Exact Asymptotics for Causal Mediation Analysis

William Ruth

2024-05-20

I'm doing some Monte Carlo to verify the new SE formulas. Recall that we're doing mediation analysis, so we've got a response, Y, an exposure, X, and a mediation, M. We also have some number of confounders, which will be grouped together in the matrix W. Broadly speaking, we fit two regression models, one to predict M using X and W, the other to predict Y using M, X and W. We then compute the mediation effect (specifically, the total effect of X on Y) as a function of the coefficients from these two regression models. An asymptotic SE for our mediation effect estimator can then be obtained from the asymptotic standard errors of our fitted regression coefficients using the δ-method.

So far, so simple. There are a few places that things start to get more complicated. First, each of the two regression models can be either linear of logistic depending on whether the corresponding response variable is continuous or binary[1]. Furthermore, we can add random effects to our regression models. In the trust study, we have random effects for the intercept, X and M (naturally, the latter only applies when predicting Y).

In each setting, I generate X and three confounders, $W = [W_1, W_2, W_3]$, each iid $\mathrm{N}(0, 1)$. In both models, I set regression slopes to 1, and choose the intercept so that the mean of the linear predictor is approximately zero[2]. I generate some datasets (typically 1000, can be adjusted as necessary) with each of $n = 100, 1000$ and 10000 observations. On each dataset, we estimate the mediation effect and its SE (using the δ-method). We then compare the empirical SE (i.e. SD over the M Monte Carlo replicates) to the mean and median of our M estimated SEs.

In summary, we can setup each model with the same boilerplate code as follows:

```r
# , cached=TRUE} Note: This code chunk is not run here, but
# it is called and run at the start of every subsequent
# section. Any changes made here will be duplicated
# throughout the rest of the document.

set.seed(1)

num_reps = 1000

all_Ns = c(100, 200, 500, 1000, 2000, 5000, 10000)
p_conf = 3  # Number of confounders


# Regression coefficients Intercepts are set separately for
# each model so that the mean of the linear predictor is
# approximately zero

## M model
```

[1]In principle, we could have Y and/or M follow any distribution with a suitable GLM formulation. I don't think I've ever seen count data (i.e. Poisson regression) used here, much less anything more exotic.

[2]In the fully continuous case, the mean linear predictor is exactly zero. When the mediator is binary, I choose the Y-intecept as though $\mathbb{E}M = \mathrm{expit}(0) = 0.5$

```
a_1 = 1  # Coefficient for X
A_2 = rep(1, times = p_conf)  # Coefficients for confounders

# Y model
b_1 = 1  # Coefficient for M
b_2 = 1  # Coefficient for X
B_3 = rep(1, times = p_conf)  # Coefficients for confounders


# Containers for output
all_a_hats = list()
all_b_hats = list()

all_a_SEs = list()
all_b_SEs = list()

all_med_hats = list()
all_med_SEs = list()
```

In the rest of this document, we carry our Monte Carlo studies for different model configurations. We start with the simplest case: continuous response, continuous mediator, and fixed-effects. We then move to a binary mediator, followed by binary response. Finally, we introduce random effects.

Continuous Response, Continuous Mediator, Fixed-Effects

Generate data

Continuous variables are modelled as linear predictor plus residual. We therefore need to set the residual variance.

```
sigma_M = 0.2
sigma_Y = 0.2
```

We also need to set the intercepts for the two models.

```
a_0 = 0
b_0 = 0
```

For pedagogical purposes, I will demonstrate the analysis on a single simulated dataset, then run the full MC study invisibly and only show the results. First, we choose a sample size, n, and generate X and W, then use them to generate M and Y.

```
n = all_Ns[1]

X = rnorm(n, mean = 0, sd = 1)
W = matrix(rnorm(n * p_conf, mean = 0, sd = 1), nrow = n, ncol = p_conf)

# Generate M
e_M = rnorm(n, 0, sigma_M)
M = a_0 + a_1 * X + W %*% A_2 + e_M

# Generate Y
e_Y = rnorm(n, 0, sigma_Y)
Y = b_0 + b_1 * M + b_2 * X + W %*% B_3 + e_Y
```

Estimate Mediation Effect

Next, we fit regression models for M and Y and extract relevant output (in the loop version, these are stored at each iteration)

```
M_data = data.frame(M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3])
M_model = lm(M ~ X + W1 + W2 + W3, data = M_data)

a_hat = summary(M_model)$coefficients[, 1]
a_SE = summary(M_model)$coefficients[, 2]
a_cov = vcov(M_model)


Y_data = data.frame(Y, M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3])
Y_model = lm(Y ~ M + X + W1 + W2 + W3, data = Y_data)

b_hat = summary(Y_model)$coefficients[, 1]
b_SE = summary(Y_model)$coefficients[, 2]
b_cov = vcov(Y_model)
```

Now we can extract relevant coefficients and estimate the mediation effect

```
a_x = a_hat[2]
b_x = b_hat[3]
b_m = b_hat[2]

med_hat = a_x * b_m + b_x
```

Estimate Standard Error

Finally, we can estimate the SE of the mediation effect using the δ-method. The way I do this is excessive here since the mediation effect has such a simple formula. Later though, the extra structure will make our lives easier.

We start by constructing the joint covariance matrix of the regression coefficients from the two models. This matrix is block-diagonal, with the blocks corresponding to the covariance matrices of the coefficients from the two models and off-diagonal entries zero.

```
a_length = nrow(a_cov)
b_length = nrow(b_cov)

joint_cov = matrix(0, nrow = a_length + b_length, ncol = a_length +
    b_length)
joint_cov[1:a_length, 1:a_length] = a_cov
joint_cov[(a_length + 1):(a_length + b_length), (a_length + 1):(a_length +
    b_length)] = b_cov
```

Next, we compute the gradient of the mediation effect with respect to each regression coefficient.

```
grad_a_0 = 0
grad_a_1 = b_m
grad_A_2 = rep(0, times = p_conf)

grad_b_0 = 0
grad_b_1 = a_x
```

```r
grad_b_2 = 1
grad_B_3 = rep(0, times = p_conf)

grad_med = c(grad_a_0, grad_a_1, grad_A_2, grad_b_0, grad_b_1,
    grad_b_2, grad_B_3)
```

Finally, we can use the delta method to estimate the SE of the mediation effect. Note that the δ-method works on asymptotic covariances, so we need to make sure to multiply/divide by n or $\sqrt{n}$ where appropriate.

```r
asymp_reg_cov = n * joint_cov

med_asymp_var = grad_med %*% asymp_reg_cov %*% grad_med
med_asymp_SE = sqrt(med_asymp_var)

med_SE = med_asymp_SE/sqrt(n)
```

Putting everything together, we have an estimated mediation effect of 2.0140277 with an estimated SE of 0.0329957.

Monte Carlo Study

We now repeat the above analysis 1000 times for each of various values of n. We also re-run the boilerplate code (not shown)

We now process the output from our simulation and summarize the results in a table. We give the mean and median of the estimated standard errors, as well as their relative errors (in %) as estimates of the empirical standard error. See Table 1. Note that the relative error decreased and became more consistent when I increased the number of Monte Carlo replicates from 100 to 1000.

```r
# Re-format output to be more useful.
data_med_hats = lapply(all_med_hats, unlist)
data_med_SEs = lapply(all_med_SEs, unlist)

# SD of estimates across Monte Carlo samples
emp_SEs = sapply(data_med_hats, sd)

# Summaries of estimated SEs (mean and median)
mean_delta_SEs = sapply(data_med_SEs, mean)
median_delta_SEs = sapply(data_med_SEs, median)


# Combine into a table
results = data.frame(n = all_Ns, Empirical = emp_SEs, Mean = mean_delta_SEs,
    Median = median_delta_SEs, Percent_Err_Mean = 100 * (mean_delta_SEs -
        emp_SEs)/emp_SEs, Percent_Err_Median = 100 * (median_delta_SEs -
        emp_SEs)/emp_SEs)
```

Continuous Response, Binary Mediator, Fixed-Effects

We now generate M as a binary variable using a logistic regression model. We continue to use a continuous model for $Y|M$. This analysis is sufficiently complicated that I wrote some helper functions in another script. See `Helpers.R`.

```r
source("../src/Helpers.R")
```

Table 1: Summary of SE estimates for mediation effect under continuous response, continuous mediator, fixed-effects model

n	Empirical	Mean	Median	Percent_Err_Mean	Percent_Err_Median
100	0.0288078	0.0291424	0.0290042	1.1612527	0.6815971
200	0.0201599	0.0202228	0.0202594	0.3119318	0.4932875
500	0.0129226	0.0127142	0.0126968	-1.6130839	-1.7476840
1000	0.0090383	0.0089748	0.0089642	-0.7023415	-0.8199452
2000	0.0062943	0.0063324	0.0063253	0.6060927	0.4925302
5000	0.0040719	0.0040040	0.0040013	-1.6668130	-1.7331162
10000	0.0028470	0.0028287	0.0028283	-0.6423586	-0.6572552

Generate data

The continuous variable is modelled as linear predictor plus residual. We therefore need to set the residual variance.

```
sigma_Y = 0.2
```

We also need to set the intercepts for the two models. Recall that we want the mean of the linear predictor to be zero in both models, and that we approximate the mean of M as the logit of the mean of its linear predictor (i.e. $\mathbb{E}M \approx \text{expit}(0) = 0.5$).

```
a_0 = 0
b_0 = -0.5
```

As in the previous section, I demonstrate the analysis on a single simulated dataset, then run the full MC study invisibly and only show the results. First, we choose a sample size, n, and generate X and W, then use them to generate M and Y.

```
n = all_Ns[1]

X = rnorm(n, mean = 0, sd = 1)
W = matrix(rnorm(n * p_conf, mean = 0, sd = 1), nrow = n, ncol = p_conf)

# Generate M
eta_vec = a_0 + a_1 * X + W %*% A_2
p_M_vec = expit(eta_vec)
M = rbinom(n, size = 1, prob = p_M_vec)

# Generate Y
e_Y = rnorm(n, 0, sigma_Y)
Y = b_0 + b_1 * M + b_2 * X + W %*% B_3 + e_Y
```

Estimate Mediation Effect

Next, we fit regression models for M and Y and extract relevant output (in the loop version, these are stored at each iteration)

```
M_data = data.frame(M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3])
M_model = glm(M ~ X + W1 + W2 + W3, data = M_data, family = binomial(link = "logit"))

a_hat = summary(M_model)$coefficients[, 1]
```

```r
a_SE = summary(M_model)$coefficients[, 2]
a_cov = vcov(M_model)


Y_data = data.frame(Y, M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3])
Y_model = lm(Y ~ M + X + W1 + W2 + W3, data = Y_data)

b_hat = summary(Y_model)$coefficients[, 1]
b_SE = summary(Y_model)$coefficients[, 2]
b_cov = vcov(Y_model)
```

Now we can extract relevant coefficients and estimate the mediation effect. Note that this is more involved than the fully continuous case. In particular, the total effect of X on Y depends on the levels of X and the confounders, W. I evaluate the effect at $X = 0$ and $W = [1, 1, 1]$. The former represents the effect of a binary exposure, but the latter is chosen completely arbitrarily.

```r
x_pred = 0
W_pred = c(1, 1, 1)

a_0_hat = a_hat[1]
a_x_hat = a_hat[2]
A_2_hat = a_hat[3:5]

b_0_hat = b_hat[1]
b_m_hat = b_hat[2]
b_x_hat = b_hat[3]
B_3_hat = b_hat[4:6]

# Linear predictor for M
eta_hat = a_0_hat + a_x_hat * x_pred + W_pred %*% A_2_hat

# Increment in the conditional expectation of M
delta_hat = get_delta(eta_hat, a_x_hat)

# Increment in the conditional expectation of Y
med_hat = get_gamma(delta_hat, b_m_hat, b_x_hat)
```

Estimate Standard Error

Finally, we can estimate the SE of the mediation effect using the δ-method. Now that we have a more complicated expression for the total effect, it makes more sense to use the general formula for the δ-method standard error.

We start by constructing the joint covariance matrix of the regression coefficients from the two models. This matrix is block-diagonal, with the blocks corresponding to the covariance matrices of the coefficients from the two models and off-diagonal entries zero.

```r
a_length = nrow(a_cov)
b_length = nrow(b_cov)

joint_cov = matrix(0, nrow = a_length + b_length, ncol = a_length +
    b_length)
joint_cov[1:a_length, 1:a_length] = a_cov
joint_cov[(a_length + 1):(a_length + b_length), (a_length + 1):(a_length +
    b_length)] = b_cov
```

Next, we compute the gradient of the mediation effect with respect to each regression coefficient.

```
d_gamma_d_theta(eta_hat, x_pred, W_pred, a_hat, b_hat)
```

```
##  [1] -0.06963400  0.05727213 -0.06963400 -0.06963400 -0.06963400  0.00000000
##  [7]  0.08617144  1.00000000  0.00000000  0.00000000  0.00000000
```

Finally, we can use the delta method to estimate the SE of the mediation effect. Note that the δ-method works on asymptotic covariances, so we need to make sure to multiply/divide by n or $\sqrt{n}$ where appropriate.

```
asymp_reg_cov = n * joint_cov

med_asymp_var = grad_med %*% asymp_reg_cov %*% grad_med
med_asymp_SE = sqrt(med_asymp_var)

med_SE = med_asymp_SE/sqrt(n)
```

Putting everything together, we have an estimated mediation effect of 1.062491 with an estimated SE of 0.312161.

Monte Carlo Study

We now repeat the above analysis 1000 times for each of various values of n. Values of X and W for which we compute the total effect are set at the beginning. We also re-run the boilerplate code (not shown)

```
# Values of X and W for which we compute the total effect
x_pred = 0
W_pred = c(1, 1, 1)
```

We now process the output from our simulation and summarize the results in a table. We give the mean and median of the estimated standard errors, as well as their relative errors (in %) as estimates of the empirical standard error. See Table 2. Note that the relative error decreased and became more consistent when I increased the number of Monte Carlo replicates from 100 to 1000.

```
# Re-format output to be more useful.
data_med_hats = lapply(all_med_hats, unlist)
data_med_SEs = lapply(all_med_SEs, unlist)

# SD of estimates across Monte Carlo samples
emp_SEs = sapply(data_med_hats, sd)

# Summaries of estimated SEs (mean and median)
mean_delta_SEs = sapply(data_med_SEs, mean)
median_delta_SEs = sapply(data_med_SEs, median)


# Combine into a table
results = data.frame(n = all_Ns, Empirical = emp_SEs, Mean = mean_delta_SEs,
    Median = median_delta_SEs, Percent_Err_Mean = 100 * (mean_delta_SEs -
        emp_SEs)/emp_SEs, Percent_Err_Median = 100 * (median_delta_SEs -
        emp_SEs)/emp_SEs)
```

Table 2: Summary of SE estimates for mediation effect under continuous response, continuous mediator, fixed-effects model

n	Empirical	Mean	Median	Percent_Err_Mean	Percent_Err_Median
100	0.0275592	0.0285038	0.0273933	3.4274209	-0.6021133
200	0.0200177	0.0199399	0.0196113	-0.3886780	-2.0300435
500	0.0127654	0.0124220	0.0123546	-2.6900591	-3.2175895
1000	0.0089975	0.0087489	0.0087114	-2.7630482	-3.1792769
2000	0.0061285	0.0061699	0.0061539	0.6751659	0.4134499
5000	0.0038802	0.0039096	0.0039055	0.7567619	0.6523393
10000	0.0027090	0.0027636	0.0027647	2.0170278	2.0575845

Binary Response, Binary Mediator, Fixed-Effects

For our last fixed-effect model, we now take both the response and mediator to be binary. I wrote some helper functions to compute relevant derivatives, all of which validated against a simple finite-difference approximation. See `Helpers.R`.

```
source("../src/Helpers.R")
```

Generate Data

As in the previous case, we set intercepts so that the means of both linear predictors are at least approximately zero. Here however, no residuals are required.

```
a_0 = 0
b_0 = -0.5
```

We first demonstrate our analysis on a single dataset, then put this analysis inside a `for` loop. The first step is to set the sample size and generate some data. We use η to denote the linear predictor for M, and ζ for the linear predictor of Y.

```
n = all_Ns[1]

X = rnorm(n, mean = 0, sd = 1)
W = matrix(rnorm(n * p_conf, mean = 0, sd = 1), nrow = n, ncol = p_conf)

# Generate M
eta_vec = a_0 + a_1 * X + W %*% A_2
p_M_vec = expit(eta_vec)
M = rbinom(n, size = 1, prob = p_M_vec)

# Generate Y
zeta_vec = b_0 + b_1 * M + b_2 * X + W %*% B_3
p_Y_vec = expit(zeta_vec)
Y = rbinom(n, size = 1, prob = p_Y_vec)
```

Estimate Mediation Effect

We now fit logistic regression models to predict M and Y, and extract output that we will need later.

```
M_data = data.frame(M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3])
M_model = glm(M ~ X + W1 + W2 + W3, data = M_data, family = binomial(link = "logit"))
```

```r
a_hat = summary(M_model)$coefficients[, 1]
a_SE = summary(M_model)$coefficients[, 2]
a_cov = vcov(M_model)


Y_data = data.frame(Y, M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3])
Y_model = glm(Y ~ M + X + W1 + W2 + W3, data = Y_data, family = binomial(link = "logit"))

b_hat = summary(Y_model)$coefficients[, 1]
b_SE = summary(Y_model)$coefficients[, 2]
b_cov = vcov(Y_model)
```

Next, we estimate the mediation effect. Note that, as in the contiuous outcome, binary mediator setting, our mediation effect depends on X and W. We again use $X = 0$ and $W = [1, 1, 1]$, although Samoilenko and Lefebvre (2023) use the sample means of these covariates.

```r
x_pred = 0
W_pred = c(1, 1, 1)

a_0_hat = a_hat[1]
a_x_hat = a_hat[2]
A_2_hat = a_hat[3:5]

b_0_hat = b_hat[1]
b_m_hat = b_hat[2]
b_x_hat = b_hat[3]
B_3_hat = b_hat[4:6]

# Linear predictors
eta_hat = a_0_hat + a_x_hat * x_pred + W_pred %*% A_2_hat
zeta_hat = b_0_hat + b_x_hat * x_pred + W_pred %*% B_3_hat

# Mediation effect See Helpers.R for the function
# get_odds_ratio
med_hat = get_odds_ratio(eta_hat, a_x, zeta_hat, b_x, b_m)
```

We now compute the gradient of the odds ratio and use the δ-method to get the standard error.

```r
# Gradient of OR wrt regression coefficients
grad_med = d_OR_d_theta(eta_hat, a_x_hat, zeta_hat, b_x_hat,
    b_m_hat, x_pred, W_pred)


# Get asymptotic SE using delta method

## Build joint covariance matrix of regression coefficients
a_length = nrow(a_cov)
b_length = nrow(b_cov)
joint_cov = matrix(0, nrow = a_length + b_length, ncol = a_length +
    b_length)
joint_cov[1:a_length, 1:a_length] = a_cov
joint_cov[(a_length + 1):(a_length + b_length), (a_length + 1):(a_length +
    b_length)] = b_cov
```

```r
## Convert to asymptotic covariance matrix
asymp_reg_cov = n * joint_cov

## Pre- and post-multiply asymptotic covariance by gradient
## of transformation
med_asymp_var = grad_med %*% asymp_reg_cov %*% grad_med

## Get small-sample standard error
med_asymp_SE = sqrt(med_asymp_var)
med_SE = med_asymp_SE/sqrt(n)
```

Putting everything together, we have an estimated mediation effect of 2.9106008, with an estimated SE of 1.0862223.

Monte Carlo Study

We now repeat the above analysis 1000 times for each of various values of n. Values of X and W for which we compute the total effect are set at the beginning. We also re-run the boilerplate code (not shown).

```r
# Values of X and W for which we compute the total effect
x_pred = 0
W_pred = c(1, 1, 1)

# Values of a_0 and b_0
a_0 = 0
b_0 = -0.5
```

We now process the output from our simulation and summarize the results in a table. We give the mean and median of the estimated standard errors, as well as their relative errors (in %) as estimates of the empirical standard error. See Table 3.

```r
# Re-format output to be more useful.
data_med_hats = lapply(all_med_hats, unlist)
data_med_SEs = lapply(all_med_SEs, unlist)

# SD of estimates across Monte Carlo samples
emp_SEs = sapply(data_med_hats, sd)

# Summaries of estimated SEs (mean and median)
mean_delta_SEs = sapply(data_med_SEs, mean)
median_delta_SEs = sapply(data_med_SEs, median)


# Combine into a table
results = data.frame(n = all_Ns, Empirical = emp_SEs, Mean = mean_delta_SEs,
    Median = median_delta_SEs, Percent_Err_Mean = 100 * (mean_delta_SEs -
        emp_SEs)/emp_SEs, Percent_Err_Median = 100 * (median_delta_SEs -
        emp_SEs)/emp_SEs)
```

Binary Response, Binary Mediator, Mixed-Effects

We now introduce random effects to our model. This changes many of the details, but the overall structure of the analysis remains the same. I still need to write some helper functions to compute relevant derivatives, all of which validated against a simple finite-difference approximation. See `Helpers.R`.

Table 3: Summary of SE estimates for mediation effect under continuous response, continuous mediator, fixed-effects model

n	Empirical	Mean	Median	Percent_Err_Mean	Percent_Err_Median
100	2.2168894	1.6051059	1.1979888	-27.5964816	-45.9608225
200	0.8874067	0.8359533	0.7517891	-5.7981709	-15.2824574
500	0.4780655	0.4639575	0.4487406	-2.9510602	-6.1340829
1000	0.3246009	0.3133180	0.3070522	-3.4759228	-5.4062403
2000	0.2161390	0.2182134	0.2157025	0.9597329	-0.2019576
5000	0.1388760	0.1369564	0.1362965	-1.3822398	-1.8574378
10000	0.0928707	0.0962476	0.0960962	3.6360866	3.4731240

```r
source("../src/Helpers.R")

library(lme4)  # For fitting GLMMs using glmer()
```

```
## Loading required package: Matrix
```

```r
library(MASS)  # For simulating multivariate normals using mvrnorm()
library(merDeriv)  # For computing the full information matrix from models fit using lme4
```

```
## Loading required package: nonnest2

## This is nonnest2 0.5-7.
## nonnest2 has not been tested with all combinations of supported model classes.

## Loading required package: sandwich

## Loading required package: lavaan

## This is lavaan 0.6-17
## lavaan is FREE software! Please report any bugs.
```

Generate Data

As in the previous case, we set intercepts so that the means of both linear predictors are at least approximately zero. We also need to set covariance parameters for the random effects. For this, we use either 20% of the true value or 0.2 if the true value is zero. Correlations between all random effects are set to 0.5. Note that we include random effects for the intercept and the exposure, but not for the mediator.

Another difference from the previous settings is that we must now construct our dataset in clusters. We will use $K = 5$ such clusters, with n observations in each cluster. We will set $n = 100$ for the first iteration of the loop.

```r
# Fixed effects for intercepts
a_0 = 0
b_0 = -0.5

# 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)
```

```r
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)

# Number of clusters
K = 5
```

We first demonstrate our analysis on a single dataset, then put this analysis inside a `for` loop. The first step is to set the sample size and generate some data. We use η to denote the linear predictor for M, and ζ for the linear predictor of Y.

```r
n = all_Ns[2]
# n = all_Ns[1]

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)

    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)
```

Estimate Mediation Effect

We now fit logistic regression models to predict M and Y, and extract output that we will need later.

```
M_data = data.frame(M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3], group = group)
M_model = glmer(M ~ X + W1 + W2 + W3 + (1 + X | group), data = M_data,
    family = binomial(link = "logit"))
```

```
## boundary (singular) fit: see help('isSingular')
```

```
M_model_info = attributes(VarCorr(M_model)$group)

a_hat = fixef(M_model)
a_RE_sds = M_model_info$stddev
a_RE_cor = M_model_info$correlation[2, 1]
a_cov = vcov(M_model, full = TRUE, ranpar = "sd")

# a_hat = summary(M_model)$coefficients[, 1] a_SE =
# summary(M_model)$coefficients[, 2] a_cov = vcov(M_model)


Y_data = data.frame(Y, M, X, W1 = W[, 1], W2 = W[, 2], W3 = W[,
    3], group = group)
Y_model = glmer(Y ~ M + X + W1 + W2 + W3 + (1 + X | group), data = Y_data,
    family = binomial(link = "logit"))
Y_model_info = attributes(VarCorr(Y_model)$group)

b_hat = fixef(Y_model)
b_RE_sds = Y_model_info$stddev
b_RE_cor = Y_model_info$correlation[2, 1]
b_cov = vcov(Y_model, full = TRUE, ranpar = "sd")

# b_hat = summary(Y_model)$coefficients[, 1] b_SE =
# summary(Y_model)$coefficients[, 2] b_cov = vcov(Y_model)
```

Next, we estimate the mediation effect. Note that, as in the contiuous outcome, binary mediator setting, our mediation effect depends on X and W. We again use $X = 0$ and $W = [1, 1, 1]$, although Samoilenko and Lefebvre (2023) use the sample means of these covariates.

```r
# Fixed-effects
x_pred = 0
W_pred = c(1, 1, 1)

a_0_hat = a_hat[1]
a_x_hat = a_hat[2]
A_2_hat = a_hat[3:5]

b_0_hat = b_hat[1]
b_m_hat = b_hat[2]
b_x_hat = b_hat[3]
B_3_hat = b_hat[4:6]


# Linear predictors
eta_hat = as.numeric(a_0_hat + a_x_hat * x_pred + W_pred %*%
    A_2_hat)
zeta_hat = as.numeric(b_0_hat + b_x_hat * x_pred + W_pred %*%
    B_3_hat)


# Random effects covariances
s_M_0 = a_RE_sds[1]
s_M_x = a_RE_sds[2]
rho_M = a_RE_cor

s_Y_0 = b_RE_sds[1]
s_Y_x = b_RE_sds[2]
rho_Y = b_RE_cor


# Sigma functions
sigma_M1 = sigma_fun(x_pred, s_M_0, s_M_x, rho_M)
sigma_M2 = sigma_fun(x_pred + 1, s_M_x, s_M_0, rho_M)

sigma_Y1 = sigma_fun(x_pred, s_Y_0, s_Y_x, rho_Y)
sigma_Y2 = sigma_fun(x_pred + 1, s_Y_x, s_Y_0, rho_Y)

# Mediation effect See Helpers.R for the function
# get_odds_ratio
med_hat = Phi(eta_hat, zeta_hat, a_x_hat, b_m_hat, b_x_hat, sigma_M2,
    sigma_Y2, sigma_M1, sigma_Y1)
```

Applying the δ-method here is a bit more complicated than in the other settings considered so far. Following the notes written by B & B, we take a two-step approach. We first transform model parameters from the two GLMMs into the vector of arguments to Φ; call such a function ξ. We then compute the gradient of Φ with respect to these arguments, and use the δ-method to get the standard error.

In order to perform the first step, we need the gradient of ξ. Fortunately, much of this has already been done for us. Specifically, B & B worked out the gradients of each of the four σ functions. All that remains is to repeat this process for the other arguments of Φ, and stack the results in a matrix. See the `Helpers.R` file for the relevant functions.

Starting first with We now compute the gradient of the odds ratio and use the δ-method to get the standard error.

```
# # Gradient of OR wrt regression coefficients grad_med =
# d_OR_d_theta(eta_hat, a_x_hat, zeta_hat, b_x_hat,
# b_m_hat, x_pred, W_pred) # Get asymptotic SE using delta
# method ## Build joint covariance matrix of regression
# coefficients a_length = nrow(a_cov) b_length =
# nrow(b_cov) joint_cov = matrix(0, nrow = a_length +
# b_length, ncol = a_length + b_length)
# joint_cov[1:a_length, 1:a_length] = a_cov
# joint_cov[(a_length + 1):(a_length + b_length), (a_length
# + 1):(a_length + b_length)] = b_cov ## Convert to
# asymptotic covariance matrix asymp_reg_cov = n *
# joint_cov ## Pre- and post-multiply asymptotic covariance
# by gradient of transformation med_asymp_var = grad_med
# %*% asymp_reg_cov %*% grad_med ## Get small-sample
# standard error med_asymp_SE = sqrt(med_asymp_var) med_SE
# = med_asymp_SE / sqrt(n)
```

Putting everything together, we have an estimated mediation effect of 2.5585832, with an estimated SE of 0.0954856.