

Supplementary Material

1 R CODE

```

1  set.seed(1234)
2  library('MCMCpack'); library("CDM"); library("dina"); library(coda)
3  J <- 30; iteration <- 20000; K <- 5; C <- 2^K
4  Rcpp::sourceCpp('update_alpha.cpp')
5  q.matrix <- matrix(0, J, K)
6  diag(q.matrix[1:5,]) <- 1
7  diag(q.matrix[6:10,]) <- 1
8  q.matrix[c(11:14, 21:26), 1] <- 1
9  q.matrix[c(11, 15:17, 21:23, 27:29), 2] <- 1
10 q.matrix[c(12, 15, 18, 19, 21, 24, 25, 27, 28, 30), 3] <- 1
11 q.matrix[c(13, 16, 18, 20, 22, 24, 26, 27, 29, 30), 4] <- 1
12 q.matrix[c(14, 17, 19, 20, 23, 25, 26, 28:30), 5] <- 1
13 slip <- rep(0.1, J); guess <- rep(0.1, J);
14 K <- ncol(q.matrix);
15 CC <- 2^K;
16 burnin <- 10000
17 alpha <- matrix(rep(0, K * CC), ncol = K)
18 h1 <- 2
19 if (K >= 2) {
20   for (l1l in 1:(K-1)) {
21     lk <- combn(1:K, l1l)
22     for (jjj in 1:(ncol(lk))) {
23       alpha[h1, lk[, jjj]] <- 1
24       h1 <- h1 + 1
25     }
26   }
27 }
28 alpha[CC, ] <- rep(1, K)
29 set.seed(1)
30 I <- 500; alpha0 <- matrix(0, I, K);
31 alpha0[1:I, ] <- alpha[sample(1:32, I, replace=TRUE), ];
32 RR <- sim.din(q.matrix = q.matrix, guess=guess, slip=slip,
33             rule = "DINA", alpha=alpha0);
34 response <- RR$dat
35 slice <- function(response, q.matrix, iteration, burnin){
36   K <- ncol(q.matrix);
37   I <- nrow(response);
38   J <- ncol(response);
39   CC <- 2^K;
40   S <- matrix(0, nrow=iteration+1, ncol=J);

```

```

41 G<- matrix(0,nrow=iteration+1,ncol=J);
42 p<-matrix(0,nrow=iteration+1,ncol=CC)
43 S[1,] <-rep(runif(1,0,0.2),J);G[1,] <-rep(runif(1,0,0.2),J);
44 alpha <- matrix(rep(0, K * CC), ncol = K)
45 h1 <- 2
46 if (K >= 2) {
47   for (lll in 1:(K- 1)) {
48     lk <- combn(1:K, lll)
49     for (jjj in 1:(ncol(lk))) {
50       alpha[h1, lk[, jjj]] <- 1
51       h1 <- h1 + 1
52     }
53   }
54 }
55 alpha[CC, ] <- rep(1, K)
56 ness.attr=rowSums(q.matrix)
57 p[1,] <- rep(1/CC,CC)
58 delta<-rep(1,CC);A<-alpha[sample(1:CC,I,replace=TRUE),];
59 for(t in 1:iteration){
60   update_a<-update_alpha(Amat=alpha,Q=q.matrix,ss=S[t,],
61   gs=G[t,],Y=response,PIs=p[t,],ALPHAS=A,delta0=delta)
62   p[t+1,]<-update_a$PIs_new
63   cc <- update_a$CLASSES+1
64   A<-alpha[cc,]
65   poss.attr <- A%%t(q.matrix)
66   eta.pp=poss.attr >=outer(rep(1,I),ness.attr)
67   ETA= 1*(eta.pp=="TRUE")+0*(eta.pp=="FALSE")
68   U1<-runif(I*J,0,t(outer(1-S[t,],rep(1,I))));
69   U2<-runif(I*J,0,t(outer(S[t,],rep(1,I))));
70   P1 <- 1-U1*response*ETA;P2<-U2*(1-response)*ETA
71   c <- apply(P1,2,min)
72   s_right <- apply(rbind(c,1-G[t,]),2,min)
73   s_left <- apply(P2,2,max)
74   S[t+1,] <- runif(J,s_left,s_right)
75   V1<-runif(I*J,0,t(outer(1-G[t,],rep(1,I))));
76   V2<-runif(I*J,0,t(outer(G[t,],rep(1,I))));
77   P1 <- 1-V1*(1-response)*(1-ETA);P2<-V2*response*(1-ETA)
78   c <- apply(P1,2,min)
79   g_right <- apply(rbind(c,1-S[t+1,]),2,min)
80   g_left <- apply(P2,2,max)
81   G[t+1,] <- runif(J,g_left,g_right)
82 }
83
84 item_est <- matrix(0,2,J)
85 item_est[1,]<- c(colMeans(S[burnin:iteration,]))

```

```

86 item_est[2,]<- c(colMeans(G[ burnin:iteration ,]))
87 rownames(item_est)<-c("slip","guessing")
88 colnames(item_est) <- 1:J
89 pi_est<- colMeans(p[ burnin:iteration ,])
90 names(pi_est) <- 1:CC
91 par_est <- list(item_est, pi_est)
92 names(par_est) <- c("item_est", "pi_est")
93 return(par_est)
94 }
95 set.seed(1234)
96 result <- slice(response,q.matrix,iteration,burnin)
97 item_result <- result$item_est;
98 pi_result <- result$pi_est;

```

```

1 // [[Rcpp::depends(RcppArmadillo,rgen,simcdm)]]
2 #include <RcppArmadillo.h>
3 #include <rgen.h>
4 #include <simcdm.h>
5
6 // [[Rcpp::export]]
7 Rcpp::List update_alpha(const arma::mat &Amat,
8                        const arma::mat &Q,
9                        const arma::vec &ss,
10                       const arma::vec &gs,
11                       const arma::mat &Y,
12                       const arma::vec &PIs,
13                       arma::mat &ALPHAS,
14                       const arma::vec &delta0)
15 {
16
17   unsigned int N = Y.n_rows;
18   unsigned int J = Q.n_rows;
19   unsigned int C = Amat.n_rows;
20   unsigned int K = Q.n_cols;
21   double ci;
22
23   arma::mat alphas_new = arma::zeros<arma::mat>(N, K);
24   arma::vec PYCS(C);
25   arma::vec CLASSES(N);
26   arma::vec PS;
27   arma::vec Ncs = arma::zeros<arma::vec>(C);
28   double etaij;
29
30   for (unsigned int i = 0; i < N; i++) {
31     PYCS = arma::ones<arma::vec>(C);

```

```

32
33     for (unsigned int c = 0; c < C; c++) {
34
35         for (unsigned int j = 0; j < J; j++) {
36             etaij = 1.0;
37             if (arma::dot(Amat.row(c), Q.row(j)) <
38                 arma::dot(Q.row(j), Q.row(j))) {
39                 etaij = 0.0;
40             }
41             if (etaij == 1.0 and Y(i, j) == 1.0) {
42                 PYCS(c) = (1.0 - ss(j)) * PYCS(c);
43             }
44             if (etaij == 0.0 and Y(i, j) == 1.0) {
45                 PYCS(c) = gs(j) * PYCS(c);
46             }
47             if (etaij == 1.0 and Y(i, j) == 0.0) {
48                 PYCS(c) = ss(j) * PYCS(c);
49             } else {
50                 PYCS(c) = (1.0 - gs(j)) * PYCS(c);
51             }
52         }
53     }
54     PS = PYCS % PIs
55     / (arma::conv_to<double>::from(PYCS.t() * PIs));
56     ci = rgen::rmultinomial(PS);
57     ALPHAS.row(i) = Amat.row(ci);
58     Ncs(ci) = 1.0 + Ncs(ci);
59     CLASSES(i) = ci;
60 }
61 return Rcpp::List::create(
62     Rcpp::Named("PYCS") = PYCS, Rcpp::Named("PS") = PS,
63     Rcpp::Named("PIs_new") = rgen::rdirichlet(Ncs + delta0),
64     Rcpp::Named("CLASSES") = CLASSES);
65 }

```