Supplemental Material

Requirement

To perform sensitivity analysis, you need to have R and Mplus installed on your computer. To make sure the following code runs smoothly, we recommend installing R Version 3.4.3 or later, although an earlier version might work.

After installing R, make sure the following packages are also installed: `tidyverse, stringr, MplusAutomation, RMediation, doParallel, and latticeExtra`. You can install all the packages in R with the following code (change or delete repos argument as needed):

```
pkg <- c("tidyverse", "stringr", "MplusAutomation", "RMediation", "doParallel
",
    "latticeExtra")
install.packages(pkg, repos = "https://cloud.r-project.org/")
```

Accessing Source Code Files

We created a zip file called **sensitivity.zip**. You can download the zip file directly by clicking on the following link (download sensitivity zip file), save it in a directory on your computer, and then unzip it. After downloading and unzipping **sensitivity.zip**, your directory should contain the following files.

1. **sensitivity_2m_par.R**: This file contains the functions `sensitivity_2m_par` that will read an Mplus output file and then conduct sensitivity analysis.
2. **plot_2m_par.R**: contains the function `plot_2m_par` to produce sensitivity confidence band plots.
3. **runmplus.R** contains the function `runmplus` to run Mplus from R. This function is called by the function `sensitivity_2m_par`, not directly by the user.
4. **med_vec.R** contains the function `med_vec` to compute point and confidence interval for an indirect effect. This function is called by the function `sensitivity_2m_par`, not directly by the user.
5. We also included our Mplus input file **lgcmm_template.inp** for illustration purposes. Detailed description of the Mplus code is provided in the next section.

Make sure that you downloaded the above R script files in one directory, where you can use the `source` function to read them into R. See the relevant source commands below.

```r
## load the following libraries.
library(tidyverse)  #tidyverse loads a bunch of libraries including dplyr
library(stringr)
library(MplusAutomation)
library(RMediation)
library(doParallel)
library(lattice)
library(latticeExtra)
## load the following R code into R
source("sensitivity_2m_par.R")  # read the file into R
source("plot_2m_par.R")  # read file into R
source("runmplus.R")
source("med_vec.R")
```

Mplus Code Structure

It is critical to make sure the Mplus code you provide meets certain criteria outlined below. Because your Mplus code will *not* run in Mplus (we explain later why), we call this an Mplus *template*. For brevity, we show `MODEL:` and `MODEL CONSTRAINT:` parts of the Mplus template to further clarify the structure of the code used in conducting sensitivity analysis. You can download our Mplus template file – See the previous section.

```
MODEL:
i s |  crave1@-3 crave2@-2 crave4@0 crave6@2 crave8@4 crave10@6 crave12@8;
t16ppdd ON ntx
    i(b1)           ! y on latent intercept (M1)
    s(b2);          ! y on latent slope (M2)
i on ntx(a1);       ! latent intercept (M1) on X
s on ntx(a2);       ! latent slope (M2) on X
i(s2_em1);              ! latent intercept residual variance
s(s2_em2);              ! latent slope residual variance
t16ppdd (s2_ey);        ! outcome variable residual variance
i with t16ppdd (cov1);  ! covariance between intercept and outcome variable
s with t16ppdd (cov2);  ! covariance between slope and outcome variable
i with s;               ! covariance between intercept and slope
MODEL CONSTRAINT:
NEW (indi inds rho1 rho2);      ! define four new quantities
indi=a1*b1;                     ! indirect effect for Mediator 1 (intercept)
inds=a2*b2;                     ! indirect effect for Mediator 2 (slope)
rho1 = cov1/sqrt(s2_em1*s2_ey); ! rho1 is the confounder correlation between
M1 and Y
rho2 = cov2/sqrt(s2_em2*s2_ey); ! rho1 is the confounder correlation between
M1 and Y
```

First, note that this code may not have any additional Mplus command (e.g., OUTPUT, PLOT, etc.) after the `MODEL CONSTRAINT` section. This is because the function `sensitivity_2m_par` described in the next section reads the template file and append additional commands to this code. Second, notice that the labels we used for the path coefficients, variance, and covariances. We recommend you use the same labels. If not, make sure that your labels are consistent throughout the Mplus template. Now, we explain each part of code:

```
i s |  crave1@-3 crave2@-2 crave4@0 crave6@2 crave8@4 crave10@6 crave12@8;
t16ppdd ON ntx
    i(b1)           ! y on latent intercept (M1)
    s(b2);          ! y on latent slope (M2)
i on ntx(a1);       ! latent intercept (M1) on X
s on ntx(a2);       ! latent slope (M2) on X
```

This part of code specifies the two-mediator model, which in our example is the latent growth curve mediation model (LGCMM). This is a fairly standard specification of a latent growth curve model with a linear slope in Mplus. Note that we use the labels $a1$, $a2$, $b1$, and $b2$ to denote the path coefficients that are later used to compute indirect effects.

```
i(s2_em1);              ! latent intercept residual variance
s(s2_em2);              ! latent slope residual variance
t16ppdd (s2_ey);        ! outcome variable residual variance
i with t16ppdd (cov1);  ! covariance between intercept and outcome variable
s with t16ppdd (cov2);  ! covariance between slope and outcome variable
i with s;               ! covariance between intercept and slope
```

This section specifies the variance and covariance for the correlated augmented model. In the first three lines we label the residual variances that will be used later. The last three lines play an important role to specify the correlated augmented model. Note that by default, Mplus fixes these covariances to zero, including the covariance between the latent intercept and slope. In the correlated augmented model, however, we explicitly specify covariances between the residuals to account for the omitted confounder bias, e.g., `cov1` and `cov2`. Note that the values of the two covariances are not and thus the model is not identified. We mentioned earlier that the model specified in the template file cannot be run in Mplus because it is not identified. The function `sensitivity_2m_par` appends additional constraints in the `MODEL CONSTRAINT` section to make the model identified. Also, as mentioned in the manuscript, the covariance between intercept and slope must be *explicitly* specified and freely estimated in Mplus code.

```
MODEL CONSTRAINT:
NEW (indi inds rho1 rho2);      ! define four new quantities
indi=a1*b1;                     ! indirect effect for Mediator 1 (intercept)
inds=a2*b2;                     ! indirect effect for Mediator 2 (slope)
rho1 = cov1/sqrt(s2_em1*s2_ey); ! rho1 is the confounder correlation between
Mediator 1 and Y
rho2 = cov2/sqrt(s2_em2*s2_ey); ! rho2 is the confounder correlation between
Mediator 2 and Y
```

The `MODEL CONSTRAINT` section defines the quantities to be estimated as well as additional constraints. We define two indirect effects for each mediator. Next, we compute `rho1` and `rho2`, correlation between the latent intercept (M1) and the outcome variable and the latent slope (M2) and the outcome variable, respectively. Later, the sensitivity analysis function `sensitivity_2m_par` will read this template file, compute confounder correlation values, and then append constraints for the confounder correlations. For example, for the confounder correlations equal to .1, `sensitivity_2m_par` function appends the following code to the `MODEL CONSTRAINT` section, and then runs the new Mplus code.

```
rho1 = .1;
rho2 = .1;
```

Conducting Sensitivity Analysis

Below, we explain the arguments for the function `sensitivity_2m_par`, which is the main function to conduct sensitivity analysis for a multiple mediation model with two mediators.

- `input_file`: name (including the path) of the Mplus template input file usually with "inp" extension. For example, the name of Mplus template file in our example is `input_file= 'lgcmm_template.inp'`.
- `rho1`: a range of values for the sensitivity confounder parameter ρ_1, which the confounder correlation between the residuals associated with a mediator (e.g., m1) and the outcome variable. For example, `rho1 = c(0,0.5)`.
- `rho2`: a range of values for the sensitivity confounder parameter ρ_2, which the confounder correlation between the residuals associated with a mediator (e.g., m2) and the outcome variable. For example, `rho2 = c(0,0.5)`.
- `lab`: a list of values that would tell R the name of independent variable x, two mediators, m_1 and m_2, the outcome variable y, labels for each of the two indirect effects, $ind1$ and $ind2$. For example, `lab = list(x = "ntx", m1 = "I",m2 = "S",y = "T16PPDD",ind1 = "ind1",ind2 = "ind2")`

The function `sensitivity_2m_par` conducts sensitivity analysis and return a list of two data frames, named `ind1` and `ind2`, that contain point and confidence interval for each of the indirect

effects. For the empirical example in the manuscript, we ran the code below. Because sensitivity analysis runs hundreds of analyses, the process might take a few minutes to complete.

```
rho1 = c(-.5, .5)    # range of value for the confounder correlation rho1
rho2 = c(-.5, .5)    # range of value for the confounder correlation rho2
## Argument "lab", we declare the labels we used in Mplus code to name variab
les
lab = list(
x = "ntx",           # name of the independent variable we used in Mplus code
m1 = "I",            # name of the mediator 1 (latent intercept variable)
m2 = "S",            # name of the mediator 2 (latent intercept slope)
y = "T16PPDD",       # name of the outcome variable
ind1 = "ind1",       # label for indirect effect through mediator 1
ind2 = "ind2"        # label for indirect effect through mediator 1
)

## the data frame "res_df" contain the results of the sensitivity analysis
res_df <-
sensitivity_2m_par(
input_file = "lgcmm_template.inp",
rho1 = rho1,
rho2 = rho2,
lab = lab
)

## A glimpse of the sensitivity analysis results for mediator 1
dplyr::glimpse(res_df$ind1)
```

Sensitivity Confidence Band Plot

The function `plot_2m_par` accepts the sensitivity analysis data generated from the function `sensitivity_2m_par` and computes two sets of sensitivity confidence band plots for each indirect effect and saves each plot in a separate "png" graphic file, with a name that is combined of "sensitivity_plot" the label for a mediator, e.g., "sensitivity_plotm1.png". Below we describe each argument of the function:

- `x`: sensitivity analysis results generated by the sensitivity_2m_par function. Note that the result is a list of two data frames.
- `rho1_facet`: facet (panel) values for the confounder correlation between Mediator 1 and the outcome variable
- `rho2_facet`: facet (panel) confounder correlation between Mediator 2 and the outcome variable
- `xlim1`: x-axis limits for the plot of indirect effect through Mediator 1

- `xlim2`: x-axis limits for the plot of indirect effect through Mediator 2
- `ylim1`: y-axis limits for the plot of indirect effect through Mediator 1
- `ylim2`: y-axis limits for the plot of indirect effect through Mediator 2
- `nrow`: the number of rows for the matrix of plots
- `ncol`: the number of columns for the matrix of plots. Default is 2.
- `device`: type of file to save the plot: 'pdf', 'png', etc. Default is 'png'.

We use the code below to produce the sensitivity confidence band plots in the manuscript.

```
plot_2m_par(x = res_df, rho1_facet = c(0, 0.1, 0.3, 0.5), rho2_facet = c(0,
    0.1, 0.3, 0.5), xlim1 = c(0.1, 0.5), xlim2 = c(0.1, 0.5), ylim1 = c(-0.4,

    0.2), ylim2 = c(-0.2, 0.2))
```

Sensitivity Contour Plot

Below we show the code to produce sensitivity contour plots. We use the function `contourplot` from the `lattice` package, which comes standard with R. Note that each contour plot uses one of the two data frames, not the list of the two data frames, generated by the `sensitivity_2m_par` function.

```
## Sensitivity contour plot for mediator 1
lattice::contourplot(
ind ~ rho1 * rho2,
data = res_df$ind1,          # data frame results generated by the sensitivity
_2m_par() function
xlab = expression(rho[1]),
ylab = expression(rho[2]),
cuts = 12,
region = TRUE,
col.regions = heat.colors,
panel = latticeExtra::panel.2dsmoother
)

## Sensitivity contour plot for mediator 2
lattice::contourplot(
ind ~ rho2 * rho1,
data = res_df$ind2,        # data frame results generated by the sensitivity_2
m_par() function
xlab = expression(rho[2]),
ylab = expression(rho[1]),
cuts = 12,
region = TRUE,
```

```
col.regions = heat.colors,
panel = latticeExtra::panel.2dsmoother
)
```