Supplement to: Opportunity for verbalization does not improve visual change detection performance: A state-trace analysis

Richard D. Morey and Florian Sense

Last compiled: 2016-03-31

Contents

The study

Download

A PDF of the paper can be downloaded here.

Abstract

Evidence suggests that there is a tendency to verbally recode visually-presented information, and that in some cases verbal recoding can boost memory performance. According to multi-component models of working memory, memory performance is increased because task-relevant information is simultaneously maintained in two codes. The possibility of dual encoding is problematic if the goal is to measure capacity for visual information exclusively. To counteract this possibility, articulatory suppression is frequently used with visual change detection tasks specifically to prevent verbalization of visual stimuli. But is this precaution always necessary? There is little reason to believe that concurrent articulation affects performance in typical visual change detection tasks, suggesting that verbal recoding might not be likely to occur in this paradigm, and if not, precautionary articulatory suppression would not always be necessary. We present evidence confirming that articulatory suppression has no discernible effect on performance in a typical visual change-detection task in which abstract patterns are briefly presented. A comprehensive analysis using both descriptive statistics and Bayesian state-trace analysis revealed no evidence for any complex relationship between articulatory suppression and performance that would be consistent with a verbal recoding explanation. Instead, the evidence favors the simpler explanation that verbal strategies were either not deployed in the task or, if they were, were not effective in improving performance, and thus have no influence on visual working memory as measured during visual change detection. We conclude that in visual change detection experiments in which abstract visual stimuli are briefly presented, pre-cautionary articulatory suppression is unnecessary.

Data

```
source('intNormal.R') # for approximation to integral
# source('http://openmx.psyc.virginia.edu/getOpenMx.R')
library('OpenMx')
```

```
## Warning: package 'OpenMx' was built under R version 3.1.3
```

```
## Loading required package: digest
```

```
## Loading required package: MASS
```

```
## Loading required package: parallel
```

```
library('gtools')
library('xtable')
plotRestricted = TRUE
loadSamples = TRUE
rand.seed = 595
set.seed(rand.seed)
loadSamples = loadSamples * plotRestricted
```

Load and clean the data

The raw data is contained in the `combined_dataset.csv` file. We load the file and create meaningful names for the factors in the data set.

```r
data <- read.csv("combined_dataset.csv", stringsAsFactors = FALSE)

data$silent = factor(data$silent)
levels(data$silent) = c("articulate","silent")

data$sequential = factor(data$sequential)
levels(data$sequential) = c("simultaneous","sequential")

data$change = factor(data$change)
levels(data$change) = c("same","change")


data$subjNumber = factor(data$subjNumber)
data$setSize = factor(data$setSize)

data$blocknum = ((data$session-1)*504 + (data$trial-1))%/%252 + 1
```

We first remove trials for which the response time is missing.

```r
# What proportion of trials?
mean(is.na(data$RT))
```

```
## [1] 0.001045689
```

```r
data = data[!is.na(data$RT),]
```

And now we get rid of short and long response times (short are those faster than 200 ms and long are those slower than 3 seconds).

```r
1-mean(data$RT>200 & data$RT<3000) # What proportion of trials?
```

```
## [1] 0.02045253
```

```r
data = data[data$RT>200 & data$RT<3000, ]
```

Data tidying

We now compute summary statistics for each participant by condition combination.

```r
# Correct trials for all combinations of conditions and participants
corrects = tapply(data$CResp,list(data$subjNumber,data$setSize,data$silent,data$sequential,data$change)

# Total number of trials
Ntotal = table(data$subjNumber,data$setSize,data$silent,data$sequential,data$change)

# Convert tables to data frames for convenience
correctsDF = as.data.frame.table(corrects)
NtotalDF = as.data.frame.table(Ntotal)

# Check to make sure they line up
```

```
if(  all(correctsDF[,1:5] == NtotalDF[,1:5]) ){
  correctsDF$N = NtotalDF$Freq
  colnames(correctsDF) = c("sub","ss","art","seq","chg","cor","N")
}else{
  stop("Could not merge data sets.")
}

# Compute estimates of probabilities and corresponding standard errors
correctsDF$phat = (correctsDF$cor + 1) / (correctsDF$N + 2)
correctsDF$stdErr = sqrt(correctsDF$phat * (1 - correctsDF$phat) / (correctsDF$N + 2))

chg = correctsDF[correctsDF$chg=="change",]
sme = correctsDF[correctsDF$chg=="same",]

# Check to make sure they line up
if(  all(sme[,1:4] == chg[,1:4]) ){
  combinedDat = sme[,1:4]
  # hits minus false alarms
  combinedDat$d = chg$phat + sme$phat - 1
  combinedDat$stdErr = sqrt(sme$stdErr^2 + chg$stdErr^2)
}else{
  stop("Could not merge same and change.")
}
```

Figures and analyses

Figure 1: Experimental paradigm

Figure 1 was created in OmniGraffle Professional. The `.graffle` file as well as the exported `.pdf` are located in `./figures/`.

Table 1: Descriptive statistics for the different conditions

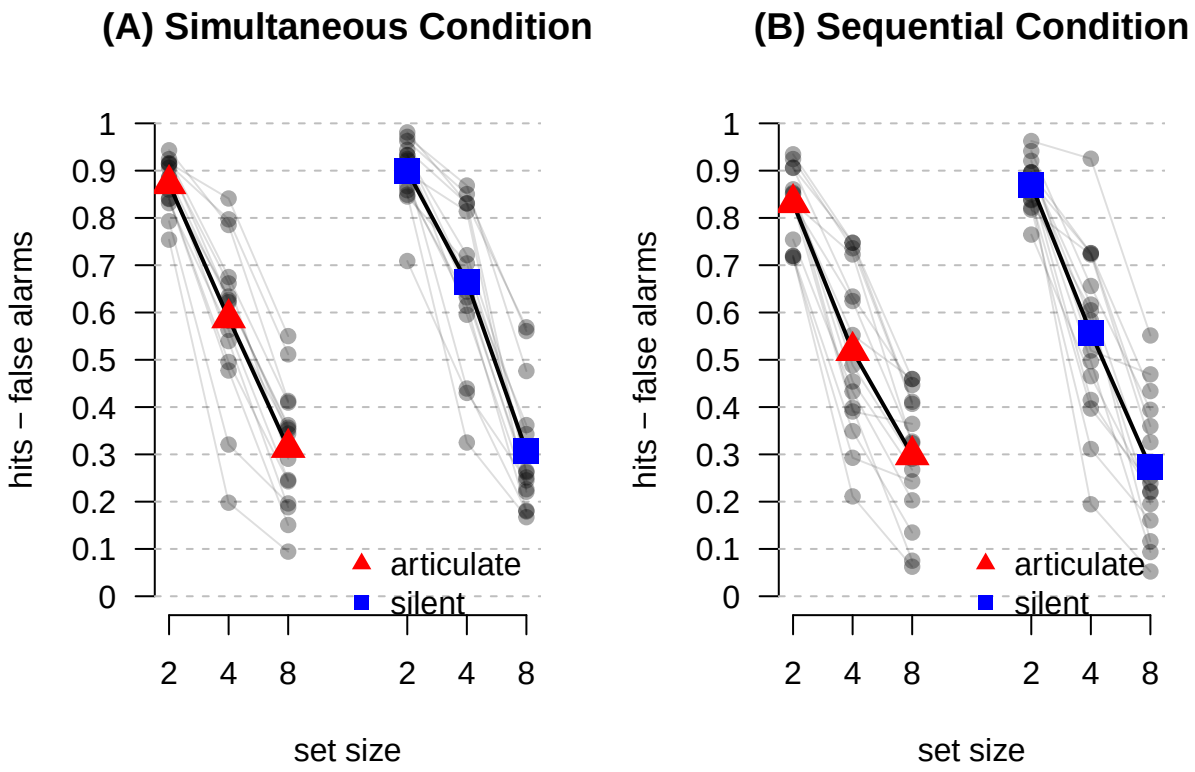
```
chg.mns = with(chg,tapply(phat, list(ss, art, seq), mean))
chg.sds = with(chg,tapply(phat, list(ss, art, seq), sd))

sme.mns = 1-with(sme,tapply(phat, list(ss, art, seq), mean))
sme.sds = with(sme,tapply(phat, list(ss, art, seq), sd))

tbl.data <- cbind( as.data.frame(chg.mns), as.data.frame(sme.mns) )
table1 <- xtable(tbl.data, caption="Mean hit and false alarm rates for all conditions across all partic
print(table1)
```

```
## % latex table generated in R 3.1.2 by xtable 1.7-4 package
## % Thu Mar 31 16:12:49 2016
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrrrrrr}
##   \hline
```

```
##  & articulate.simultaneous & silent.simultaneous & articulate.sequential & silent.sequential & articu
##   \hline
## 2 & 0.95 & 0.95 & 0.94 & 0.94 & 0.08 & 0.06 & 0.11 & 0.07 \\
##   4 & 0.84 & 0.88 & 0.82 & 0.82 & 0.25 & 0.22 & 0.31 & 0.26 \\
##   8 & 0.72 & 0.70 & 0.70 & 0.70 & 0.41 & 0.39 & 0.41 & 0.42 \\
##    \hline
## \end{tabular}
## \caption{Mean hit and false alarm rates for all conditions across all participants.}
## \label{tab:descriptiveStats}
## \end{table}
```

Figure 2: Descriptive statistics for the different conditions

```r
par(mfrow = c( 1, 2), las=1)

for(cond in c('simultaneous', 'sequential')) {

  plot(NA, ylim=c(0, 1), xlim=c(1, 7), ylab='hits - false alarms', xlab='set size', main=paste(ifelse(c
  axis(1, at=c(1:3, 5:7), labels=rep(c(2,4,8), 2), mgp=c(1, 1, 0))
  axis(2, at=seq(0, 1, .1), labels=seq(0, 1, .1))
  abline(h=seq(0, 1, .1), col='gray', lty=2)

  # one line per pp
  for(s in unique(combinedDat$sub)) {
    y1 <- combinedDat$d[combinedDat$sub == s & combinedDat$art == 'articulate' & combinedDat$seq == cond
    y2 <- combinedDat$d[combinedDat$sub == s & combinedDat$art == 'silent' & combinedDat$seq == cond]
    lines(x=c(1:3), y=y1, col='#00000022')
    lines(x=c(5:7), y=y2, col='#00000022')
    points(x=c(1:3, 5:7), y=c(y1, y2), col='#00000055', pch=19)
  }

  # group means as symbols
  gM <- aggregate(d ~ ss + seq + art, combinedDat, mean)
  gM <- gM$d[gM$seq == cond]
  lines(x=c(1:3), y=gM[1:3], col='#000000', lwd=2)
  lines(x=c(5:7), y=gM[4:6], col='#000000', lwd=2)
  points(x=c(1:3, 5:7), y=gM, col=rep(c('#FF0000', '#0000FF'), each=3), pch=rep(c(17, 15), each=3), cex=

  # add a legend
  legend(x=3.75, y=.15, legend=c('articulate', 'silent'), col=c('#FF0000', '#0000FF'), pch=c(17,15), bty
}
```

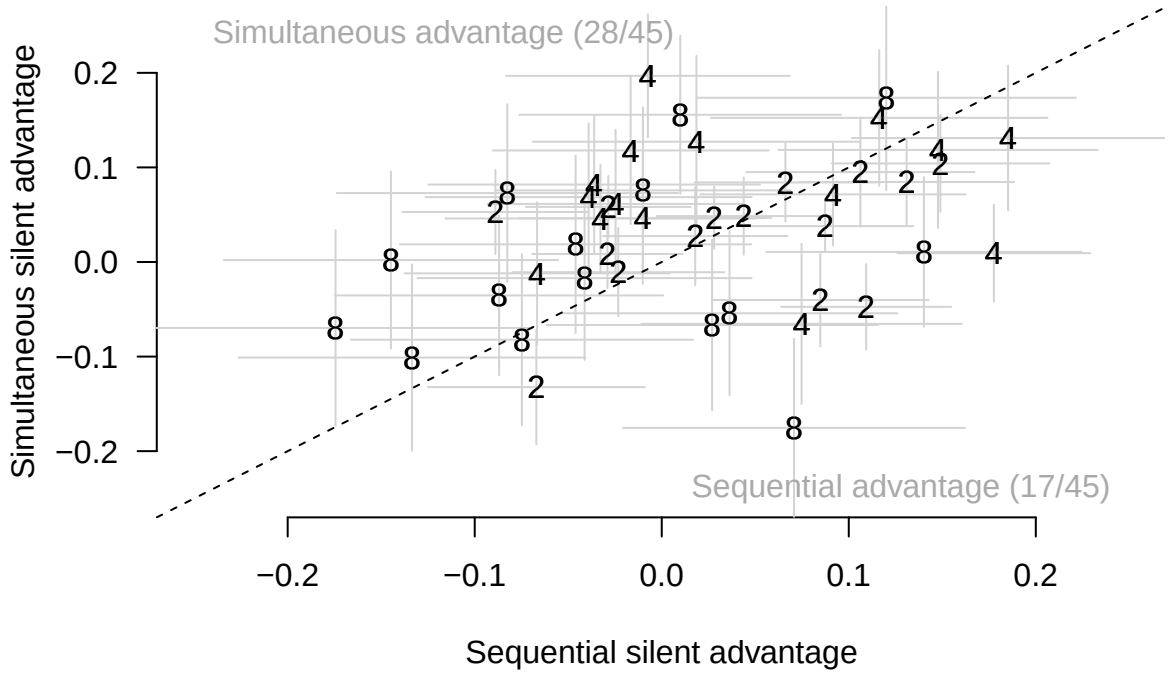


This was used as a template for the Latex syntax of the table but it has been adapted manually quite a bit.

Figure 3A: Sequential silent advantage vs. simultaneous silent advantage

```
eff.est = combinedDat[combinedDat$art=="silent","d"] - combinedDat[combinedDat$art=="articulate","d"]
eff.se = sqrt(combinedDat[combinedDat$art=="silent","stdErr"]^2 + combinedDat[combinedDat$art=="articul

effDF = cbind(combinedDat[combinedDat$art=="silent",-c(3,5,6)],eff = eff.est, stdErr = eff.se)

par(las=1, bty="n")
plot(effDF[effDF$seq == "sequential","eff"], effDF[effDF$seq == "simultaneous","eff"],
     ylim = c(-.25,.25), xlim = c(-.25,.25), ylab = "Simultaneous silent advantage",
     xlab = "Sequential silent advantage", pch = as.character(effDF[effDF$seq == "sequential","ss"]))

segments(effDF[effDF$seq == "sequential","eff"] - effDF[effDF$seq == "sequential","stdErr"],
       effDF[effDF$seq == "simultaneous","eff"],
       effDF[effDF$seq == "sequential","eff"] + effDF[effDF$seq == "sequential","stdErr"],
       effDF[effDF$seq == "simultaneous","eff"],
       col="lightgray")

segments(effDF[effDF$seq == "sequential","eff"],
         effDF[effDF$seq == "simultaneous","eff"] - effDF[effDF$seq == "simultaneous","stdErr"],
         effDF[effDF$seq == "sequential","eff"],
         effDF[effDF$seq == "simultaneous","eff"] + effDF[effDF$seq == "simultaneous","stdErr"],
         col="lightgray")

points(effDF[effDF$seq == "sequential","eff"], effDF[effDF$seq == "simultaneous","eff"],
 pch = as.character(effDF[effDF$seq == "sequential","ss"]))
```

```
abline(0,1, lty=2)
text(-.24,.24,adj=0,"Simultaneous advantage (28/45)", col="darkgray")
text(.24,-.24,adj=1,"Sequential advantage (17/45)", col="darkgray")
```




```
sum((effDF[effDF$seq == "sequential","eff"] - effDF[effDF$seq == "simultaneous","eff"]) > 0)
```

```
## [1] 17
```

```
nrow(effDF[effDF$seq == "sequential",])
```

```
## [1] 45
```

Figure 3B: Target effect across time

```
corr.by.block = with(data,tapply(CResp,list(blocknum,setSize,silent,sequential),mean))
adv.for.silent = corr.by.block[,,"silent",] - corr.by.block[,,"articulate",]
effect.by.pres = adv.for.silent[,,"sequential"] - adv.for.silent[,,"simultaneous"]

par(las=1, bty="n")
matplot(effect.by.pres, ylab="Effect (prop. correct; seq - sim)",
        xlab="Block (2 per session)",ty='b',pch=c("2","4","8"),ylim=c(-.21,.21), col=1)
abline(h=0,col="gray")
text(10,.2,"Silent advantage bigger for sequential presentation",adj=1, col="darkgray")
text(10,-.2,"Silent advantage bigger for simultaneous presentation",adj=1, col="darkgray")
```

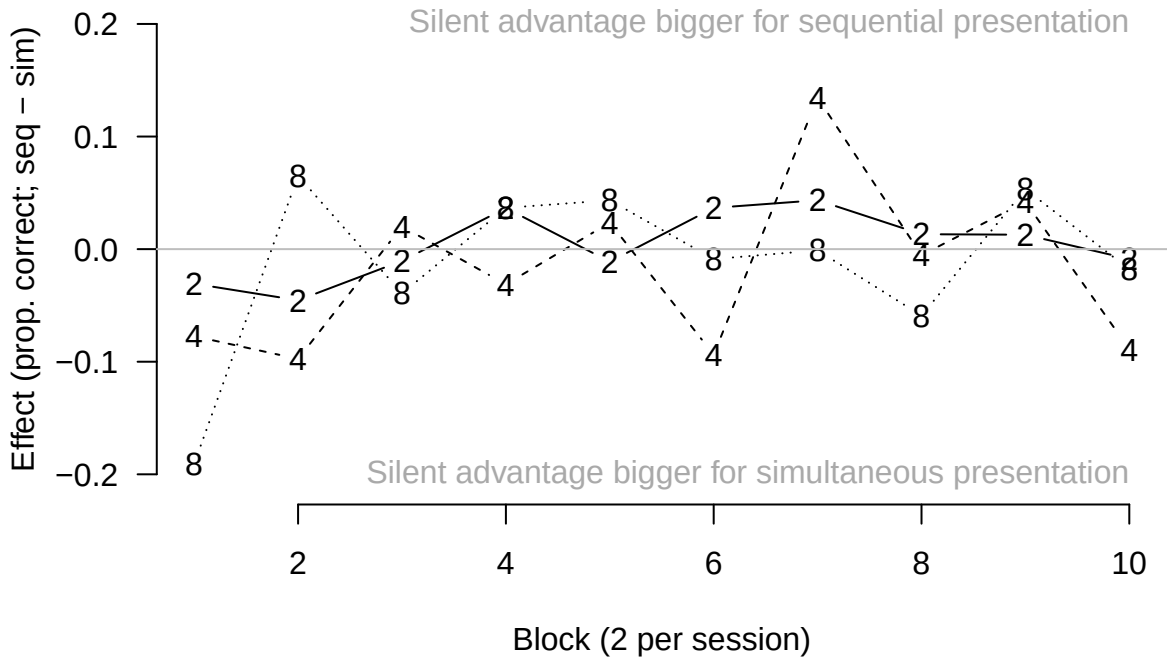


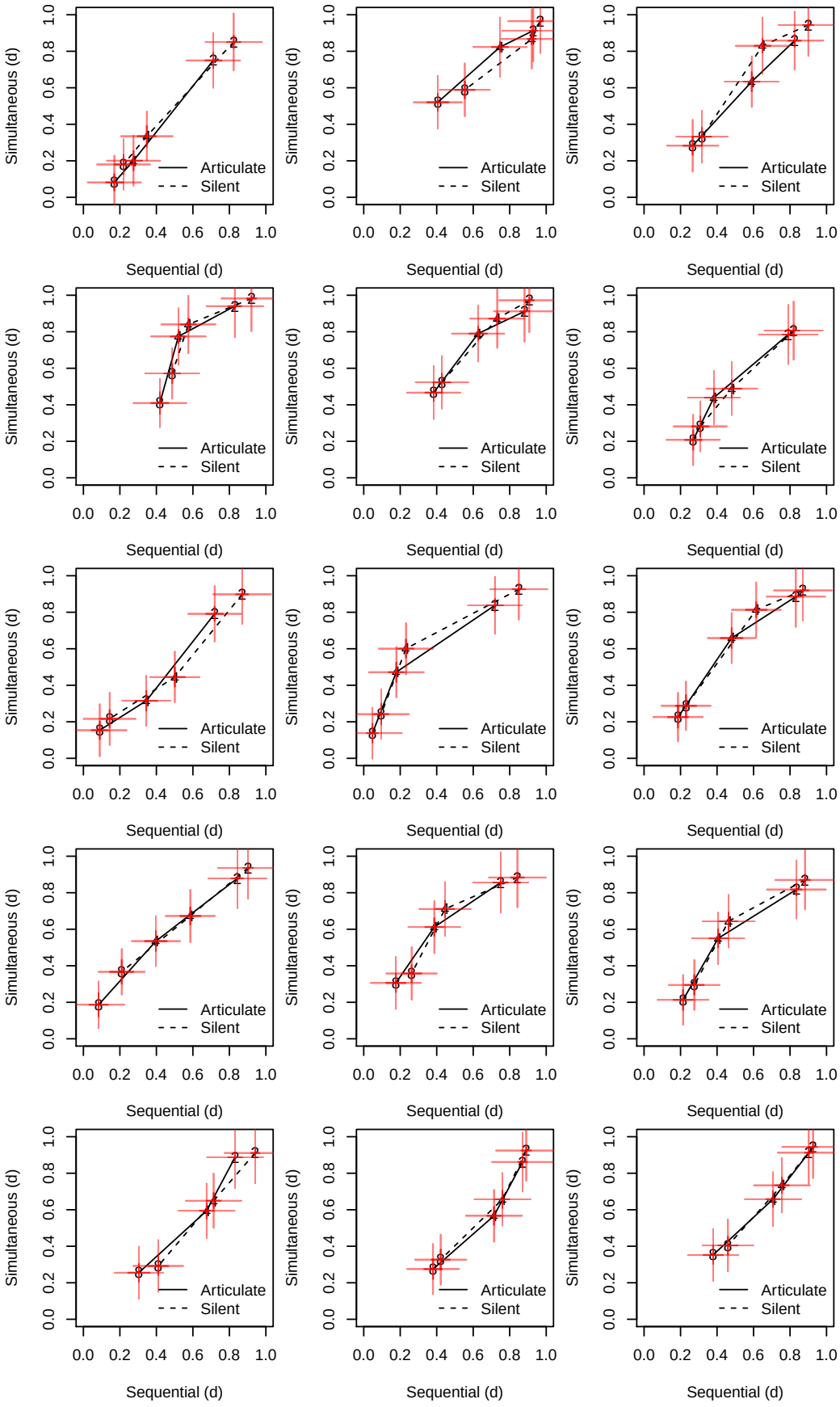
Figure 4: Individual state-trace plots

Make plots and compute Bayes factors.

```
# Reserve object for Bayes factors
BF = 1:length(unique(combinedDat$sub))
names(BF) = unique(combinedDat$sub)
bf.ord = order(read.table(file="BFs.txt"))

subs = unique(combinedDat$sub)[bf.ord]

par(mfrow=c(5,3), mar=c(4,4,.5,.5))

sub.samples = list()
if(loadSamples) load("samples.Rda")

# Do each subject analysis separately
for(sub in subs){
  datSub = combinedDat[combinedDat$sub==sub,]

  # set up for normal approximation to posterior

  # means
  simMu = datSub$d[1:6]
  seqMu = datSub$d[1:6 + 6]

  # standard deviations
  simStdErr = datSub$stdErr[1:6]
  seqStdErr = datSub$stdErr[1:6 + 6]

  # labels (for order restrictions)
```

```
simLab = paste(datSub$ss[1:6],datSub$art[1:6],sep=".")
seqLab = paste(datSub$ss[1:6 + 6],datSub$art[1:6 + 6],sep=".")

# Determine all permutations
perms = permutations(6,6)
# Restrict to permutations that make sense
accOrds = apply(perms,1,checkOrdering,labs=simLab)
ords = perms[accOrds,]

# Compute probabilities of orderings for seq and simulat
probsSim = apply(ords,1,post.prob.order,mus=simMu,sig2=simStdErr^2)
probsSeq = apply(ords,1,post.prob.order,mus=seqMu,sig2=seqStdErr^2)
names(probsSim) = names(probsSeq)= apply(ords,1,paste,collapse=',')

# Renormalize
probsSim = probsSim / sum(probsSim)
probsSeq = probsSeq / sum(probsSeq)

# Assume independence
jointOrderProbs = outer(probsSim,probsSeq)

# ignore non overlapping model
jointOrderProbs[1,1] = NA
jointOrderProbs = jointOrderProbs/sum(jointOrderProbs,na.rm=TRUE)

#prob monotone (diagonal)
probMono = sum(diag(jointOrderProbs),na.rm=TRUE)
probNonMono = 1 - probMono

priorProbMono = (dim(jointOrderProbs)[1]-1) / (length(jointOrderProbs)-1)


# Bayes factor is posterior odds over prior odds
BF[sub] = (probMono / probNonMono) / (priorProbMono / (1-priorProbMono))



if(loadSamples){
  samples.sim = sub.samples[[sub]][["sim"]]
  samples.seq = sub.samples[[sub]][["seq"]]
  }else if(plotRestricted){
    sufficient = FALSE
    samples.sim = NULL
    while(!sufficient){
      samples.sim = rbind(
        samples.sim,
        restrict.samples.mean(simMu,simStdErr,simLab,M=10000,articRestrict=TRUE))
      if(nrow(samples.sim)>5000) sufficient = TRUE
      }

    sufficient = FALSE
    samples.seq = NULL
    while(!sufficient){
```

```
      samples.seq = rbind(
        samples.seq,
        restrict.samples.mean(seqMu,seqStdErr,simLab,M=10000,articRestrict=TRUE))
      if(nrow(samples.seq)>5000) sufficient = TRUE
      }
    }

if(plotRestricted){
  my.d = c(colMeans(samples.sim),colMeans(samples.seq))
  my.serr = c(apply(samples.sim,2,sd),apply(samples.seq,2,sd))
} else {
  my.d = datSub$d
  my.serr = datSub$stdErr
}


# Plot
plot(my.d[1:6 + 6],my.d[1:6],pch=as.character(datSub$ss[1:6]),main="",xlab="Sequential (d)",ylab="Simu
lines(my.d[1:3 + 6],my.d[1:3],lty=1)
lines(my.d[1:3 + 3 + 6],my.d[1:3 + 3],lty=2)
arrows(my.d[1:6 + 6]-my.serr[1:6 + 6],my.d[1:6],my.d[1:6 + 6] + my.serr[1:6 + 6], my.d[1:6],code=3,ang
arrows(my.d[1:6 + 6],my.d[1:6]-my.serr[1:6],my.d[1:6 + 6], my.d[1:6]+my.serr[1:6],code=3,angle=0,col=

legend("bottomright",legend=c("Articulate","Silent"),lty=1:2, bty='n')
}
```


Simultaneous (d)
Sequential (d)
Articulate
Silent
0.0
0.2
0.4
0.6
0.8
1.0

```
if(plotRestricted)
    sub.samples[[sub]] = list(seq = samples.seq, sim = samples.sim)

if(plotRestricted)
  save("sub.samples", file="samples.Rda")
```

```
# What are the valid orders?
t(apply(ords,1,function(row,labs) labs[row],labs=simLab))
```

```
##      [,1]           [,2]           [,3]           [,4]
## [1,] "8.articulate" "4.articulate" "2.articulate" "8.silent"
## [2,] "8.articulate" "4.articulate" "8.silent"     "2.articulate"
## [3,] "8.articulate" "4.articulate" "8.silent"     "4.silent"
## [4,] "8.articulate" "8.silent"     "4.articulate" "2.articulate"
## [5,] "8.articulate" "8.silent"     "4.articulate" "4.silent"
##      [,5]           [,6]
## [1,] "4.silent"     "2.silent"
## [2,] "4.silent"     "2.silent"
## [3,] "2.articulate" "2.silent"
## [4,] "4.silent"     "2.silent"
## [5,] "2.articulate" "2.silent"
```

```
# write.table(file="BFs.txt",BF) # this is how the file was created
```

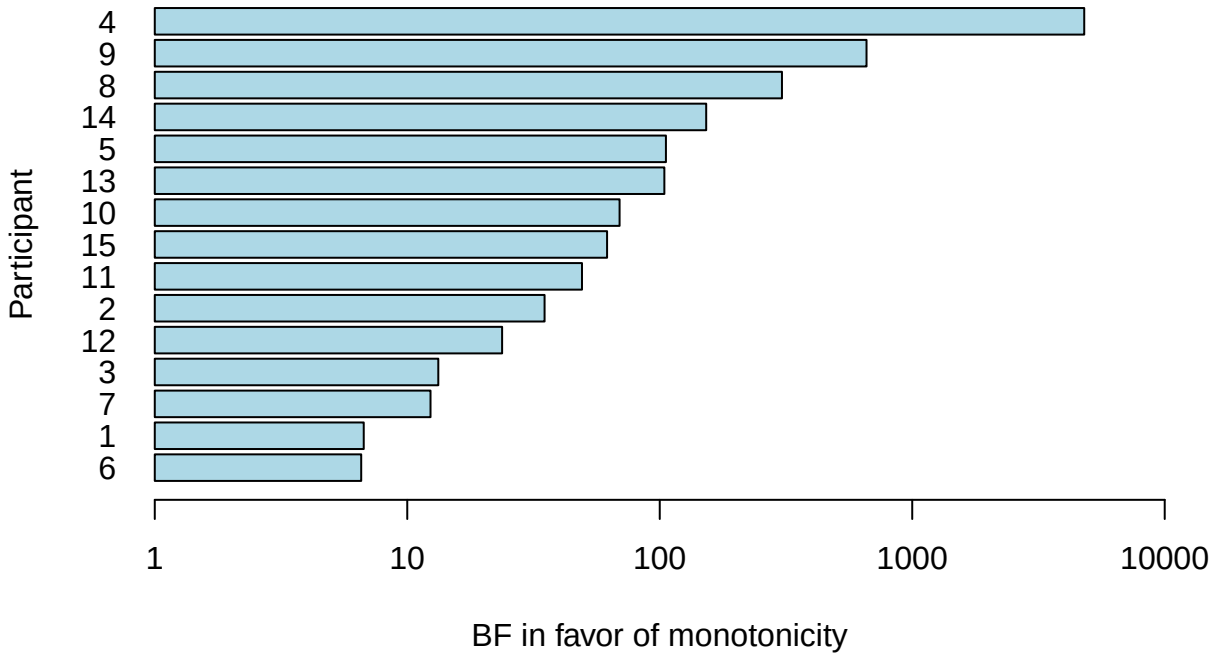
More on Bayes factors and a traditional ANOVA for comparison

More on Bayes factors

It must be noted that the Bayes factors described in this paper are Bayes factors favoring the monotonicity of the observed points. Technically, the null hypothesis in state-trace analysis is that all possible points are so ordered, but testing this hypothesis would require much stronger parametric assumptions. The current approach trades strong assumptions for weaker conclusions.

An additional plot, showing only the Bayes factors (not included in the paper).

```
BF = sort(BF)
newNum = match(names(BF),unique(data$subjNumber))
par(las=1)
q = barplot(BF,horiz=TRUE,axes=FALSE,xlab="BF in favor of monotonicity",ylab="Participant",log="x",name
axis(1)
axis(2,at=q,lab=newNum,tick=FALSE)
```



Traditional ANOVA

To compare our results and conclusions to those obtained with more traditional methods, we include a conventional repeated measures ANOVA here:

```
summary(aov(d ~ art * seq * ss + Error(sub / (art * seq * ss)), combinedDat))
```

```
##
## Error: sub
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 14  1.816  0.1297
##
## Error: sub:art
##           Df  Sum Sq  Mean Sq F value Pr(>F)
## art        1 0.02815 0.028151   4.232 0.0588 .
## Residuals 14 0.09313 0.006652
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sub:seq
##           Df Sum Sq Mean Sq F value Pr(>F)
## seq        1 0.1099 0.10987   6.523 0.0229 *
## Residuals 14 0.2358 0.01684
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sub:ss
##           Df Sum Sq Mean Sq F value Pr(>F)
## ss         2  9.755   4.877   271.8 <2e-16 ***
## Residuals 28  0.502   0.018
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sub:art:seq
##           Df  Sum Sq  Mean Sq F value Pr(>F)
## art:seq    1 0.00274 0.002742   0.938  0.349
## Residuals 14 0.04092 0.002923
##
## Error: sub:art:ss
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## art:ss     2 0.04240 0.021198   7.499 0.00247 **
## Residuals 28 0.07915 0.002827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sub:seq:ss
##           Df  Sum Sq  Mean Sq F value Pr(>F)
## seq:ss     2 0.03457 0.017286   3.519 0.0433 *
## Residuals 28 0.13753 0.004912
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: sub:art:seq:ss
##            Df  Sum Sq  Mean Sq F value Pr(>F)
## art:seq:ss  2 0.00470 0.002350   1.057  0.361
## Residuals  28 0.06227 0.002224
```

Session information

For maximum reproducibility.

```
print(sessionInfo(), locale = FALSE)
```

```
## R version 3.1.2 (2014-10-31)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] xtable_1.7-4 gtools_3.4.1 OpenMx_2.2.4 MASS_7.3-35  digest_0.6.8
##
## loaded via a namespace (and not attached):
## [1] evaluate_0.8    formatR_1.0     htmltools_0.2.6 knitr_1.12
## [5] rmarkdown_0.9.2 stringr_0.6.2   tools_3.1.2     yaml_2.1.13
```