# Supplemental Material for Brick & Bailey (submitted)

This document carries the code and output used in Brick and Bailey (submitted). Specifically, we illustrate the use of Matrices of Implied Causation using a mediation model and three examples from the literature. They are:

0. A dummy mediation model
1. Three models from Tomarken & Waller (2003), in this case simplex model variants in their Example 1.
2. CLPM in comparison to RI-CLPM
3. RI-CLPM intervention
4. Comparison of alternate specifications of CLPM
5. Complete MIC tables for CLPM models

Note that the MICr package is considered to be in Beta testing at the time of publication. Changes to the user interface and/or fixes to bugs that may exist in the code are likely, and this supplemental entry may not reflect the most recent version of the package. Please see the documentation, including the vignette "BrickBaileyExamples" in the latest version of the package for a complete discussion.

## Setup and packages

First we need MICr, OpenMx, and also some tools for plotting. MICr can be downloaded from github using devtools, if needed.

```
# install.packages(d evtools)
# library(devtools)
# install_github("trbrick/MICr")
library(MICr)
library(OpenMx)
# Turn on plots if semPlot is available:
if(library(semPlot, logical.return=TRUE))  drawPlots<-TRUE else drawPlots <- FALSE
```

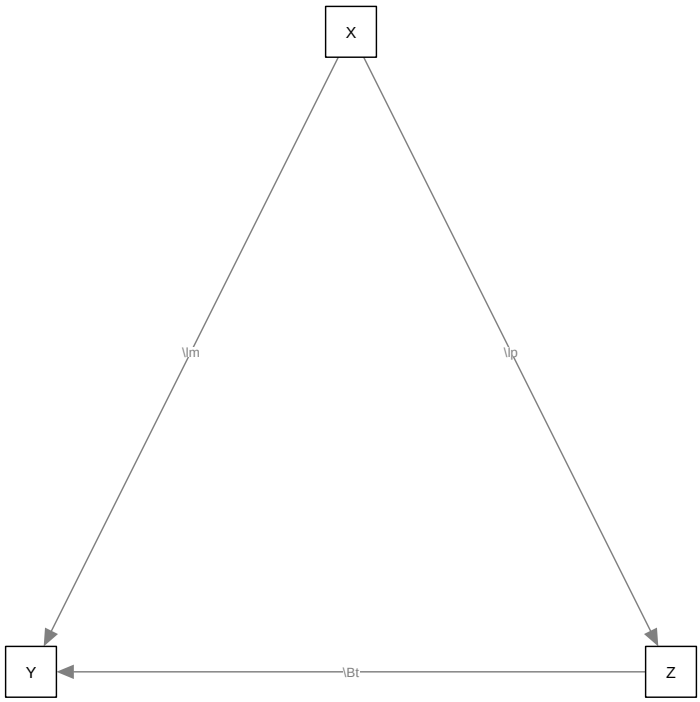
## Example 0: Pedagogical mediation model

The first is a dummy example:

```
XYZmodel <- mxModel("XYZ", manifestVars=c("X", "Y", "Z"), type="RAM",
  mxPath(from="Z", to="Y", values=.6, labels="\\Beta"),
  mxPath(from="X", to=c("Y", "Z"), values=c(.2, .8), labels=c("\\lambda", "\\alpha")),
  mxMatrix("Iden",3,name="I"))
```

Whenever we run a model like this, I'll add a sempaths plot. These models are complex, so the plot generated there is rarely a good view of what the model should look like. The figures in the paper itself are a better view of the models, but for generation purposes we create them here. They are generated with Onyx (see http://onyx.brandmaier.de/ for more), although some are modified for readability.

```
if(drawPlots) semPaths(XYZmodel)
```




We can render the MIC for this model directly.

```
MIC(XYZmodel)
```

```
     X Y   Z
X 1.00 0 0.0
Y 0.68 1 0.6
Z 0.80 0 1.0
```

The MIC is read like the `A` matrix in a RAM model: the path from X to Z is in the column for X and the row for Z (here, it is .8). As expected, this shows the total effect–the model-implied causal influence–from each variable to each other variable.

This can be rendered more succinctly as a MIC Table:

```
MICTable(XYZmodel)
```

```
Table: Implied Causation Table: XYZ

from   to     XYZ
-----  ---  -----
X      Z     0.80
X      Y     0.68
```

```
Z      Y     0.60
```

# Example 1: Three models from Tomarken & Waller (2003)

The first example shows three cases from Tomarken & Waller's 2003 paper. These three models fit identically, but make very different causal predictions.

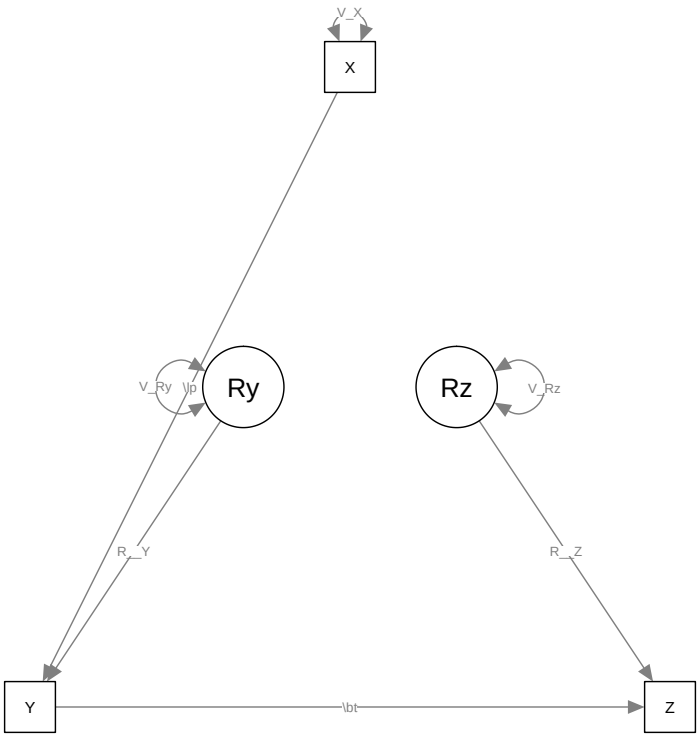
## Tomarken Example 1A:

The first model is a

```
tomarken1A <- mxModel("Tomarken1A",
  type="RAM", manifestVars = c("X", "Y", "Z"), latentVars = c("Ry", "Rz"),

  # Latent to manifest paths
  mxPath(from="Ry", to="Y",  free=FALSE, value=.6, label=c("Ry_to_Y") ),
  mxPath(from="Rz", to="Z",  free=FALSE, value=.6, label=c("Rz_to_Z") ),

  # Manifest chain
  mxPath(from="X",  to="Y",  free=TRUE, values=0.8, labels=c("\\alpha") ),
  mxPath(from="Y",  to="Z",  free=TRUE, values=0.8, labels=c("\\beta") ),

  # Variances
  mxPath(from=c("Ry", "Rz", "X"),
         arrows=2, free=TRUE, values=1.0,
         labels=c("V_Ry", "V_Rz", "V_X") )
)
```

Sempaths plots of these models show the differences fairly well.

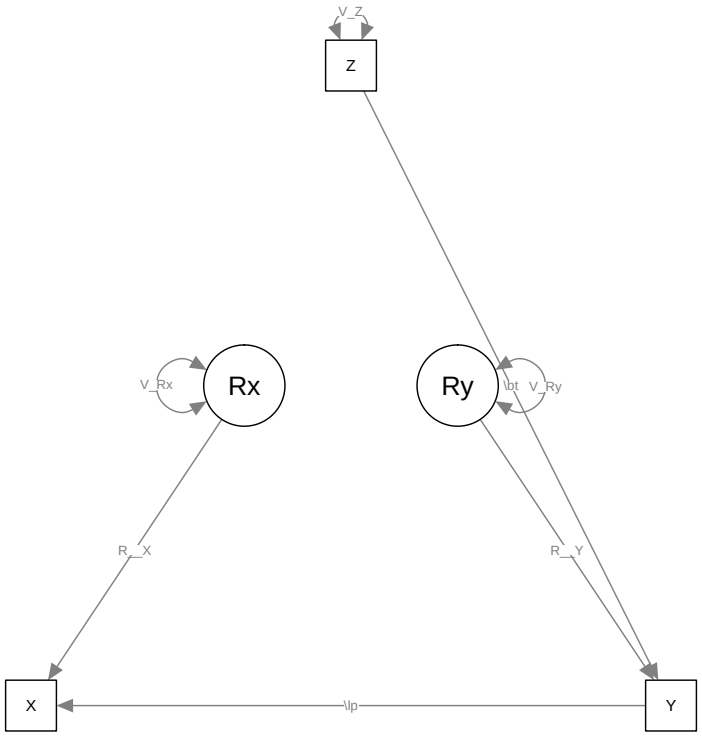
```
if(drawPlots) semPaths(tomarken1A)
```


V_X
X
V_Ry
\lp
Ry
Rz
V_Rz
R_Y
R_Z
Y
\bt
Z

## Tomarken Example 1B

Model 1B resembles model 1A, but the chain flows in the opposite direction, with Z predicting Y and Y predicting X.

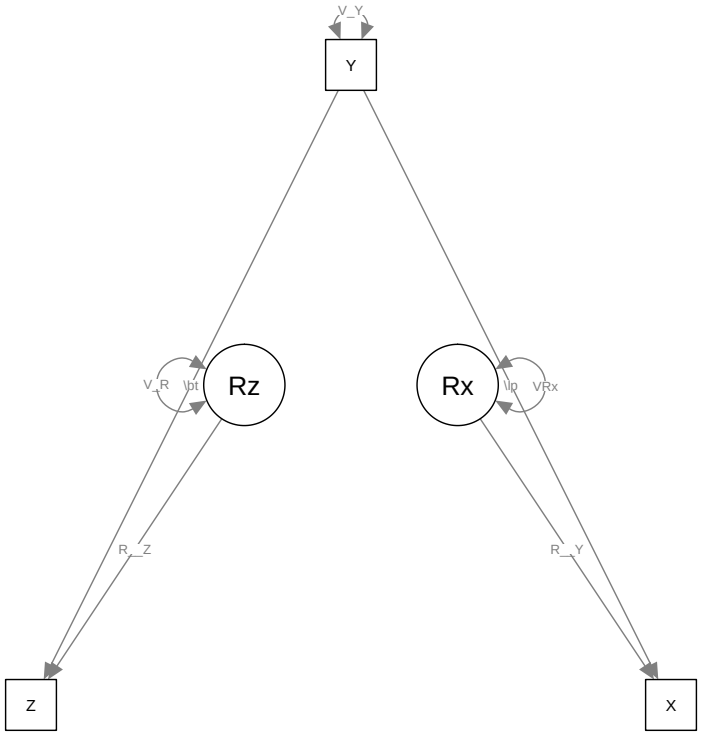
```
tomarken1B <- mxModel("Tomarken1B",
  type="RAM", manifestVars = c("X","Y","Z"), latentVars = c("Rx", "Ry"),

    # Latent to manifest paths
  mxPath(from="Ry", to="Y",  free=FALSE, value=.6, label=c("Ry_to_Y") ),
  mxPath(from="Rx", to="X",  free=FALSE, value=.6, label=c("Rx_to_X") ),

  # Manifest chain
  mxPath(from="Z",  to="Y",  free=TRUE, values=0.8, labels=c("\\beta") ),
  mxPath(from="Y",  to="X",  free=TRUE, values=0.8, labels=c("\\alpha") ),

  # Variances
  mxPath(from=c("Ry", "Rx", "Z"),
         arrows=2, free=TRUE, values=1.0,
         labels=c("V_Ry", "V_Rx", "V_Z") )
)
# Throws a warning because the model has not been run.
if(drawPlots) semPaths(tomarken1B)
```



## Tomarken Example 1C

The last Tomarken example model is a fork, with Y predicting X and Z.

```
tomarken1C <- mxModel("Tomarken1C",
    type="RAM", manifestVars = c("X","Y","Z"), latentVars = c("Rz","Rx"),

    # Fork loadings
    mxPath(from="Y", to=c("Z","X"), free=TRUE, values=.8, labels=c("\\beta","\\alpha") ),

    # Latent residual terms
    mxPath(from="Rz", to="Z",  free=TRUE, value=.6, label="Rz_to_Z" ),
    mxPath(from="Rx", to="X",  free=TRUE, value=.6, label="Ry_to_Y" ),

    mxPath(from=c("Y", "Rx", "Rz"),
            arrows=2, free=FALSE, values=1.0,
           labels=c("V_Y", "VRx", "V_Rz"))
    )
# Throws a warning because the model has not been run.
if(drawPlots) semPaths(tomarken1C)
```



### Tomarken MICs

The MICs can be rendered directly as matrices.

```
MIC(tomarken1A)
```

```
      X   Y Z   Ry  Rz
X  1.00 0.0 0 0.00 0.0
Y  0.80 1.0 0 0.60 0.0
Z  0.64 0.8 1 0.48 0.6
Ry 0.00 0.0 0 1.00 0.0
Rz 0.00 0.0 0 0.00 1.0
```

```
MIC(tomarken1B)
```

```
   X   Y    Z  Rx   Ry
X  1 0.8 0.64 0.6 0.48
Y  0 1.0 0.80 0.0 0.60
Z  0 0.0 1.00 0.0 0.00
Rx 0 0.0 0.00 1.0 0.00
Ry 0 0.0 0.00 0.0 1.00
```

```
MIC(tomarken1C)
```

```
   X   Y Z  Rz  Rx
X  1 0.8 0 0.0 0.6
Y  0 1.0 0 0.0 0.0
Z  0 0.8 1 0.6 0.0
Rz 0 0.0 0 1.0 0.0
Rx 0 0.0 0 0.0 1.0
```

A more succinct approach is to plot the MICTable that compares all three. Because the residuals are not of particular interest, we focus only on the influences from and to X, Y, and Z. Note that this table differs in its sorting function from the one displayed in the paper.

```
MICTable(tomarken1A, tomarken1B, tomarken1C, from=c("X", "Y", "Z"), to=c("X", "Y", "Z"))
```

```
Table: Implied Causation Table: Tomarken1A, Tomarken1B, Tomarken1C Existence & Sign & Scale

from   to    Tomarken1A   Tomarken1B   Tomarken1C
-----  ---  -----------  -----------  -----------
X      Y           0.80         0.00          0.0
Y      Z           0.80         0.00          0.8
X      Z           0.64         0.00          0.0
Y      X           0.00         0.80          0.8
Z      X           0.00         0.64          0.0
Z      Y           0.00         0.80          0.0
```

## Example 2: CLPM vs. RI-CLPM

Example 2 comes from Bailey et al. (2018). Three fairly intricate models, and we'll do one better by dropping the person-level intercepts.

First, need the data and the correlation matrix–it's in the paper, luckily.

```
means <- c(68.08, 93.09, 107.40, 116.80, 49.91, 73.87, 90.65, 102.87)
times <- c("K", 1:3)
nTimes <- length(times)
```

```
manifests <- paste(rep(c("Read", "Math"), each=nTimes), rep(times, 2), sep="")
occasions <- paste(rep(c("OR", "OM"), each=nTimes), rep(times, 2), sep="")
sds <- c(14.17, 18.02, 15.51, 14.82, 12.84, 17.64, 16.66, 15.63)
corMat <- matrix(c(
1.00, 0.79, 0.71, 0.64, 0.73, 0.66, 0.62, 0.59,
0.79, 1.00, 0.86, 0.78, 0.71, 0.73, 0.70, 0.67,
0.71, 0.86, 1.00, 0.85, 0.69, 0.71, 0.74, 0.69,
0.64, 0.78, 0.85, 1.00, 0.67, 0.70, 0.73, 0.72,
0.73, 0.71, 0.69, 0.67, 1.00, 0.83, 0.78, 0.75,
0.66, 0.73, 0.71, 0.70, 0.83, 1.00, 0.86, 0.82,
0.62, 0.70, 0.74, 0.73, 0.78, 0.86, 1.00, 0.88,
0.59, 0.67, 0.69, 0.72, 0.75, 0.82, 0.88, 1.00), nrow=8)
covMat <- diag(sds) %&% corMat
names(means) <- manifests
dimnames(covMat) <- list(manifests, manifests)
dimnames(corMat) <- list(manifests, manifests)
```

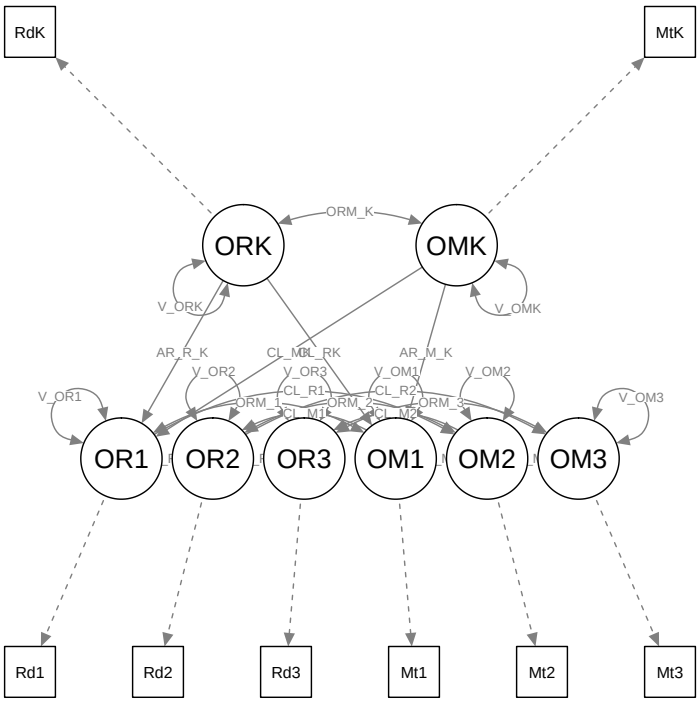
## CLPM1

The first model is a traditional CLPM model. In order to align the names of variables between the two CLPMs, we specify the CLPM to align more closely with the RI-CLPM in the next model. In a later section, we demonstrate that this model is precisely equivalent to a traditional CLPM formulation. Here, though, we create OM and OR variables that represent the latent values underlying the observed OM/OR paths. The autoregressive paths follow from OMK->OM1, OM1->OM2, etc., rather than MathK->Math1. For simplicity in adapting the model later, we also include the latent RI variables, although these have their loadings on Math and Reading fixed to zero so that they do not influence the expectations, implications, or fit of the model.

```
ModelCLPM1 <- mxModel("CLPM1", type="RAM", manifestVars = manifests,
                      latentVars=c("Reading", "Math", occasions),

                      # OR/OM -> Math/Reading paths
                      mxPath(from=occasions, to=manifests,
                             arrows=1, values=1, free=FALSE),

                      # Occasion variances
                      mxPath(from=occasions, arrows=2, values=1, free=TRUE,
                             labels=paste0("V_", occasions), lbound=.1),

                      # Covariation of error/process
                      mxPath(from=occasions[1:4], to=occasions[5:8],
                             arrows=2, free=TRUE, values=.25,
                             labels=paste0("ORM_", times)),

                      # AR of Reading
                      mxPath(from=occasions[1:3], to=occasions[2:4],
                             arrows=1, free=TRUE, values=.5,
                             labels=paste0("AR_R_", times[1:3])),

                      # AR of Math
                      mxPath(from=occasions[5:7], to=occasions[6:8],
                             arrows=1, free=TRUE, values=.3,
                             labels=paste0("AR_M_", times[1:3])),
```

```
                    # Cross-lagged R->M
                    mxPath(from=occasions[1:3], to=occasions[6:8],
                           arrows=1, free=TRUE, values=.10,
                           labels=paste0("CL_R", times[1:3], "_M", times[2:4])),

                    # Cross-lagged M->R
                    mxPath(from=occasions[5:7], to=occasions[2:4],
                           arrows=1, free=TRUE, values=.03,
                           labels=paste0("CL_M", times[1:3], "_R", times[2:4])),

                    mxData(type="cor", observed = corMat, numObs = 9612)
)
CLPM1 <- mxTryHard(ModelCLPM1)
```

```
if(drawPlots) semPaths(CLPM1)
```



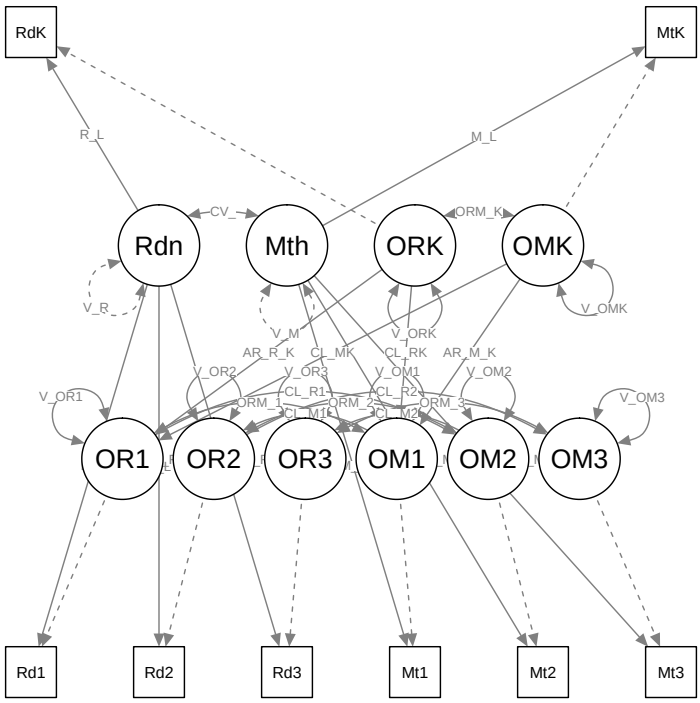

We demonstrate the equivalence of the two formulations of the CLPM in a later section of the supplemental material.

## CLPM2

CLPM2 is modified from CLPM1, and adds loadings and a free covariance for the latent intercepts so that they have influence on the model. Because they now have paths to the model, these elements contribute to fit.

```
ModelCLPM2 <- mxModel(CLPM1,

                      # RI factor loadings (fixed across time)
                       mxPath(from="Reading", to=paste0("Read", times),
                              values=1, free=TRUE, labels=paste0("Read_Loadings")),
                       mxPath(from="Math", to=paste0("Math", times),
                              values=1, free=TRUE, labels=paste0("Math_Loadings")),

                      # Free RI covariance; RIs are standardized for identification.
                      mxPath(from=c("Reading", "Math"),
                             arrows=2, connect="unique.pairs", free=c(FALSE, TRUE, FALSE),
                             values=c(1,.8, 1), labels=c("V_R", "CV_RM", "V_M")),

                      name="CLPM2: RI-CLPM"
)
CLPM2 <- mxTryHard(ModelCLPM2)
```

```
if(drawPlots) semPaths(CLPM2)
```




## CLPM MICs

Once we have the details of more specific models, we can examine them in detail to compare the differences. We are specifically interested in the effects of Kindergarten occasion measurements on the final outcomes.

```
MICTable(CLPM1, CLPM2, from=c("OMK", "ORK"),
         to=paste0(rep(c("Read", "Math"), each=3), 1:3), se=FALSE)
```

Table: Implied Causation Table: CLPM1, CLPM2: RI-CLPM

| from | to | CLPM1 | CLPM2: RI-CLPM |
|---|---|---|---|
| OMK | Math1 | 0.7454506 | 0.2353754 |
| OMK | Math2 | 0.6010844 | 0.0928140 |
| ORK | Read1 | 0.5816741 | 0.4725900 |
| OMK | Math3 | 0.5199614 | 0.0484700 |
| ORK | Read2 | 0.4458971 | 0.3108500 |
| OMK | Read3 | 0.3669879 | 0.0384980 |
| ORK | Read3 | 0.3447392 | 0.1939855 |
| OMK | Read2 | 0.3399478 | 0.0446695 |
| OMK | Read1 | 0.2853778 | 0.0522898 |
| ORK | Math3 | 0.1823172 | 0.0379357 |
| ORK | Math2 | 0.1764470 | 0.0646917 |
| ORK | Math1 | 0.1158210 | 0.0611456 |

The specific details of these comparisons are described in greated detail in the paper. Of note, the autoregressive and cross-lagged influences differ strongly between the two, suggesting causal differences that may help to distinguish between the models, even though the fit statistics are somewhat similar.

The complete MICs for these two models is displayed at the end of the document.

## Example 3: Adding Interventions

In order to examine our hypothetical interventions, we recreate the same CLPM model, but this time add three new manifest variables: `XRead`, `ReadStudy`, and `AllStudy`, representing the three hypothesized interventions. Because these variables are independent, they do not influence each others' causal estimates, and can be included in the model simultaneously.

```
# Interventions:
ModelIxn<- mxModel("InterventionModel", type="RAM",
                manifestVars = c(manifests, "XRead", "ReadStudy", "AllStudy"),
                latentVars=c("Reading", "Math", occasions),

                # RI factor loadings
                mxPath(from="Reading", to=paste0("Read", times),
                       arrows=1, values=1, free=TRUE,
                       labels=paste("Read", times, sep="_")),
                mxPath(from="Math", to=paste0("Math", times),
                       arrows=1, values=1, free=TRUE,
                       labels=paste("Math", times, sep="_")),

                # Latent math/reading paths
                mxPath(from=occasions, to=manifests, arrows=1, values=1, free=FALSE),

                # Occasion variances
                mxPath(from=occasions, arrows=2, values=1, free=TRUE,
                       labels=paste0("V_", occasions), lbound=.1),

                # RI Var/Covar.  Icpts standardized by definition.
```

```
                mxPath(from=c("Reading", "Math"),
                       arrows=2, connect="unique.pairs", values=c(1,.8, 1),
                       free=c(FALSE, TRUE, FALSE), labels=c("VR", "CV_RM", "VM")),

                # Covariation of error/process
                mxPath(from=occasions[1:4], to=occasions[5:8], arrows=2,
                       free=TRUE, values=.25, labels=paste0("ORM_", times)),

                # AR of Reading
                mxPath(from=occasions[1:3], to=occasions[2:4], arrows=1,
                       free=TRUE, values=.5, labels=paste0("AR_R_", times[1:3])),

                # AR of Math
                mxPath(from=occasions[5:7], to=occasions[6:8], arrows=1,
                       free=TRUE, values=.3, labels=paste0("AR_M_", times[1:3])),

                # Cross-lagged R->M
                mxPath(from=occasions[1:3], to=occasions[6:8], arrows=1,
                       free=TRUE, values=.10,
                       labels=paste0("CL_R", times[1:3], "_M", times[2:4])),

                # Cross-lagged M->R
                mxPath(from=occasions[5:7], to=occasions[2:4], arrows=1,
                       free=TRUE, values=.03,
                       labels=paste0("CL_M", times[1:3], "_R", times[2:4])),

                # RI factor loadings
                mxPath(from="Reading", to=paste0("Read", times), arrows=1,
                       values=1, free=TRUE, labels=paste0("Read_Loadings")),
                mxPath(from="Math", to=paste0("Math", times), arrows=1,
                       values=1, free=TRUE, labels=paste0("Math_Loadings"))
        )
```

Because the interventions are manifest variables in the model, we will need to give them each a variance. Asymptotically, the appropriate variance for a balanced intervention coded 0/1 is .25, but for completeness, we compute it empirically:

```
aValue <- rep(c(0,1), each=9612/2) # Create a vector of length N that is half 0s and half 1s.
var(aValue) # compute its variance: approximately .25
```

```
[1] 0.250026
```

We now add the interventions themselves to the model, including both our estimates of the intervention variances and their influence on each subsequent timepoint.

```
ModelIxn<- mxModel(ModelIxn,
                # Variance of manipulations:
                mxPath(from=c("XRead", "ReadStudy", "AllStudy"), arrows=2,
                       values=.25, free=TRUE,
                       labels=paste0("V_", c("XRead", "ReadStudy", "AllStudy"))),

                # Xread influences only Occasion reading at grade 1
                mxPath(from="XRead", to="OR1",
                       values=1, free=TRUE),
                # ReadStudy influences reading at both grades 1 and 2, but less.
```

```
                mxPath(from="ReadStudy", to=c("OR1",  "OR2"),
                       values=.6, free=TRUE),
                # AllStudy influence reading and math at grades 1 and 2, but even less
                mxPath(from="AllStudy", to=c("OM1", "OM2", "OR1",  "OR2"),
                       values=c(.4, .3, .4, .3),
                       free=TRUE)
)
```

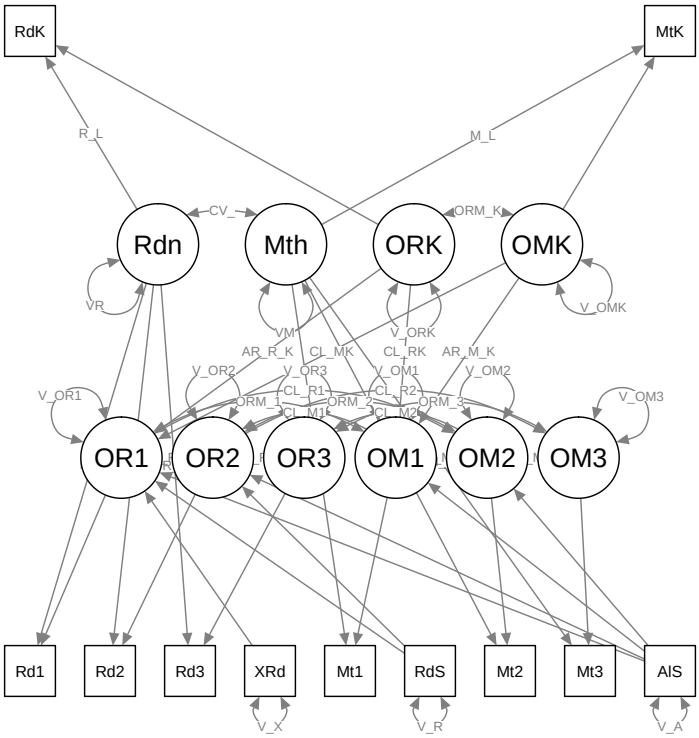
To save ourselves typing, we propagate the parameter estimates we got from the RI-CLPM when fitted to real data. This could be done by including these values as starting values in the model above, but here it's done automatically for us. Note that this requires that parameters are named identically between the models.

```
# Get parameters from the fitted model
ixnParams <- omxGetParameters(CLPM2)

# Set to fitted parameters
ModelIxn <- omxSetParameters(ModelIxn, names(ixnParams), ixnParams, free=TRUE)
```

Again, we can plot this, but it's sufficiently complicated that it's difficult to see all the details in the plot; see Figure 4 in the paper for a better rendering.

```
if(drawPlots) semPaths(ModelIxn)
```

```
Warning in OpenMx::mxStandardizeRAMpaths(object, SE = TRUE): standard errors
will not be computed because model 'InterventionModel' has not yet been run, and
no matrix was provided for argument 'cov'
```


RdK
MtK
R_L
M_L
CV_
ORM_K
Rdn
Mth
ORK
OMK
VR
V_OMK
VM
V_ORK
AR_R_K
CL_MK
CL_RK
AR_M_K
V_OR2
V_OR3
V_OM1
V_OM2
V_OR1
CL_R1
CL_R2
V_OM3
ORM_1
ORM_2
ORM_3
CL_M1
CL_M2
OR1
OR2
OR3
OM1
OM2
OM3
Rd1
Rd2
Rd3
XRd
Mt1
RdS
Mt2
Mt3
AlS
V_X
V_R
V_A

Of particular import for this comparison, we can look at the influence of the three interventions based on this model, including standard errors computed using the delta method.

```
MICTable(ModelIxn, from=c("XRead", "ReadStudy", "AllStudy"), to=c("Read3"), N=9612)
```

```
Table: Implied Causation Table: InterventionModel

from        to       InterventionModel   InterventionModel_SE
----------  ------  ------------------  ---------------------
ReadStudy   Read3            0.5990476              0.0103816
AllStudy    Read3            0.4075670              0.0109620
XRead       Read3            0.4008324              0.0087279
```

From here, we can clearly see that for reading at grade 3, the ReadStudy intervention has the largest model-implied causal influence on the outcome. Note once again that this may not be the whole story.

We can also draw out the size of the standard errors relative to the difference in effects so that we could approximate the N needed for a standard error small enough to differentiate them. Note that the two interventions are just under .007 apart, so we need a standard error just under .0035 (or .007/1.96) to distinguish these at the .01 level.

```
simMIC <- MICTable(ModelIxn, from=c("XRead", "ReadStudy", "AllStudy"), to=c("Read3"), N=9612)
```

```
Table: Implied Causation Table: InterventionModel

from        to       InterventionModel   InterventionModel_SE
----------  ------  ------------------  ---------------------
ReadStudy   Read3            0.5990476              0.0103816
AllStudy    Read3            0.4075670              0.0109620
XRead       Read3            0.4008324              0.0087279
```

```
targetSE <- (simMIC$InterventionModel[simMIC$from=="AllStudy"] - simMIC$InterventionModel[simMIC$from==
```

Note that standard errors for SEMs are frequently not the best approach to determine confidence intervals and therefore significance or power. Likelihood profile confidence intervals (computed using `mxCI()`) would be a better choice. Further, please note that in many cases, it may be a good idea to think hard about how much precision we have in our computations.

However, since this data is purely simulated, we can just crank up OpenMx's functional precision and assume that our covariance matrix is that precise.

```
mxOption(NULL, "Function Precision", 1e-18)
mxOption(NULL, "Optimality Tolerance", 1e-18)
Nvals <- 62020:62030  # In the ballpark.

# Get all SEs
SE <- sapply(Nvals, function(n) {MICTable(ModelIxn,
                                          from=c("XRead", "ReadStudy", "AllStudy"), to=c("Read3"), N=n,
                                          print=FALSE)$InterventionModel_SE[3]})

# Find the ones where it crosses the target
min(Nvals[which(SE < targetSE)])
```

```
[1] 62027
```

So, if we're confident in our precision, we need 62027 participants to distinguish these. Be aware that the differences between subsequent N sizes are on the order of 1e-8, which is quite small, and that depending on

the precision of our original estimates, some rounding may be in order.

```
# Reset OpenMx Options
mxOption(NULL, "Function Precision", reset=TRUE)
mxOption(NULL, "Optimality Tolerance", reset=TRUE)
```

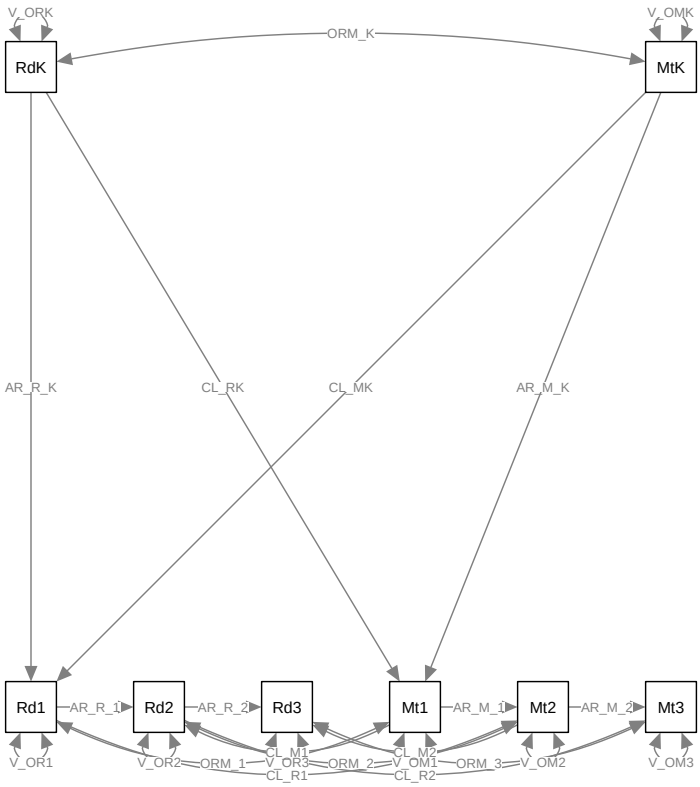
## Specification of the CLPM following Figure 2

The CLPM diagram provided in Figure 2 in the paper is drawn in the traditional manner for a CLPM model, which differs slightly from the specification above. Although the model specifications differ slightly, the models are precisely equivalent. The specification used above matches Bailey, et al., 2018, and is more directly comparable to the RI-CLPM that follows.

To demonstrate the equivalence between CLPM1 above and the diagrammed specification, we here generate CLPM1B, the Figure 2 specification, and compare the model estimates to model CLPM1 (specified following Bailey et al. 2018) as it is specified above.

```
ModelCLPM1B <- mxModel("CLPM1B", type="RAM", manifestVars = manifests,
                       latentVars=c("Reading", "Math"),

                       # Manifest variances
                       mxPath(from=manifests, arrows=2, values=1, free=TRUE,
                              labels=paste0("V_", occasions), lbound=.1),

                       # Covariation of error/process
                       mxPath(from=manifests[1:4], to=manifests[5:8],
                              arrows=2, free=TRUE, values=.25,
                              labels=paste0("ORM_", times)),

                       # AR of Reading
                       mxPath(from=manifests[1:3], to=manifests[2:4],
                              arrows=1, free=TRUE, values=.5,
                              labels=paste0("AR_R_", times[1:3])),

                       # AR of Math
                       mxPath(from=manifests[5:7], to=manifests[6:8],
                              arrows=1, free=TRUE, values=.3,
                              labels=paste0("AR_M_", times[1:3])),

                       # Cross-lagged R->M
                       mxPath(from=manifests[1:3], to=manifests[6:8],
                              arrows=1, free=TRUE, values=.10,
                              labels=paste0("CL_R", times[1:3], "_M", times[2:4])),

                       # Cross-lagged M->R
                       mxPath(from=manifests[5:7], to=manifests[2:4],
                              arrows=1, free=TRUE, values=.03,
                              labels=paste0("CL_M", times[1:3], "_R", times[2:4])),

                       mxData(type="cor", observed = corMat, numObs = 9612)
)
CLPM1B <- mxTryHard(ModelCLPM1B)
```

```
if(drawPlots) semPaths(CLPM1B)
```




These two models are precisely equivalent. The only matter important to this paper is that the parameter estimates are identical; here we use `omxCheckIdentical()`, which will throw an error if the two vectors are not exactly the same to within machine precision (e.g. precise to within 1*10^-16).

```
# Throws an error if the results are not identical
omxCheckIdentical(omxGetParameters(CLPM1), omxGetParameters(CLPM1B))
```

No error; they are identical.

## Complete MIC Tables for CLPM models

The complete MIC Tables for the CLPM and RI-CLPM are provided here for completeness.

```
MICTable(CLPM1, CLPM2, splitByType=FALSE, caption="Complete MIC Table for CLPM Examples")
```

Table: Complete MIC Table for CLPM Examples

| from | to | CLPM1 | CLPM2: RI-CLPM | CLPM1_SE | CLPM2: RI-CLPM_SE |
|---|---|---|---|---|---|
| ORK | ReadK | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| OR1 | Read1 | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| OR2 | Read2 | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| OR3 | Read3 | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| OMK | MathK | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |

| | | | | | |
|---|---|---|---|---|---|
| OM1 | Math1 | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| OM2 | Math2 | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| OM3 | Math3 | 1.0000000 | 1.0000000 | 0.0000000 | 0.0000000 |
| OM2 | Math3 | 0.8165340 | 0.5150840 | 0.0071495 | 0.0157178 |
| OM2 | OM3 | 0.8165340 | 0.5150840 | 0.0071495 | 0.0157178 |
| OM1 | Math2 | 0.7471633 | 0.3746827 | 0.0074506 | 0.0218472 |
| OM1 | OM2 | 0.7471633 | 0.3746827 | 0.0074506 | 0.0218472 |
| OMK | Math1 | 0.7454506 | 0.2353754 | 0.0082397 | 0.0209981 |
| OMK | OM1 | 0.7454506 | 0.2353754 | 0.0082397 | 0.0209981 |
| OR1 | Read2 | 0.7315350 | 0.6519428 | 0.0074011 | 0.0140989 |
| OR1 | OR2 | 0.7315350 | 0.6519428 | 0.0074011 | 0.0140989 |
| OR2 | Read3 | 0.6847922 | 0.5975803 | 0.0076570 | 0.0138304 |
| OR2 | OR3 | 0.6847922 | 0.5975803 | 0.0076570 | 0.0138304 |
| OM1 | Math3 | 0.6251771 | 0.1936602 | 0.0077443 | 0.0148549 |
| OM1 | OM3 | 0.6251771 | 0.1936602 | 0.0077443 | 0.0148549 |
| OMK | Math2 | 0.6010844 | 0.0928140 | 0.0079652 | 0.0119487 |
| OMK | OM2 | 0.6010844 | 0.0928140 | 0.0079652 | 0.0119487 |
| ORK | Read1 | 0.5816741 | 0.4725900 | 0.0086745 | 0.0162279 |
| ORK | OR1 | 0.5816741 | 0.4725900 | 0.0086745 | 0.0162279 |
| OR1 | Read3 | 0.5354579 | 0.4008324 | 0.0075255 | 0.0159881 |
| OR1 | OR3 | 0.5354579 | 0.4008324 | 0.0075255 | 0.0159881 |
| OMK | Math3 | 0.5199614 | 0.0484700 | 0.0075421 | 0.0067132 |
| OMK | OM3 | 0.5199614 | 0.0484700 | 0.0075421 | 0.0067132 |
| ORK | Read2 | 0.4458971 | 0.3108500 | 0.0079066 | 0.0160383 |
| ORK | OR2 | 0.4458971 | 0.3108500 | 0.0079066 | 0.0160383 |
| OMK | Read3 | 0.3669879 | 0.0384980 | 0.0074757 | 0.0110517 |
| OMK | OR3 | 0.3669879 | 0.0384980 | 0.0074757 | 0.0110517 |
| ORK | Read3 | 0.3447392 | 0.1939855 | 0.0070606 | 0.0133847 |
| ORK | OR3 | 0.3447392 | 0.1939855 | 0.0070606 | 0.0133847 |
| OMK | Read2 | 0.3399478 | 0.0446695 | 0.0081560 | 0.0165260 |
| OMK | OR2 | 0.3399478 | 0.0446695 | 0.0081560 | 0.0165260 |
| OM1 | Read3 | 0.2873163 | 0.0745129 | 0.0075369 | 0.0143892 |
| OM1 | OR3 | 0.2873163 | 0.0745129 | 0.0075369 | 0.0143892 |
| OMK | Read1 | 0.2853778 | 0.0522898 | 0.0086746 | 0.0218221 |
| OMK | OR1 | 0.2853778 | 0.0522898 | 0.0086746 | 0.0218221 |
| OM2 | Read3 | 0.2232537 | 0.1271825 | 0.0076570 | 0.0150321 |
| OM2 | OR3 | 0.2232537 | 0.1271825 | 0.0076570 | 0.0150321 |
| OR1 | Math3 | 0.1889522 | 0.0552154 | 0.0077346 | 0.0125919 |
| OR1 | OM3 | 0.1889522 | 0.0552154 | 0.0077346 | 0.0125919 |
| ORK | Math3 | 0.1823172 | 0.0379357 | 0.0071315 | 0.0079836 |
| ORK | OM3 | 0.1823172 | 0.0379357 | 0.0071315 | 0.0079836 |
| ORK | Math2 | 0.1764470 | 0.0646917 | 0.0077062 | 0.0103196 |
| ORK | OM2 | 0.1764470 | 0.0646917 | 0.0077062 | 0.0103196 |
| OM1 | Read2 | 0.1759794 | 0.0449477 | 0.0074011 | 0.0185441 |
| OM1 | OR2 | 0.1759794 | 0.0449477 | 0.0074011 | 0.0185441 |
| OR1 | Math2 | 0.1545707 | 0.0884097 | 0.0074506 | 0.0135940 |
| OR1 | OM2 | 0.1545707 | 0.0884097 | 0.0074506 | 0.0135940 |
| ORK | Math1 | 0.1158210 | 0.0611456 | 0.0082397 | 0.0122218 |
| ORK | OM1 | 0.1158210 | 0.0611456 | 0.0082397 | 0.0122218 |
| OR2 | Math3 | 0.0857648 | 0.0148432 | 0.0071495 | 0.0115213 |
| OR2 | OM3 | 0.0857648 | 0.0148432 | 0.0071495 | 0.0115213 |