Example: Computing positive definite (PD) bounds for a correlation element inside of a structured correlation matrix

Michael K. Kim, Michael J. Daniels

2025-05-21

BEFORE MCMC

(setting all this up should help save computation time during MCMC)

For our unbalanced longitudinal data with missingness, a subject can have a maximum of J time points and a maximum of L measurements at each time point. For example, for $J = 3$ and $L = 4$, subject i can have the following outcomes: $\boldsymbol{y}_i = (y_{ijl})' = (y_{i11}, y_{i12}, y_{i13}, y_{14}, \quad y_{i21}, y_{i22}, y_{i23}, y_{i24}, \quad y_{i31}, y_{i32}, y_{i33}, y_{i34})'$.

```
J <- 3 # maximum number of time points for a given subject
L <- 4 # maximum number of measurements at a given time point for a given subject
p_max <- J*L # maximum total number of outcomes
```

```
(j_order0_max <- rep(1:J, each=L))
```

```
##  [1] 1 1 1 1 2 2 2 2 3 3 3 3
```

```
(l_order0_max <- rep(1:L, times=J))
```

```
##  [1] 1 2 3 4 1 2 3 4 1 2 3 4
```

Construct a structured correlation matrix using strings.

```
R_order_latex <- function(p, js, ls, matrix_format="upper") {
  matrix_structure <- diag(nrow=p,ncol=p)
  for (m in 1:p) {
    for (n in 1:p) {
      row_j <- js[m]
      row_l <- ls[m]
      col_j <- js[n]
      col_l <- ls[n]
      if ((row_j==col_j) && (row_l!=col_l)) {
        smaller_l <- min(row_l, col_l)
        bigger_l <- max(row_l, col_l)
        matrix_structure[m,n] <- paste0("eta_{",smaller_l,bigger_l,"}")
      } else if ((row_j!=col_j) && (row_l==col_l)) {
        the_l <- row_l
        matrix_structure[m,n] <- paste0("rho_{(",the_l,")}")
      } else if ((row_j!=col_j) && (row_l!=col_l)) {
```

```
        matrix_structure[m,n] <- paste0("gamma")
      }
    }
  }

  if (matrix_format=="all") {
    return(matrix_structure)
  } else if (matrix_format=="upper") {
    upper_matrix_structure <- matrix_structure
    upper_matrix_structure[lower.tri(matrix_structure)] <- NA
    return(upper_matrix_structure)
  } else if (matrix_format=="lower") {
    lower_matrix_structure <- matrix_structure
    lower_matrix_structure[upper.tri(matrix_structure)] <- NA
    return(lower_matrix_structure)
  }
}

(R_latex_upper <- R_order_latex(p_max, j_order0_max, l_order0_max, "upper"))
```

```
##       [,1] [,2]       [,3]       [,4]       [,5]         [,6]         [,7]
##  [1,] "1"  "eta_{12}" "eta_{13}" "eta_{14}" "rho_{(1)}" "gamma"      "gamma"
##  [2,] NA   "1"        "eta_{23}" "eta_{24}" "gamma"      "rho_{(2)}" "gamma"
##  [3,] NA   NA         "1"        "eta_{34}" "gamma"      "gamma"      "rho_{(3)}"
##  [4,] NA   NA         NA         "1"        "gamma"      "gamma"      "gamma"
##  [5,] NA   NA         NA         NA         "1"          "eta_{12}"  "eta_{13}"
##  [6,] NA   NA         NA         NA         NA           "1"          "eta_{23}"
##  [7,] NA   NA         NA         NA         NA           NA           "1"
##  [8,] NA   NA         NA         NA         NA           NA           NA
##  [9,] NA   NA         NA         NA         NA           NA           NA
## [10,] NA   NA         NA         NA         NA           NA           NA
## [11,] NA   NA         NA         NA         NA           NA           NA
## [12,] NA   NA         NA         NA         NA           NA           NA
##       [,8]         [,9]         [,10]       [,11]       [,12]
##  [1,] "gamma"      "rho_{(1)}" "gamma"      "gamma"      "gamma"
##  [2,] "gamma"      "gamma"      "rho_{(2)}" "gamma"      "gamma"
##  [3,] "gamma"      "gamma"      "gamma"      "rho_{(3)}" "gamma"
##  [4,] "rho_{(4)}" "gamma"      "gamma"      "gamma"      "rho_{(4)}"
##  [5,] "eta_{14}"  "rho_{(1)}" "gamma"      "gamma"      "gamma"
##  [6,] "eta_{24}"  "gamma"      "rho_{(2)}" "gamma"      "gamma"
##  [7,] "eta_{34}"  "gamma"      "gamma"      "rho_{(3)}" "gamma"
##  [8,] "1"          "gamma"      "gamma"      "gamma"      "rho_{(4)}"
##  [9,] NA           "1"          "eta_{12}"  "eta_{13}"  "eta_{14}"
## [10,] NA           NA           "1"          "eta_{23}"  "eta_{24}"
## [11,] NA           NA           NA           "1"          "eta_{34}"
## [12,] NA           NA           NA           NA           "1"
```

Construct the vector of the unique correlation elements using strings.

```
n_etas <- choose(L,2)
muscle_combns <- combn(1:L,2)
eta_names <- sapply(1:n_etas, function(combn_indx)
  paste0("eta_{",muscle_combns[1,combn_indx],muscle_combns[2,combn_indx],"}"))
```

```
rho_names <- paste0("rho_{(",1:L,")}")
r_names <- c(eta_names, rho_names, "gamma")
Q <- length(r_names)
r_names
```

```
##  [1] "eta_{12}"  "eta_{13}"  "eta_{14}"  "eta_{23}"  "eta_{24}"  "eta_{34}"
##  [7] "rho_{(1)}" "rho_{(2)}" "rho_{(3)}" "rho_{(4)}" "gamma"
```

Positions of first occurrences of the unique correlation elements (pivot elements) inside the structured correlation matrix:

```
pivot_and_implied_locs <- function(R_latex_upper, r_names) {
  # transpose lets us read pivot elements by row-by-row
  unique_elems_R <- unique(c(t(R_latex_upper)))
  unique_elems_R <- unique_elems_R[!unique_elems_R %in% NA]
  unique_elems_R <- unique_elems_R[!unique_elems_R %in% "1"]
  r_order <- match(r_names, unique_elems_R)
  unique_elems_R <- unique_elems_R[r_order]

  # 'pivot_locs' listed in same order as in 'r_names'
  pivot_locs <- matrix(nrow=length(unique_elems_R),ncol=2)
  implied_locs <- list()
  for (u in 1:length(unique_elems_R)) {
    elem_locs <- which(R_latex_upper==unique_elems_R[u],arr.ind=TRUE)
    if (nrow(elem_locs)>1) {
      # sort s.t. first row of matrix = pivot location
      elem_locs <- elem_locs[order(elem_locs[,1],elem_locs[,2],decreasing=FALSE),]
    }
    # automatically stores as row in 'pivot_locs' matrix
    pivot_locs[u,] <- elem_locs[1,]
    # coerce that every list's element = matrix form
    if (is.vector(elem_locs[-1,])) {
      # case 1: only one implied location => turn vector into matrix form
      implied_locs[[u]] <- t(as.matrix(elem_locs[-1,]))
    } else {
      # case 2: 0 implied location => already matrix form
      # case 3: >2 implied locations => already matrix form
      implied_locs[[u]] <- elem_locs[-1,]
    }
  }

  return(list(pivot_locs, implied_locs))
}

pivot_implied_locs <- pivot_and_implied_locs(R_latex_upper, r_names)
pivot_locs <- pivot_implied_locs[[1]]
pivot_locs_present <- pivot_locs
rownames(pivot_locs_present) <- r_names
pivot_locs_present
```

```
##           [,1] [,2]
## eta_{12}     1    2
## eta_{13}     1    3
## eta_{14}     1    4
```

```
## eta_{23}     2    3
## eta_{24}     2    4
## eta_{34}     3    4
## rho_{(1)}    1    5
## rho_{(2)}    2    6
## rho_{(3)}    3    7
## rho_{(4)}    4    8
## gamma        1    6
```

Re-order the indices of $1:p$ outcomes such that the largest possible submatrix inside the re-ordered correlation matrix contains the unique kth correlation element only once. This is to set up for the "Barnard method" from Barnard, McCulloch, Meng (2000).

Note: the following function only acquires one variation of the largest possible submatrix, but the upside is that the logic inside the function is applicable for *any* structured correlation matrix with some minor tweaks. This should allow a decent approximation of the PD interval of a correlation element, especially if the dimensions of the largest possible submatrix is close to that of the correlation matrix.

```
biggest_submatrix_indxs <- function(r_names, pivot_locs, k,
                                    p, js, ls) {
  goal_corr <- r_names[k]
  goal_indx1 <- pivot_locs[k,1]
  goal_indx2 <- pivot_locs[k,2]

  subblock_indxs <- c(goal_indx1, goal_indx2)
  all_indxs <- 1:p
  indxs_to_go_through <- all_indxs[-subblock_indxs]
  for (m in indxs_to_go_through) {
    js_possible <- js[c(subblock_indxs,m)]
    ls_possible <- ls[c(subblock_indxs,m)]

    R_possible <- R_order_latex(length(js_possible), js_possible, ls_possible, "upper")
    goal_corr_freq <- nrow(which(R_possible==goal_corr, arr.ind=TRUE))
    if (goal_corr_freq==1) {
      subblock_indxs <- append(subblock_indxs,m)
    }
  }
  return(subblock_indxs)
}

subblock_indxs_list <- lapply(1:Q, function(k) {
  biggest_submatrix_indxs(r_names, pivot_locs, k,
                          p_max, j_order0_max, l_order0_max) })
subblock_indxs_list_present <- subblock_indxs_list
names(subblock_indxs_list_present) <- r_names
subblock_indxs_list_present
```

```
## $`eta_{12}`
##  [1]  1  2  3  4  5  7  8  9 11 12
##
## $`eta_{13}`
##  [1]  1  3  2  4  5  6  8  9 10 12
##
## $`eta_{14}`
##  [1]  1  4  2  3  5  6  7  9 10 11
```

```
##
## $`eta_{23}`
##  [1]  2  3  1  4  5  6  8  9 10 12
##
## $`eta_{24}`
##  [1]  2  4  1  3  5  6  7  9 10 11
##
## $`eta_{34}`
##  [1]  3  4  1  2  5  6  7  9 10 11
##
## $`rho_{(1)}`
##  [1]  1  5  2  3  4  6  7  8 10 11 12
##
## $`rho_{(2)}`
##  [1]  2  6  1  3  4  5  7  8  9 11 12
##
## $`rho_{(3)}`
##  [1]  3  7  1  2  4  5  6  8  9 10 12
##
## $`rho_{(4)}`
##  [1]  4  8  1  2  3  5  6  7  9 10 11
##
## $gamma
## [1] 1 6 2
```

OTHER HELPFUL FUNCTIONS AND RESULTS BEFORE MCMC

"Barnard method" from Barnard, McCulloch, Meng (2000) (needed for MCMC to compute PD interval)

```
barnard <- function(i,j,R) {
  R_1 <- R; R_m1 <- R; R_0 <- R
  R_1[i,j] <- 1; R_1[j,i] <- 1
  R_m1[i,j] <- -1; R_m1[j,i] <- -1
  R_0[i,j] <- 0; R_0[j,i] <- 0
  det.R_1 <- det(R_1)
  det.R_m1 <- det(R_m1)
  det.R_0 <- det(R_0)

  a <- 0.5*(det.R_1 + det.R_m1 - 2*det.R_0)
  b <- 0.5*(det.R_1 - det.R_m1)
  c <- det.R_0
  sqrt.b24ac <- sqrt(b^2-4*a*c)
  two_times_a <- 2*a

  r1 <- (-b - sqrt.b24ac)/(two_times_a)
  r2 <- (-b + sqrt.b24ac)/(two_times_a)
```

```
  root1 <- min(r1,r2)
  root2 <- max(r1,r2)

  interval <- c(root1,root2)
  return(interval)
}
```

Positions of all occurrences of the unique correlation elements inside the structured correlation matrix (whenever there's a change in the value of a specific correlation element, this helps efficiently update the structured correlation matrix during the actual (not our current toy example) run of MCMC):

```
uniqueR_locs_func <- function(R_latex_something, r_names) {
  unique_elems_R <- r_names
  uniqueR_locs <- list()
  for (u in 1:length(unique_elems_R)) {
    elem_locs <- which(R_latex_something==unique_elems_R[u],arr.ind=TRUE)
    if (nrow(elem_locs)>1) {
      # sort
      elem_locs <- elem_locs[order(elem_locs[,1],elem_locs[,2],decreasing=FALSE),]
    }
    uniqueR_locs[[u]] <- elem_locs
  }
  return(uniqueR_locs)
}

R_latex_all <- R_latex_upper
R_latex_all[lower.tri(R_latex_all)] <- t(R_latex_all)[lower.tri(R_latex_all)]
uniqueR_locs_all <- uniqueR_locs_func(R_latex_all, r_names)
#uniqueR_locs_all_present <- uniqueR_locs_all
#names(uniqueR_locs_all_present) <- r_names
#uniqueR_locs_all_present
```

Transform a vector of the unique correlation elements into the structured correlation matrix (function isn't needed for the actual (not our current toy example) run of MCMC, since it would be computationally inefficient, but the function is still useful when doing testing/examples. In our toy example, we will just use this function to compute a structured a correlation matrix with some numerical values.)

```
rtoR <- function(r_vec, Q, I_mat, uniqueR_locs) {
  for (k in 1:Q) {
    I_mat[uniqueR_locs[[k]]] <- r_vec[k]
  }
  return(I_mat) # really returning the R_mat
}

# MODIFY THIS BASED ON YOUR PC!
personal_directory <-
  "/Users/michaelkim/Desktop/Research/mvn_discrete_time/github_share/"
target_parameters <- readRDS(paste0(personal_directory,
                                    "/y_sim_1/target/Early_4L_post_med.RDS"))
R_old <- rtoR(target_parameters$target_r, Q, diag(p_max), uniqueR_locs_all)
#all(eigen(R_old)$values > 0) # confirm positive definiteness
```

```
#isSymmetric(R_old) # confirm symmetry
#all(diag(R_old) == 1) # confirm diagonal elements = 1
```

DURING MCMC

(the computationally heavy part)

```
PD_interval <- vector("list", length=Q)
for (k in 1:Q) {
  indxs_temp <- subblock_indxs_list[[k]]
  R_old_temp <- R_old[indxs_temp, indxs_temp]
  PD_interval[[k]] <- barnard(1,2,R_old_temp)
}
PD_interval # tighter than (-1,1)
```

```
## [[1]]
## [1] -0.5268433  0.9971946
##
## [[2]]
## [1] -0.3977662  0.9207561
##
## [[3]]
## [1] -0.3585117  0.9329113
##
## [[4]]
## [1] -0.4268613  0.8958582
##
## [[5]]
## [1] -0.3120599  0.9550387
##
## [[6]]
## [1] -0.5056557  0.9455540
##
## [[7]]
## [1] -0.3412577  0.8306434
##
## [[8]]
## [1] -0.3381096  0.8456905
##
## [[9]]
## [1] -0.5287141  0.9342741
##
## [[10]]
## [1] -0.4137435  0.8840938
##
## [[11]]
## [1] -0.6274472  0.9270638
```