Quasi-Bayesian Inference for Multilevel Mediation Analysis

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

```
library(MASS)

expit = function(x){
  return(1/(1 + exp(-x)))
}

logit = function(x){
  return(log(x/(1-x)))
```

```r
}
```

```r
# Number of times to simulate GLMM coefficients
num_reps = 50

# Model parameters

n = 200 # Sample size in each group
K = 5   # Number of groups
p_conf = 3  # Number of confounders

## Fixed effects
a_0 = 0
a_1 = 1
A_2 = rep(1, times = p_conf)

b_0 = -0.5
b_1 = 1
b_2 = 1
B_3 = rep(1, times = p_conf)

## Random effects
sigma_a_0 = 0.2
sigma_a_1 = 0.2 * abs(a_1)
cor_a0_a1 = 0.2
cov_a0_a1 = sigma_a_0 * sigma_a_1 * cor_a0_a1
Sigma_a = matrix(c(sigma_a_0^2, cov_a0_a1, cov_a0_a1, sigma_a_1^2), nrow = 2, ncol = 2)

sigma_b_0 = 0.2 * abs(b_0)
sigma_b_1 = 0.2 * abs(b_1)
cor_b0_b1 = 0.2
cov_b0_b1 = sigma_b_0 * sigma_b_1 * cor_b0_b1
Sigma_b = matrix(c(sigma_b_0^2, cov_b0_b1, cov_b0_b1, sigma_b_1^2), nrow = 2, ncol = 2)
```

```r
all_Xs = list()
all_Ws = list()

for(k in 1:K){
  X = rnorm(n, mean=0, sd=1)
  W = matrix(rnorm(n*p_conf, mean=0, sd=1), nrow = n, ncol = p_conf)
```

```r
  all_Xs[[k]] = X
  all_Ws[[k]] = W
}

# Generate M
all_Ms = list()
for(k in 1:K){
  eta_vec_fixed = a_0 + a_1*all_Xs[[k]] + all_Ws[[k]]%*%A_2

  ## Add random effects
  a_ran = mvrnorm(1, mu = rep(0, 2), Sigma = Sigma_a)
  eta_vec = eta_vec_fixed + a_ran[1] + a_ran[2]*all_Xs[[k]]

  ## Generate M
  p_M_vec = expit(eta_vec)
  M = rbinom(n, size = 1, prob = p_M_vec)
  all_Ms[[k]] = M
}

# Generate Y
all_Ys = list()
for(k in 1:K){
  zeta_vec_fixed = b_0 + b_1*all_Ms[[k]] + b_2 * all_Xs[[k]] + all_Ws[[k]]%*%B_3

  ## Add random effects
  b_ran = mvrnorm(1, mu = rep(0, 2), Sigma = Sigma_b)
  zeta_vec = zeta_vec_fixed + b_ran[1] + b_ran[2]*all_Xs[[k]]

  ## Generate Y
  p_Y_vec = expit(zeta_vec)
  Y = rbinom(n, size = 1, prob = p_Y_vec)
  all_Ys[[k]] = Y
}


# Consolidate groups
X = do.call(c, all_Xs)
W = do.call(rbind, all_Ws)
M = do.call(c, all_Ms)
Y = do.call(c, all_Ys)
group = rep(1:K, each = n)
```

```r
data = data.frame(Y=Y, M=M, X=X, W1 = W[,1], W2 = W[,2], W3 = W[,3], group = group)
```