Demonstration of the COMBO R Package for Two-Stage Models

Created by Kimberly A. Hochstedler. Contact: kah343@cornell.edu

2023-02-21

In this vignette, we provide a demonstration of the R Package *COMBO* (correcting misclassified binary outcomes) for analyzing two-stage models. This package provides methods for fitting logistic regression models when two sequential binary outcomes are potentially misclassified. Technical details about estimation are not included in this demonstration. For additional information on the methods used in this R Package, please consult "Statistical inference for association studies in the presence of binary outcome misclassification" by Kimberly A. Hochstedler and Martin T. Wells.

Model and Conceptual Framework

Let $Y = j$ denote an observation's true outcome status, taking values $j \in \{1, 2\}$. Suppose we are interested in the relationship between Y and a set of predictors, X, that are correctly measured. This relationship constitutes the *true outcome mechanism*. Let $Y^{*(1)} = k$ be the first-stage observed outcome status, taking values $k \in \{1, 2\}$. Let $Y^{*(2)} = \ell$ be the second-stage observed outcome status, taking values $\ell \in \{1, 2\}$. $Y^{*(1)}$ and $Y^{*(2)}$ are potentially misclassified versions of Y. Let $Z^{(1)}$ and $Z^{(2)}$ denote sets of predictors related to first-stage and second-stage misclassification, respectively. The mechanism that generates the observed outcomes, $Y^{*(1)}$ and $Y^{*(2)}$, given the true outcome, Y, is called the *observation mechanism*. **Figure 1** displays the conceptual model. The following equations express the conceptual process mathematically.

$$\text{True outcome mechanism: } \text{logit}\{P(Y = j|X; \beta)\} = \beta_{j0} + \beta_{jX} X$$

$$\text{First-stage observation mechanism: } \text{logit}\{P(Y^{*(1)} = k|Y = j, Z^{(1)}; \gamma^{(1)})\} = \gamma^{(1)}_{kj0} + \gamma^{(1)}_{kjZ^{(1)}} Z^{(1)}$$

$$\text{Second-stage observation mechanism: } \text{logit}\{P(Y^{*(2)} = \ell|Y^{*(1)} = k, Y = j, Z^{(2)}; \gamma^{(2)})\} = \gamma^{(2)}_{\ell kj0} + \gamma^{(2)}_{\ell kjZ^{(2)}} Z^{(2)}$$

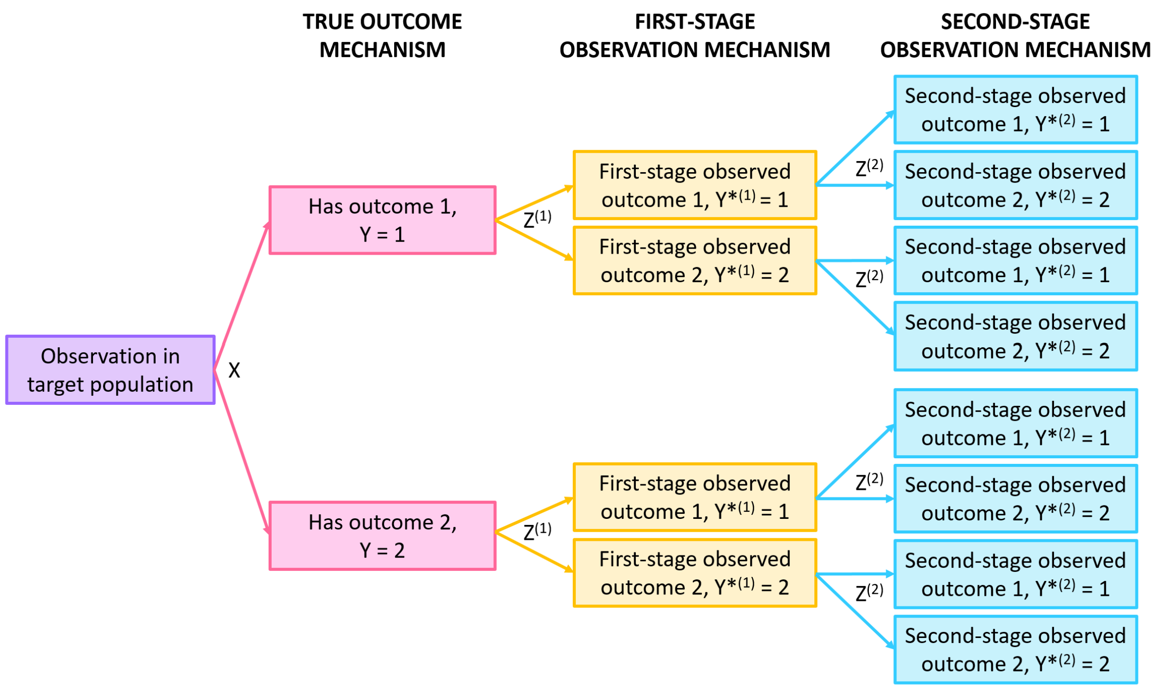

TRUE OUTCOME MECHANISM
FIRST-STAGE OBSERVATION MECHANISM
SECOND-STAGE OBSERVATION MECHANISM
Observation in target population
X
Has outcome 1, Y = 1
Has outcome 2, Y = 2
Z(1)
First-stage observed outcome 1, Y*(1) = 1
First-stage observed outcome 2, Y*(1) = 2
Z(2)
Second-stage observed outcome 1, Y*(2) = 1
Second-stage observed outcome 2, Y*(2) = 2

Figure 1: Conceptual Model

Simulate data

We begin this demonstration by generating data using the `COMBO_data_2stage()` function. The binary outcome data simulated by this scheme is subject to misclassification. The predictor related to the true outcome mechanism is "x", the predictor related to the first-stage observation mechanism is "z", and the predictor related to the second-stage observation mechanism is "v".

```r
library(COMBO)
library(dplyr)
library(stringr)

# Set seed.
set.seed(123)

# Set sample size, x and z distribution information.
n <- 1000
x_mu <- 0
x_sigma <- 1
z_shape <- 1
v_shape <- 1

# Set true parameter values.
true_beta <- matrix(c(1, -2), ncol = 1)
true_gamma <- matrix(c(.5, 1, -.5, -1), nrow = 2, byrow = FALSE)
true_delta <- array(c(1.5, 1, .5, .5, -.5, 0, -1, -1), dim = c(2, 2, 2))

# Generate data.
my_data <- COMBO_data_2stage(sample_size = n,
                              x_mu = x_mu, x_sigma = x_sigma,
                              z_shape = z_shape, v_shape = v_shape,
                              beta = true_beta, gamma = true_gamma,
                              delta = true_delta)

# Save list elements as vectors.
Ystar <- my_data[["obs_Ystar"]]
Ytilde <- my_data[["obs_Ytilde"]]
x_matrix <- my_data[["x"]]
z_matrix <- my_data[["z"]]
v_matrix <- my_data[["v"]]
```

Effect estimation

We propose estimation methods using the Expectation-Maximization algorithm (EM) and Markov Chain Monte Carlo (MCMC). Each method checks and corrects instances of label switching, as described in Hochstedler and Wells (2022). In the code below, we provide functions for implementing these methods.

```
# Supply starting values for all parameters.
starting_values <- rep(1,14)
beta_start <- matrix(starting_values[1:2], ncol = 1)
gamma_start <- matrix(starting_values[3:6], ncol = 2, nrow = 2, byrow = FALSE)
delta_start <- array(starting_values[7:14], dim = c(2,2,2))

# Estimate parameters using the EM-Algorithm.
EM_results <- COMBO_EM_2stage(Ystar, Ytilde,
                              x_matrix = x_matrix, z_matrix = z_matrix,
                              v_matrix = v_matrix,
                              beta_start = beta_start, gamma_start = gamma_start,
                              delta_start = delta_start)

EM_results
```

```
##        Parameter  Estimates        SE Convergence
## 1          beta1  1.4857609 0.3238008        TRUE
## 2          beta2 -2.3265703 0.4681841        TRUE
## 3        gamma11  0.3211518 0.1672749        TRUE
## 4        gamma21  1.1840635 0.2141507        TRUE
## 5        gamma12 -1.1636818 0.3454145        TRUE
## 6        gamma22 -0.7654293 0.2742761        TRUE
## 7      delta1111  1.0949581 0.2177921        TRUE
## 8      delta2111  1.4164513 0.3075869        TRUE
## 9      delta1121  0.1675273 0.6064205        TRUE
## 10     delta2121  1.1181799 0.3890781        TRUE
## 11     delta1112 -1.0898377 1.5147954        TRUE
## 12     delta2112  0.1323470 0.9157495        TRUE
## 13     delta1122 -0.9378235 0.3907864        TRUE
## 14     delta2122 -1.4545354 0.4430074        TRUE
## 15   naive_beta1  2.0395634 0.3586543        TRUE
## 16   naive_beta2 -3.0261792 0.5118091        TRUE
## 17 naive_delta11  0.6391778 0.1795188        TRUE
## 18 naive_delta21  1.1309208 0.2665479        TRUE
## 19 naive_delta12 -0.6268924 0.3503954        TRUE
## 20 naive_delta22 -1.2690082 0.5441009        TRUE
```

```
# Specify parameters for the prior distributions.
unif_lower_beta <- matrix(c(-5, -5, NA, NA), nrow = 2, byrow = TRUE)
unif_upper_beta <- matrix(c(5, 5, NA, NA), nrow = 2, byrow = TRUE)

unif_lower_gamma <- array(data = c(-5, NA, -5, NA, -5, NA, -5, NA),
                          dim = c(2,2,2))
unif_upper_gamma <- array(data = c(5, NA, 5, NA, 5, NA, 5, NA),
                          dim = c(2,2,2))

unif_upper_delta <- array(rep(c(5, NA), 8), dim = c(2,2,2,2))
unif_lower_delta <- array(rep(c(-5, NA), 8), dim = c(2,2,2,2))

beta_prior_parameters <- list(lower = unif_lower_beta, upper = unif_upper_beta)
gamma_prior_parameters <- list(lower = unif_lower_gamma, upper = unif_upper_gamma)
delta_prior_parameters <- list(lower = unif_lower_delta, upper = unif_upper_delta)

# Estimate parameters using MCMC.
MCMC_results <- COMBO_MCMC_2stage(Ystar, Ytilde,
                                  x = x_matrix, z = z_matrix, v = v_matrix,
                                  prior = "uniform",
                                  beta_prior_parameters = beta_prior_parameters,
                                  gamma_prior_parameters = gamma_prior_parameters,
                                  delta_prior_parameters = delta_prior_parameters,
                                  naive_delta_prior_parameters = gamma_prior_parameters,
                                  number_MCMC_chains = 4,
                                  MCMC_sample = 6000, burn_in = 2000)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2000
##    Unobserved stochastic nodes: 14
##    Total graph size: 81063
##
## Initializing model
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2000
##    Unobserved stochastic nodes: 6
##    Total graph size: 36030
##
## Initializing model
```

```
MCMC_results$posterior_means_df
```

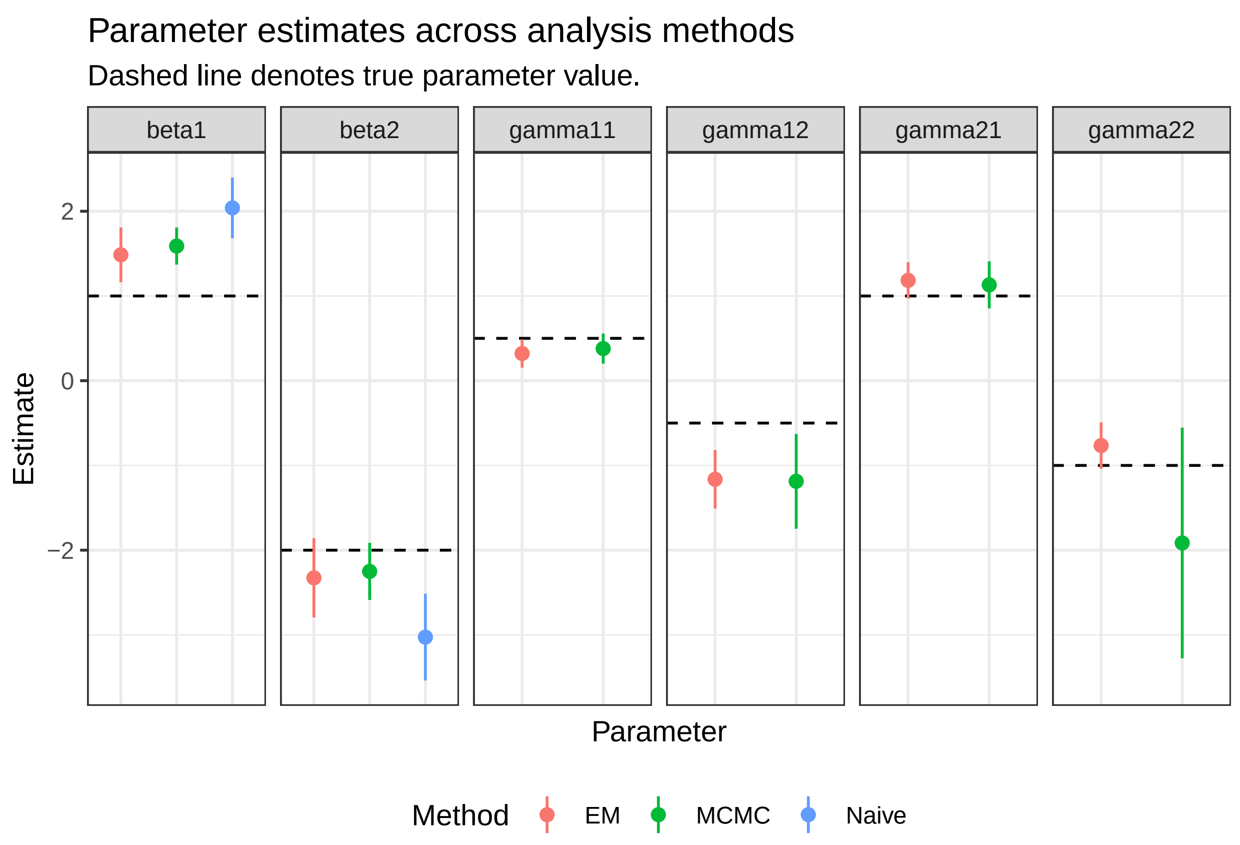
```
## # A tibble: 14 x 3
##    parameter      posterior_mean posterior_median
##    <fct>                   <dbl>            <dbl>
##  1 beta[1,1]              1.59             1.58
##  2 beta[1,2]             -2.25            -2.21
##  3 gamma[1,1,1]           0.379            0.376
##  4 gamma[1,2,1]          -1.19            -1.16
##  5 gamma[1,1,2]           1.13             1.10
##  6 gamma[1,2,2]          -1.91            -1.61
##  7 delta[1,1,1,1]         0.844            0.793
##  8 delta[1,2,1,1]         0.569            0.288
##  9 delta[1,1,2,1]        -0.0165          -0.0578
## 10 delta[1,2,2,1]        -0.955           -0.790
## 11 delta[1,1,1,2]         1.26             1.23
## 12 delta[1,2,1,2]         2.36             2.25
## 13 delta[1,1,2,2]        -0.659           -1.03
## 14 delta[1,2,2,2]        -2.42            -2.29
```

```
MCMC_results$naive_posterior_means_df
```

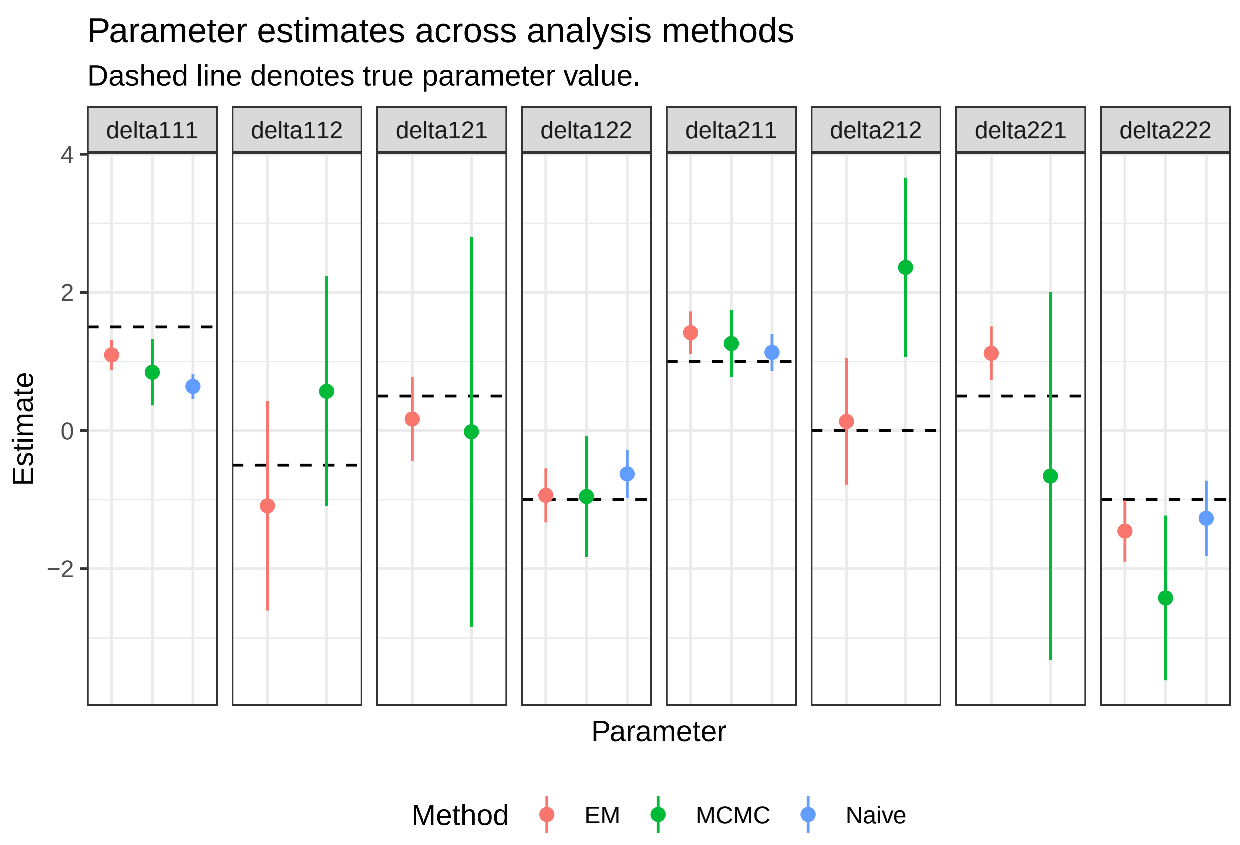
```
## # A tibble: 6 x 3
##   parameter          posterior_mean posterior_median
##   <chr>                       <dbl>            <dbl>
## 1 naive_beta[1,1]             0.519            0.520
## 2 naive_beta[1,2]            -0.967           -0.965
## 3 naive_delta[1,1,1]          1.75             1.60
## 4 naive_delta[1,1,2]          3.13             3.14
## 5 naive_delta[1,2,1]         -2.16            -2.02
## 6 naive_delta[1,2,2]         -0.957           -0.358
```

Plotting effect estimates

Figure 2 shows the parameter estimates (+/- one standard deviation) for different analysis methods: EM, MCMC, and a “naive” two-stage regression.



Parameter estimates across analysis methods
Dashed line denotes true parameter value.
beta1
beta2
gamma11
gamma12
gamma21
gamma22
2
0
−2
Estimate
Parameter
Method
EM
MCMC
Naive



Estimating sensitivity and specificity

For each analysis method, we may use the estimated γ parameters to compute estimates of first-stage and second-stage sensitivity and specificity as a function of the covariates, z and v, respectively. Here, we compute these values under the EM algorithm estimates, MCMC estimates, and using the generated data.

```
# Create matrix of gamma parameter estimates from the EM algorithm.
EM_gamma <- matrix(EM_results$Estimates[3:6], ncol = 2, byrow = FALSE)
EM_delta <- array(EM_results$Estimates[7:14], dim = c(2,2,2))

# Compute misclassification probabilities.
EM_misclassification_prob <- misclassification_prob(EM_gamma,
                                                    matrix(z_matrix, ncol = 1))
EM_misclassification_prob2 <- misclassification_prob2(EM_delta,
                                                      matrix(v_matrix, ncol = 1))

# Find the average sensitivity and specificity.
EM_sensitivity_df <- EM_misclassification_prob %>%
  filter(Y == 1) %>% filter(Ystar == 1)
EM_sensitivity <- mean(EM_sensitivity_df$Probability)

EM_specificity_df <- EM_misclassification_prob %>%
  filter(Y == 2) %>% filter(Ystar == 2)
EM_specificity <- mean(EM_specificity_df$Probability)
```

```r
EM_sensitivity2_df <- EM_misclassification_prob2 %>%
  filter(Y == 1) %>% filter(Ystar == 1) %>% filter(Ytilde == 1)
EM_sensitivity2 <- mean(EM_sensitivity2_df$Probability)

EM_specificity2_df <- EM_misclassification_prob2 %>%
  filter(Y == 2) %>% filter(Ystar == 2) %>% filter(Ytilde == 2)
EM_specificity2 <- mean(EM_specificity2_df$Probability)
```

```r
# Create matrix of gamma parameter estimates from MCMC.
MCMC_gamma <- matrix(MCMC_results$posterior_means_df$posterior_mean[3:6],
                     ncol = 2, byrow = TRUE)
MCMC_delta <- array(MCMC_results$posterior_means_df$posterior_mean[c(7, 11,
                                                                     9, 13,
                                                                     8, 12,
                                                                     10, 14)],
                    dim = c(2,2,2))

# Compute misclassification probabilities.
MCMC_misclassification_prob <- misclassification_prob(MCMC_gamma,
                                                      matrix(z_matrix, ncol = 1))
MCMC_misclassification_prob2 <- misclassification_prob2(MCMC_delta,
                                                        matrix(v_matrix, ncol = 1))

# Find the average sensitivity and specificity
MCMC_sensitivity_df <- MCMC_misclassification_prob %>%
  filter(Y == 1) %>% filter(Ystar == 1)
MCMC_sensitivity <- mean(MCMC_sensitivity_df$Probability)

MCMC_specificity_df <- MCMC_misclassification_prob %>%
  filter(Y == 2) %>% filter(Ystar == 2)
MCMC_specificity <- mean(MCMC_specificity_df$Probability)

MCMC_sensitivity2_df <- MCMC_misclassification_prob2 %>%
  filter(Y == 1) %>% filter(Ystar == 1) %>% filter(Ytilde == 1)
MCMC_sensitivity2 <- mean(MCMC_sensitivity2_df$Probability)

MCMC_specificity2_df <- MCMC_misclassification_prob2 %>%
  filter(Y == 2) %>% filter(Ystar == 2) %>% filter(Ytilde == 2)
MCMC_specificity2 <- mean(MCMC_specificity2_df$Probability)
```

```r
# Use the generated data to compute the actual sensitivity and specificity rate.
data_classification_table <- table(my_data[["obs_Ystar"]], my_data[["true_Y"]])

data_classification_table2 <- table(my_data[["obs_Ytilde"]], my_data[["obs_Ystar"]], my_data[["true_Y"]

true_sensitivity <- prop.table(data_classification_table, 2)[1,1]

true_specificity <- prop.table(data_classification_table, 2)[2,2]

true_sensitivity_2stage <- data_classification_table2[1,1,1] /
  sum(data_classification_table2[,1,1])

true_specificity_2stage <- data_classification_table2[2,2,2] /
```

```
sum(data_classification_table2[,2,2])
```

	Sensitivity, P(Y*(1) = 1 | Y = 1)	Specificity, P(Y*(1) = 2 | Y = 2)
Data	0.771	0.798
EM	0.768	0.853
MCMC	0.773	0.910

	P(Y*(2) = 1 |Y*(1) = 1, Y = 1)	P(Y*(2) = 2 | Y*(1) = 2, Y = 2)
Data	0.900	0.853
EM	0.888	0.874
MCMC	0.853	0.911

Table 1 shows the actual sensitivity and specificity values for the data, in addition to the average sensitivity and specificity estimates computed from EM-Algorithm and MCMC parameter estimates and the covariate z and v.

References

Hochstedler, K.A. and Wells, M.T. "Statistical inference for association studies in the presence of binary outcome misclassification", (2022). In preparation.