")+  
 theme\_bw() +  
 theme(panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 axis.title.y = element\_text(size=12),  
 axis.title.x = element\_text(size=12),  
 strip.text = element\_text(size=12),  
 strip.background = element\_rect(colour="black", fill=("0"))  
 ) +  
  
 scale\_x\_continuous(breaks = scales::pretty\_breaks(n = 6)) +   
 scale\_y\_continuous(breaks = scales::pretty\_breaks(n = 10))+  
 facet\_grid(Choose~Memory, labeller=labeller(Memory = labels)) +   
 theme(aspect.ratio = 1)  
  
ciChoosePlot



svg("C:/Users/User/Documents/Kym/PhD-Thesis/analysis/Experiment 3/ciChoosePlot.svg")  
print(ciChoosePlot) # Plot 1 --> in the first page of PDF  
#print(myplot2) # Plot 2 ---> in the second page of the PDF  
dev.off()

## png   
## 2

pdf("C:/Users/User/Documents/Kym/PhD-Thesis/analysis/Experiment 3/ciChoosePlot.pdf")  
print(ciChoosePlot) # Plot 1 --> in the first page of PDF  
#print(myplot2) # Plot 2 ---> in the second page of the PDF  
dev.off()

## png   
## 2

## 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 = 1)\*(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

## [1] "Model fitting begins at 2019-09-26 18:42:00"  
## [1] "Model fitting stopped at 2019-09-26 18:42:00"  
## Time difference of 0.05700302 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 -8.10041 0.8876801 6 0.9895141

## estimates  
## mu 0

Weak memory

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

## [1] "Model fitting begins at 2019-09-26 18:42:00"  
## [1] "Model fitting stopped at 2019-09-26 18:42:00"  
## Time difference of 0.05000186 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 -7.671666 0.8204468 5 0.9757109

## estimates  
## mu 0  
## sigma 1

Very Weak memory

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

## [1] "Model fitting begins at 2019-09-26 18:42:00"  
## [1] "Model fitting stopped at 2019-09-26 18:42:00"  
## Time difference of 0.05000305 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 -6.614149 1.065079 5 0.9571478

## estimates lower.conf upper.conf  
## mu 0 NA NA  
## sigma 1 NA NA

Combined

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

## [1] "Model fitting begins at 2019-09-26 18:42:00"  
## [1] "Model fitting stopped at 2019-09-26 18:42:00"  
## Time difference of 0.1340082 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 -1150.457 20.57892 5 0.0009726615

rank.para <- cbind(rank.weak.para,rank.comb.para,rank.strong.para)  
colnames(rank.para) <- c("weak", "combined", "strong")  
rank.para

## weak combined strong  
## mu 0 0.9592718 0  
## sigma 1 1.0000000 0