#Section 2 - Ping 2 Step - The 2 "steps" in Ping's procedure are to run the additive model (step 1) and to then use output from this to fix values in the full model (step 2). For purposes of accomplishing this in R, we have broken this down into several steps. There are different options for Steps 3 and 4, but they all produce the same output.

##########

#This analysis will be conducted on the intdata excel file.

#This file contains the raw and centered versions of two indicators each for two latent predictors.

#It also contains the four cross products of those centered predictors and a single dependent variable.

#################

#Step 0. Load relevant packages and data.

#First, we install and library lavaan.

install.packages("lavaan")

library(lavaan)

#Next we read in the data file

library(readxl)

intdata <- read\_excel("C:/Users/jcortina/Dropbox/MSEM/New R code/intdata.xlsx")

View(intdata)

################

#Step 1. Create additive model (the first step in the Ping 2 step procedure)

#For the Ping 2 step procedure,

#at this stage we need only the variables from the additive portion of the model.

#This code creates the R object that reflects the additive model,

#which is tested first in the Ping 2 step procedure.

#Measurement model with reference indicators (aff1t1c, ti1t1c) as well as structural model

model.pingadditive <- '

Afft1 =~ 1\*aff1t1c+aff2t1c

Trnt1 =~ 1\*ti1t1c+ti2t1c

NPVHC ~ Afft1 + Trnt1'

##############

#Step 2. Execute additive model and save relevant information

model.pingadditive.fit <- sem(model.pingadditive, data=intdata)

summary(model.pingadditive.fit, fit.measures=TRUE, standardized=TRUE)

#############

#Step 3 Option 1.

#You can input the "estimates" from "summary" output above into the calculator: https://georgemasonio.shinyapps.io/ModSEM/

#The directions below describe the "summary" output that should be used as input into the calculator

#For loadings:

 #Use information from the "Latent Variables" table that comes from the summary output. You will want the "Estimate" column

#For error variances:

 #Use information from the "Variances" table that comes from the summary output. You will want the "Estimate" column

#For latent covariance:

 #Use information from the "Covariances" table that comes from the summary output. You will want the "Estimate" column

#Once you have values from the online calculator, proceed to Step 4, Option 1 in order to input the values

##############

#Step 3 Option 2.

#You can manually input the numbers from the "summary" output above into the following equations. The correct values from "summary" have already been inputted below.

#Use information from the "Latent Variables" table that comes from the summary output. You will want the "Estimate" column

 x1lambda <- 1 #Indicator 1 (e.g. x1) Loading on First Latent Variable

 x2lambda <- .959 #Indicator 2 (e.g. x2) Loading on First Latent Variable

 x3lambda <- 1 #Indicator 3 (e.g. x3) Loading on Second Latent Variable

 x4lambda <- .964 #Indicator 4 (e.g. x4) Loading on Second Latent Variable

#Use information from the "Variances" table that comes from the summary output. You will want the "Estimate" column

 LV1 <- 2.23 #Latent Variable 1 Variance

 LV2 <- 1.69 #Latent Variable 2 Variance

 x1e <- .524 #Indicator 1 (e.g.x1) Error Variance

 x2e <- .552 #Indicator 2 (e.g.x2) Error Variance

 x3e <- .467 #Indicator 3 (e.g. x3) Error Variance

 x4e <- .538 #Indicator 4 (e.g. x4) Error Variance

#Use information from the "Covariances" table that comes from the summary output. You will want the "Estimate" column

 LV1LV2 <- -1.466 #Covariance between latent variables

##############

#Step 3 Option 3.

#Alternatively, use the R code below to automatically save your additive model output and avoid any manual input

save <- lavInspect(model.pingadditive.fit, what = "est")

loadings <- as.data.frame(save$lambda)

variances <- as.data.frame(save$theta)

latentVar <- as.data.frame(save$psi)

LV1 <- latentVar["Afft1","Afft1"] #Latent Variable 1 Variance

LV2 <- latentVar["Trnt1","Trnt1"] #Latent Variable 2 Variance

LV1LV2 <- latentVar["Afft1","Trnt1"] #Covariance between latent variables

x1lambda <- loadings["aff1t1c","Afft1"] #Indicator 1 (e.g. x1) Loading on Latent Variable

x2lambda <- loadings["aff2t1c","Afft1"] #Indicator 2 (e.g. x2) Loading on Latent Variable

x3lambda <- loadings["ti1t1c","Trnt1"] #Indicator 3 (e.g. x3) Loading on Latent Variable

x4lambda <- loadings["ti2t1c","Trnt1"] #Indicator 4 (e.g. x4) Loading on Latent Variable

x1e <- variances["aff1t1c","aff1t1c"] # Indicator 1 (e.g.x1) Error Variance

x2e <- variances["aff2t1c","aff2t1c"] #Indicator 2 (e.g.x2) Error Variance

x3e <- variances["ti1t1c","ti1t1c"] #Indicator 3 (e.g. x3) Error Variance

x4e <- variances["ti2t1c","ti2t1c"] #Indicator 4 (e.g. x4) Error Variance

################

#Step 4. Calculate measurement properties of the latent product

################

#Step 4 Option 1.

#If you used the calculator, replace the 1's below with values from the calculator.

#The comments explain which output from to the calculator to assign to each R object

#Measure Error Variances for Latent product indicators

Meas\_err\_var\_x1x3 <- 1 #Measurement error variance x1x3

Meas\_err\_var\_x1x4 <- 1 #Measurement error variance x1x4

Meas\_err\_var\_x2x3 <- 1 #Measurement error variance x2x3

Meas\_err\_var\_x2x4 <- 1 #Measurement error variance x2x4

#Loadings for Latent product indicators

Loadx1x3 <- 1 #Loading x1x3

Loadx1x4 <- 1 #Loading x1x4

Loadx2x3 <- 1 #Loading x2x3

Loadx2x4 <- 1 #Loading x2x4

#Variance for the Latent product

LatProVar <- 1

###############

#Step 4 Option 2.

#The code below will use the input from Step 3 (using Options 2 or 3) to calculate the properties for the latent product

#Calculate Measure Error Variances for Latent product indicators (Eq. 6 in Cortina et al., 2021)

Meas\_err\_var\_x1x3 <- x1lambda\*x1lambda\*LV1\*x3e+

 x3lambda\*x3lambda\*LV2\*x1e+(x1e\*x3e)

Meas\_err\_var\_x1x4 <- x1lambda\*x1lambda\*LV1\*x4e+

 x4lambda\*x4lambda\*LV2\*x1e+(x1e\*x4e)

Meas\_err\_var\_x2x3 <- x2lambda\*x2lambda\*LV1\*x3e+

 x3lambda\*x3lambda\*LV2\*x2e+(x2e\*x3e)

Meas\_err\_var\_x2x4 <- x2lambda\*x2lambda\*LV1\*x4e+

 x4lambda\*x4lambda\*LV2\*x2e+(x2e\*x4e)

#Calculate Loadings for Latent product indicators (Eq. 8)

Loadx1x3 <- x1lambda\*x3lambda

Loadx1x4 <- x1lambda\*x4lambda

Loadx2x3 <- x2lambda\*x3lambda

Loadx2x4 <- x4lambda\*x2lambda

#Calculate Variance for the Latent product (Eq. 7)

LatProVar <- (LV1 \* LV2) + (LV1LV2 \* LV1LV2)

################

#Step 5. Create the full model

#This code creates the R object that reflects the second step in the Ping 2 step procedure.

#Measurement model with reference indicators (aff1t1c, ti1t1c, and a1t1). The paste function is required because of the combination of model variables and computed objects in several of the lines

model.interping <- paste('

Afft1 =~ 1\*aff1t1c + aff2t1c

 Trnt1 =~ 1\*ti1t1c + ti2t1c

 Inter =~', Loadx1x3,'\*a1t1 + ',Loadx1x4,'\*a1t2 + ',Loadx2x3,'\*a2t1 + ',Loadx2x4,'\*a2t2

 a1t1~~',Meas\_err\_var\_x1x3,'\*a1t1

 a1t2~~',Meas\_err\_var\_x1x4,'\*a1t2

 a2t1~~',Meas\_err\_var\_x2x3,'\*a2t1

 a2t2~~',Meas\_err\_var\_x2x4,'\*a2t2

 Inter~~', LatProVar,'\*Inter #fixes variance of the latent product

 aff1t1c~~a1t1 #Freeing error covariances for terms that share components as implied by kenny and judd

 aff1t1c~~a1t2

 aff2t1c~~a2t1

 aff2t1c~~a2t2

 ti1t1c~~a1t1

 ti1t1c~~a2t1

 ti2t1c~~a1t2

 ti2t1c~~a2t2

 a1t1~~a1t2

 a1t1~~a2t1

 a2t1~~a2t2

 a1t2~~a2t2

 NPVHC ~ Afft1 + Trnt1 + Inter')

################

#Step 6. Fit the full model

model.interping.fit <- sem(model.interping, data=intdata)

summary(model.interping.fit, fit.measures=TRUE, standardized=TRUE)

#Because sets of variables that contain products are not multivariate normal,

#distribution free estimators can be useful.

#The simplest is probably the Diagonally Weighted Least Squares (DWLS) estimator,

#which is based on a fit function in which values are weighted by their sampling variances,

#i.e., the diagonal of the sampling variance/covariance matrix.

#Instead, use the following code to fit the model

model.interping.fit <- sem(model.interping, data=intdata, estimator="WLSMV")

summary(model.interping.fit, fit.measures=TRUE, standardized=TRUE)