SEM

latent variable definition =~ (is measured by)

regression ~ (is regressed on)

(residual) (co)variance ~~ (is correlated with)

intercept ~1 (intercept)

new model parameter := (is equal to)

**CFA for 1-factor:**

* Step 1: subset the data into a data frame with only the indicators for the CFA measurement model and ask for the observed variance-covariance matrix:

NameOfData\_CFAIndicators<-select(NameOfData, Var1, Var2,…)

NameOfData\_CFAIndicators\_ObCov<-cov(NameOfData\_CFAIndicators, use = “pairwise.complete.obs”)

Example:

Welfare\_Support<-select(WelfareOpinion,gvslvol,gvslvue,gvhlthc,gvcldcr,gvjbevn,gvpdlwk)

Welfare\_Support\_ObCov<-cov(Welfare\_Support,use = "pairwise.complete.obs")

* Step 2: we can also ask for a correlation matrix:

NameOfData\_CFAindicatiors\_ObCor<-cov2cor(NameOfData\_CFAIndicators\_ObCov)

Example:

Welfare\_Support\_ObCor<-cov2cor(Welfare\_Support\_ObCov)

* Step 3: plot the correlation or covariance matrix:

corrplot::corrplot(NameOfData\_CFAindicatiors\_ObCor/ NameOfData\_CFAIndicators\_ObCov,

is.corr=FALSE,

method = “ShapeOfDiagonal – square/circle”,

type = “PositionOfMatrix – lower/upper”,

addCoef.col = “ColorOfFontForCor&Cov”)

Example: for a covariance matrix that we want to show the left bottom side of the matrix with the diagonal value highlighted through square and the values are all in blue:

corrplot::corrplot(Welfare\_Support\_ObCov,

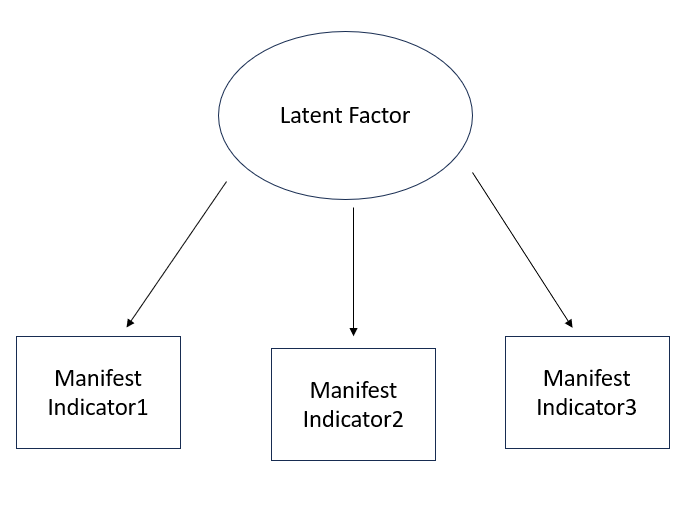
is.corr = FALSE,

method = "square",

type = "lower",

addCoef.col = "blue")

* Step 4: draw the measurement model with latent variable and indicators like the below, and run CFA for the default of the first indicator factor loading constraining at 1:



model\_NameOfLatentFactor\_xInd<-‘LatentFactor=~NameOfIndicator1+ NameOfIndicator2+ NameOfIndicator3…’

modelFit\_NameOfLatentFactor\_xInd<-cfa(model = model\_NameOfLatentFactor\_xInd, data = NameOfData\_CFAIndicators)

summary(modelFit\_NameOfLatentFactor\_xInd, standardized = TRUE)

example:

model\_ws\_3Ind<-'welfareSupport=~gvslvol+ gvslvue+gvhlthc'

cfa\_fit\_3Ind<-cfa(model = model\_ws\_3Ind, data=Welfare\_Support)

summary(cfa\_fit\_3Ind,standardized=TRUE)

🡪the output ‘Variances’ are residual variances

🡪If we want to change the marker indicator, we put it as the first in the model definition, like:

model\_ws\_3Ind<-'welfareSupport=~gvslvue+ gvslvol+gvhlthc'

* (Step 5:) To constrain the latent variable variance to 1:

model\_alt<-‘LatentFactor=~NA\* NameOfIndicator1+ NameOfIndicator2+ NameOfIndicator3…

LatentFactor~~1\*LatentFactor’

cfa\_model\_alt<-cfa(model = model\_alt, data = NameOfData\_CFAIndicators/DataIncludingIndicators)

summary(cfa\_model\_alt, standardized = TRUE)

!NA means not to constrain the factor loading of the indicator to 1

* Step 6: To start the model fitting procedure, we need to first compare the observed and implied variance-covariance matrix:

Sample Observed Variance-Covariance Matrix: lavInspect(NameOfCFAfit, “sampstat”)

Implied Variance-Covariance Matrix: fitted(NameOfCFAfit)

Difference between sample and implied Variance-Covariance Matrix: residuals(NameOfCFAfit)

Example:

lavInspect(model\_fit\_ws\_6Ind,"sampstat")

fitted(model\_fit\_ws\_6Ind)

residuals(model\_fit\_ws\_6Ind)

* (Step 7: ) To ask for the standard error for the factor loading:

lavTech(NameOfCFAfit, “se”)

Example:

lavTech(model\_fit\_ws\_6Ind,"se")

* Step 8: to ask for the confidence interval of the factor loading and the confidence interval of each indicators’ common variances:

parameterEstimates(NameOfCFAfit, standardized=TRUE)

Example:

parameterEstimates(model\_fit\_ws\_6Ind,standardized=TRUE)

* **!!!STEP 9!!!**: to see the model fit with statistics for Chi-square, RMSEA, CFI, TFI, SRMR etc, we use:

summary(NameOfCFAFit, fit.measures=TRUE, standardized=TRUE)

Example:

summary(model\_fit\_ws\_6Ind,fit.measures=TRUE, standardized=TRUE)

OR

To select only the fit measures we want:

fitMeasures(NameOfCFAfit, c(“AIC”,"chisq", "df", "pvalue", "cfi", "tli","rmsea"), output = "matrix")

Example:

fitMeasures(fit\_ws\_6\_1, c("chisq", "df", "pvalue", "cfi", "tli","rmsea"), output = "matrix")

* (Step 10: )To extract information instead reading all the information, such as standardized factor loading:

inspect(NameOfCFAfit, what = “std”)$lambda

Example:

inspect(model\_fit\_ws\_6Ind,what = "std")$lambda

Alternatively, we can also ask a tidier version of the confidence interval of the standardized factor loading, significant level etc)

tidy\_results<-table\_results(NameOfCFAfit, columns = c(“label”,”est\_sig”,”se”,”confint”), digits = decimals)

Example:

tidy\_results<-table\_results(model\_fit\_ws\_6Ind,

columns = c("label","est\_sig","se","confint"),

digits=3)

To see only the standardized factor loading:

tidy\_results<-filter(tidy\_results, str\_detect(label,”LatentVariableName”))

Example:

tidy\_results<-filter(tidy\_results,str\_detect(label,"welfareSupport"))

* Step 11: Residual Variance:
* To ask for the standardized residual (unique) variances of each indicator:

uniqueVar<-round(inspect(NameOfCFAfit, “std”)$theta, digits = decimals)

Example: uniqueVar<-round(inspect(model\_fit\_ws\_6Ind,"std")$theta,digits = 3)

🡪 the output is a matrix and the unique variance of each indicator is on the diagonal

* To ask for the regular residual (unique) variances before standardization:

RegVar<-round(inspect(NameOfCFAfit,”est”)$theta, digits = decimals)

Example: RegUniqueVar<-round(inspect(model\_fit\_ws\_6Ind,"est")$theta,digits = 3)

* To ask for the R-square????

R2<-round(inspect(NameOfCFAfit, “r2”), digits = decimals)

Example: R2<-round(inspect(model\_fit\_ws\_6Ind,"r2"),digits = 3)

* To make all the information nicer in a data frame:

ResidualVariance<-data.frame(UniqueVariance=diag(uniqueVar),

RegularResidualVariance=diag(RegVar),

RSqaure=R2)

Example:

residualVar<-data.frame(RegularResiduals=diag(RegUniqueVar),

StandUniResiduals=diag(uniqueVar),

R\_square=R2)

* Modification Indices:

mi<-inspect(NameOfCFAfit, “mi”)

mi.sorted<-mi[order(-mi$mi),]

head(mi.sorted)

🡪 output: spec.all is the correlation of the residual variances of the two indicators

* For the chi-square difference test between the original model and the modified model, we use:

anova(NameOfCFAfit1, NameOfCFAfit2)

**CFA for more than 1 factor:**

* Apart from the basic steps as above, we need to define the model differently:

NameOfMeasurementModel<-‘

LatentVar1=~Ind1+Ind2+Ind3

LatentVar2=~Ind4+Ind5+Ind6

LatentVar3=~Ind7+Ind8+Ind9

‘

Example:

model\_3factor<-'

wc\_econo=~sbstrec+sbbsntx

wc\_socio=~sbprvpv+sbeqsoc+sbcwkfm

wc\_moral=~sblazy+sblwcoa+sblwlka

'

* To check the factor loading is similar:

NameOfCFAfit<-cfa(NameOfMeasurementModel, data=NameOfDataWithOnlyIndicatorsForThisVariable)

summary(NameOfCFAfit)

Example:

fit\_3factor<-cfa(model\_3factor,data=welfare\_criticism)

summary(fit\_3factor,standardized=TRUE)

**Plotting**

* First, define the layout:

Example for a three factors with first factor (2 indicators), second factor (3 indicators) and third factor(3 indicators)

layout1<-get\_layout(“factor1”,””,””,”factor2”,””,””,”factor3”,””,

“Ind1”,”Ind2”,”Ind3”,”Ind4”,” Ind5”,” Ind6”,” Ind7”,” Ind8”,rows=2)

layout1 🡪 to check if the layout is good

* Then, build the SEM plot

NameOfPlot<-graph\_sem(model=NameOfCFAfit, layout=layout1,angle=170)

NameOfPlot

Example:

layout1<-get\_layout("wc\_econo","","","wc\_socio","","","wc\_moral","",

"sbstrec","sbbsntx","sbprvpv","sbeqsoc","sbcwkfm","sblazy","sblwcoa","sblwlka",

rows=2)

layout1

plot\_wc<-graph\_sem(model=fit\_3factor, layout=layout, angle=200)

plot\_wc

**MIMIC**

* Before fitting models, need to make dummies for categorical variables!!
* Step 1: in the model, we need to define both the measurement model and the structural model with the relationship between the independent variables and dependent variables:

NameOfModel<-‘

LatentVar=~Ind1+Ind2+Ind3

LatentVar~IndVar1+IndVar2

’

* Step 2: fit the model and run it out

FitModelName<-cfa(model=NameOfModelAbove, data=NameOfDataWithAllVar)

summary(FitModelName, standardized=TRUE)

OR

summary(FitModelName, fit.measures=TRUE, standardized=TRUE)

Example:

model\_WelSup\_mimic<-'

WelfareSup=~gvslvol+gvslvue+gvhlthc

WelfareSup~gndr+eduyrs

'

fit\_WelSup\_mimic<-cfa(model = model\_WelSup\_mimic,data = ESS4\_BE\_lab2)

summary(fit\_WelSup\_mimic,standardized=TRUE)

summary(fit\_WelSup\_mimic,fit.measures=TRUE,standardized=TRUE)

* Interpretation of regression in the structural model:

- for metric variables, we look at std.all

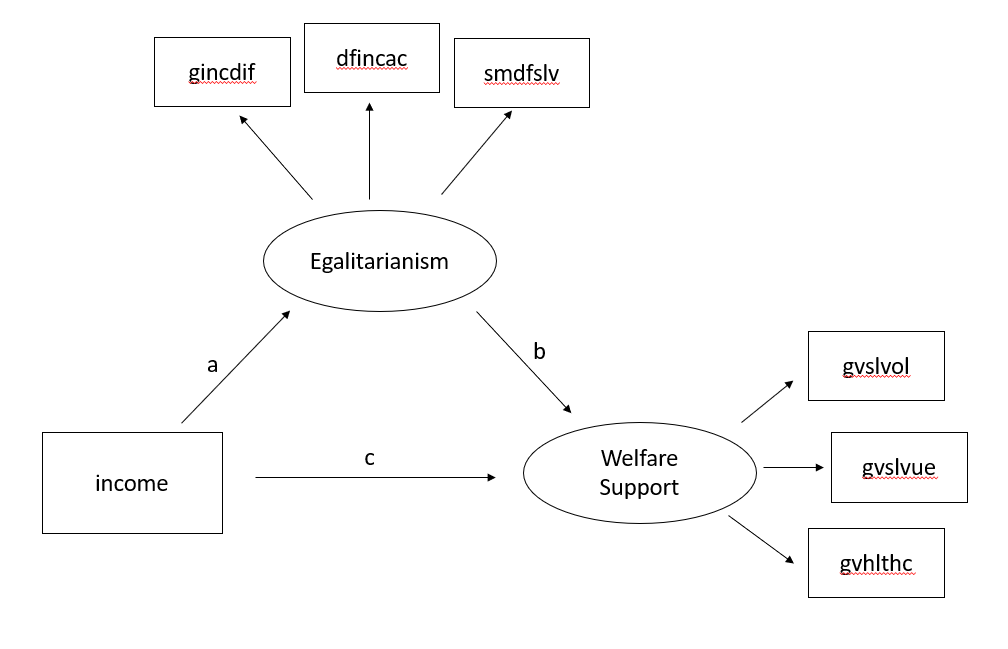
- for categorical variables (dummies), we look at std.lv

- The model fit statistics are no longer informative once we have regression/a complete structural model 🡪 model fit is mostly for measurement model

**Mediation Model**

**🡪 causality between mediator and independent variables are not established!**

* Step 1: we need to draw the diagram out for the measurement model, structural model and also NAME the path



* Step 2: we need to put the measurement model, structural model and the path inside the model:

NameOfMediationModel<-‘

LatentVar1=~Ind1+Ind2+Ind3

LatentVar2=~Ind4+Ind5+Ind6

LatentVar2~a\*LatentVar1

LatentVar1~b\*IndVar1

LatentVar2~c\*IndVar1

Indirect:=a\*b

Total:=a\*b+c

’

* Step 3: we put it to estimate

FitMediationModelName<-cfa(model=NameOfMediationModelAbove, data=DataWithAllVar)

summary(FitMediationModelName, standardized=TRUE)

Example:

model\_welsup\_mediation<-'

WelfareSup=~gvslvol+gvslvue+gvhlthc

Egalitar=~gincdif+dfincac+smdfslv

Egalitar~a\*hinctnta

WelfareSup~b\*Egalitar

WelfareSup~c\*hinctnta

indirect:=a\*b

total:=c+a\*b

'

fit\_welsup\_mediation<-cfa(model = model\_welsup\_mediation,data=ESS4\_BE\_lab2)

summary(fit\_welsup\_mediation,standardized=TRUE)

* Interpretation:

- we can look at the end of the output to see if indirect effect and total effect is significant

- indirect effect: ab=-0.051, income increases 1 sd, welfare support will decrease 0.051 sd through egalitarianism

**MGCFA**

* Configural invariance:

1. Step 1: Read the data in, the categorical variable that will separate the group (like country, gender) needs to be coerced into factor:

Example:

ESS4\_BE\_clean$gndr<-factor(ESS4\_BE\_clean$gndr,

levels=c("1","2"),

labels = c("male","female"))

1. Step 2: define the model for the factor in measurement model as before:

NameOfModel<-‘NameOfFactor=~Ind1+Ind2+Ind3’

Example:

model\_ws\_byGender<-'welfareSup=~gvslvol+gvslvue+gvhlthc'

1. Step 3: fit the model, but this time, add in the group argument (Don’t forget the quotation mark surrounding the categorical variable that separate the group)

NameOfCFAfit\_Configure<-cfa(model=NameOfModel, data = NameOfData, **group = “CatVarName”**)

Example:

fit\_wsConfi\_byGender<-cfa(model = model\_ws\_byGender,

data = ESS4\_BE\_clean,

group = "gndr")

1. Step 4: view the results:

summary(NameOfCFAfit\_configure, standardized = TRUE)

Example:

summary(fit\_wsConfi\_byGender,standardized=TRUE)

* Metric invariance:

1. After the above steps, we keep the model definition as it is, and we need to constrain the factor loadings of the same items to be the same across all groups.

Therefore, we change the step of the cfa fit as below:  
NameOfCFAfit\_metric<-cfa(model=NameOfModel, data=NameOfData,

group = “CatVarName”,

group.equal=c(“loadings”))

Example:

fit\_wsMetric\_byGender<-cfa(model=model\_ws\_byGender,

data = ESS4\_BE\_clean,

group = "gndr",

group.equal=c("loadings"))

summary(fit\_wsMetric\_byGender,standardized=TRUE)

* Scalar Invariance:

After all the above steps to have metric invariance, we will keep the model definition as it is and keep the factor loading constrain as it is, and we will add the constrain to the intercepts to make it the same across all groups for the same indicators:

NameOfCFAfit\_scalar<-cfa(model=NameOfModel, data=NameOfData,

group=”CatVarName”,

group.equal = c(“loadings”,”intercepts”))

Example:

fit\_wsScalar\_byGender<-cfa(model = model\_ws\_byGender, data=ESS4\_BE\_clean,

group = "gndr",

group.equal=c("loadings","intercepts"))

summary(fit\_wsScalar\_byGender,standardized=TRUE)

* Strict Invariance (with all the residual variances constrain to the same):

After all the above steps to have scalar invariance, we will keep the model definition as it is, keep the factor loadings and intercepts constrain as it is, and we will add the constrain to residual variance to make it the same for the same indicator across groups:  
NameOfCFAfit\_strict<- cfa(model=NameOfModel, data=NameOfData,

group=”CatVarName”,

group.equal = c(“loadings”,”intercepts”,”residuals”))

Example:

fit\_wsStrict\_byGender<-cfa(model = model\_ws\_byGender,data=ESS4\_BE\_clean,

group = "gndr",

group.equal=c("loadings","intercepts","residuals"))

summary(fit\_wsStrict\_byGender,standardized=TRUE)

* Structural Invariance (with factor variances and covariances the same to see if all the groups are the same for this measurement – heterogeneity)

NameOfCFAfit\_struture<- cfa(model=NameOfModel, data=NameOfData,

group=”CatVarName”,

group.equal = c(“loadings”,”intercepts”,”residuals”,”lv.variances”,”lv.covariances”))

Example:

fit\_WSStructure\_byGender<-cfa(model = model\_ws\_byGender,data=ESS4\_BE\_clean,

group = "gndr",

group.equal=c("loadings","intercepts","residuals",

"lv.variances","lv.covariances"))

summary(fit\_WSStructure\_byGender,standardized=TRUE)

* To get all the global fit statistics together:

Use self-created function 🡪 Meijun GitHub <https://github.com/Meijuny/Structural_Equation/blob/main/Function%20for%20MGCFA%20Fit.R> :

Example:

fitStatCompare(ConfiguralFit = fit\_wsConfi\_byGender,MetricFit = fit\_wsMetric\_byGender,

ScalarFit = fit\_wsScalar\_byGender,StrictFit = fit\_wsStrict\_byGender,

StructuralFit = fit\_WSStructure\_byGender)

* To get the chi-square difference test results together:

Use self-created function 🡪 Meijun Github

<https://github.com/Meijuny/Structural_Equation/blob/main/Function%20for%20MGCFA%20Fit.R> :

Example:

ChiSqDifferenceTest\_results(ConfiguralFit = fit\_wsConfi\_byGender,

MetricFit = fit\_wsMetric\_byGender,

ScalarFit = fit\_wsScalar\_byGender,

StrictFit = fit\_wsStrict\_byGender,

StructuralFit = fit\_WSStructure\_byGender)

!!The same results can be obtained from **lavTestLRT()**:

lavTestLRT(fit\_wsConfi\_byGender, fit\_wsMetric\_byGender, fit\_wsScalar\_byGender, fit\_wsStrict\_byGender)

* Based on the ANOVA test, we figure out which invariance is significantly worse, and get the MI for it:

1. Step 1: we first ask for the coded MI:

lavTestScore(fitOfCFAInvariance)

Example: if the chi-square difference test shows significantly worsening at scalar invariance, we do:

lavTestScore(fit\_wsScalar\_byGender)

1. Step 2: based on the coded MI, we find the biggest chi-square difference, and go to see what are those codes are for:

Example: if it shows p5 == p16, then we need to use parTable to see what p5 and p16 corresponding to:

Example:

parTable(fit\_wsScalar\_byGender)

Find the line corresponding to p5 == p16 to find the variables

1. Step 3: free the variable parameters

Example: if we find out p5 == p16 is gvslvue ~~ gvslvue

Then we need to modify the scalar cfa fit as below:

fit\_wsStrict\_byGender\_free<-cfa(model = model\_ws\_byGender,data=ESS4\_BE\_clean,

group = "gndr",

group.equal=c("loadings","intercepts","residuals"),

**group.partial=c(gvslvue ~~ gvslvue)**)

Output reading!!

* To set free intercept: IndicatorName~1 (Example: gvslvol ~1)
* To set free loadings: LatentVarName=~IndicatorName (Example: welfareSup =~ gvslvue)
* To set the residual variance for one specific indicator free or covariance between several indicators free, we do: IndicatorName~~IndicatorName OR IndicatorName1~~IndicatorName2

(Example: gvslvol ~~ gvslvol)

1. Step 4: ask for the anova chi-square difference test for all the models again:

Example:

ChiSqDifferenceTest\_results(ConfiguralFit = fit\_wsConfi\_byGender,

MetricFit = fit\_wsMetric\_byGender,

ScalarFit = fit\_wsScalar\_byGender,

**StrictFit = fit\_wsStrict\_byGender\_free**,

StructuralFit = fit\_WSStructure\_byGender)

* To set certain parameters in certain groups free, we need to manually specify the model:
* For full scalar equivalence:

ModelName<-‘

LatentVar=~c(L1,L1,L1,...)\*Ind1+c(L2,L2,L2,...)\*Ind2+c(L3,L3,L3,...)\*Ind3

Ind1~c(I1,I1,I1,…)\*1

Ind2~c(I2,I2,I2,…)\*1

Ind3~c(I3,I3,I3,…)\*1

LatentVar~c(0,NA,NA,…)\*0

’

Example:

model\_wsScalar<-'

welfareSup=~c(L1,L1,L1)\*gvslvol+c(L2,L2,L2)\*gvslvue+c(L3,L3,L3)\*gvhlthc

gvslvol~c(I1,I1,I1)\*1

gvslvue~c(I2,I2,I2)\*1

gvhlthc~c(I3,I3,I3)\*1

gvslvol~~gvslvol

gvslvue~~gvslvue

gvhlthc~~gvhlthc

welfareSup~c(0,NA)\*0

'

* To set the factor loading of the second indicator in the third group free:

ModelName<-‘

LatentVar=~c(L1,L1,L1,...)\*Ind1+c(L2,L2,**NA**,...)\*Ind2+c(L3,L3,L3,...)\*Ind3

Ind1~c(I1,I1,I1,…)\*1

Ind2~c(I2,I2,I2,…)\*1

Ind3~c(I3,I3,I3,…)\*1

LatentVar~c(0,NA,NA,…)\*0

’

Example:

model\_wsScalar<-'

welfareSup=~c(L1,L1,L1)\*gvslvol+c(L2,L2,NA)\*gvslvue+c(L3,L3,L3)\*gvhlthc

gvslvol~c(I1,I1,I1)\*1

gvslvue~c(I2,I2,I2)\*1

gvhlthc~c(I3,I3,I3)\*1

gvslvol~~gvslvol

gvslvue~~gvslvue

gvhlthc~~gvhlthc

welfareSup~c(0,NA)\*0

'

* To set the intercept of the second indicator in the second group free:

ModelName<-‘

LatentVar=~c(L1,L1,L1,...)\*Ind1+c(L2,L2,L2,...)\*Ind2+c(L3,L3,L3,...)\*Ind3

Ind1~c(I1,I1,I1,…)\*1

Ind2~c(I2,**NA**,I2,…)\*1

Ind3~c(I3,I3,I3,…)\*1

LatentVar~c(0,NA,NA,…)\*0

’

Example:

model\_wsScalar<-'

welfareSup=~c(L1,L1,L1)\*gvslvol+c(L2,L2,NA)\*gvslvue+c(L3,L3,L3)\*gvhlthc

gvslvol~c(I1,I1,I1)\*1

gvslvue~c(I2,NA,I2)\*1

gvhlthc~c(I3,I3,I3)\*1

gvslvol~~gvslvol

gvslvue~~gvslvue

gvhlthc~~gvhlthc

welfareSup~c(0,NA)\*0

'

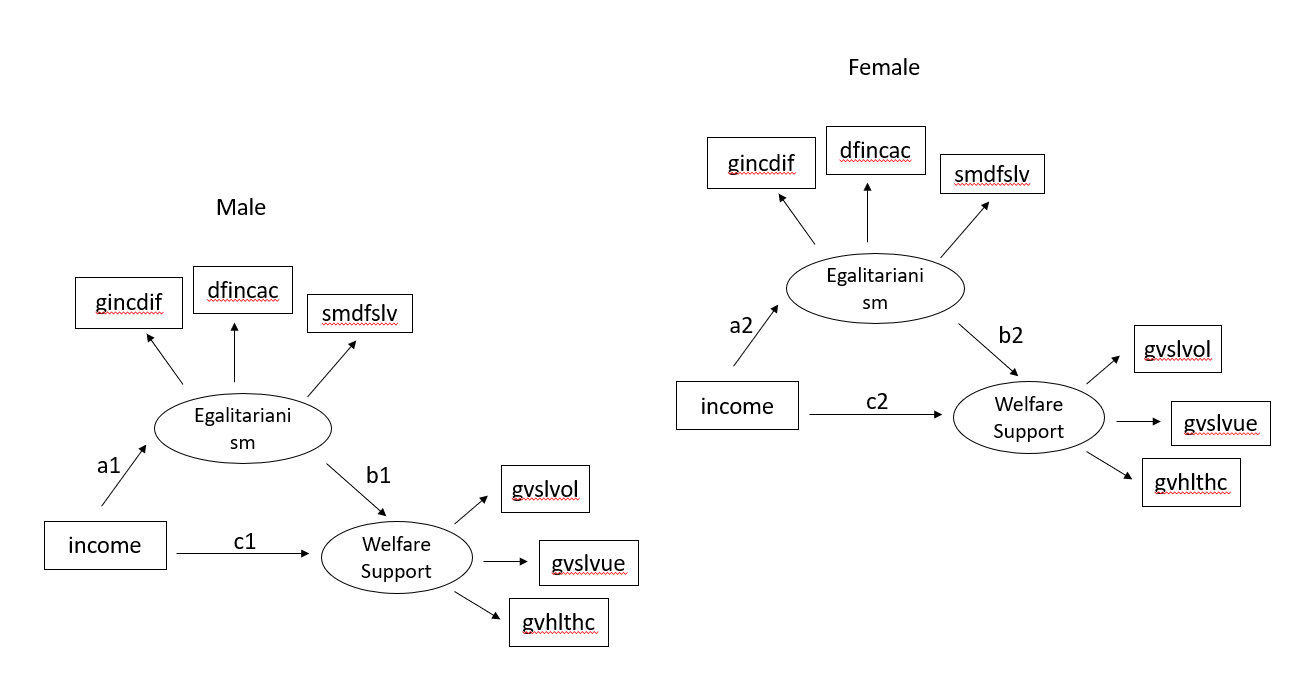
**MGSEM**

* If we want to compare the regression coefficient in MGSEM, then we only need to constrain the factor loadings to be equal;

If we want to compare the regression coefficient AND the latent mean across group, then we need to also constrain the intercepts to be equal

* Step 1: Draw the SEM model for groups so that we can specify the path later:

Example:



* Define the model (the factor of the group has already been coerced in the MGCFA step) based on the diagram:

NameOfSEMmodel<-‘

LatentVar1=~Ind1+Ind2+Ind3

LatentVar2=~Ind4+Ind5+Ind6

MediatorORLatentVar2~c(a1,a2)\*ManifestVar1

DepVarORLatentVar1~c(b1,b2,...)\*LatentVar2+c(c1,c2,...)\*ManifestVar1+c(d1,d2,...)\*ManifestVar2

IndirectEffect\_ManifestVar1\_group1:=a1\*b1

TotalEffect\_ManifestVar1\_group1:a1\*b1+c1

IndirectEffect\_ManifestVar1\_group2:=a2\*b2

TotalEffect\_ManifestVar1\_group2:a2\*b2+c2

’

Example:

model\_ws\_onEgaliIncome<-'

WelSupp=~gvslvol+gvslvue+gvhlthc

Egali=~gincdif+dfincac+smdfslv

Egali~c(a1,a2)\*hinctnta

WelSupp~c(b1,b2)\*Egali+c(c1,c2)\*hinctnta

IncomeIndirect\_Male:=a1\*b1

IncomeIndirect\_Female:=a2\*b2

IncomeTotal\_Male:=a1\*b1+c1

IncomeTotal\_Female:=a2\*b2+c2

'

* Do the CFA function and constrain the factor loadings:

NameOfSEMFit<-cfa(model = NameOfSEMmodel, data = NameOFData, group = “CatVarForGroupings”, **group.equal = c(“loadings”)**)

Example:

fit\_ws\_onEgaliIncome<-cfa(model = model\_ws\_onEgaliIncome,

data = ESS4\_BE\_clean,

group = "gndr",

**group.equal=c("loadings")**)

summary(fit\_ws\_onEgaliIncome, standardized=TRUE)

* For the next step, to make the coefficient the same, we need to constrain it in the model definition

NameOfSEMconstrainModel<-‘

LatentVar1=~Ind1+Ind2+Ind3

LatentVar2=~Ind4+Ind5+Ind6

MediatorORLatentVar2~c(a1,**a1**)\*ManifestVar1

DepVarORLatentVar1~c(b1,**b1**,...)\*LatentVar2+c(c1,**c1**,...)\*ManifestVar1+c(d1,**d1**,...)\*ManifestVar2

IndirectEffect\_ManifestVar1:=a1\*b1

TotalEffect\_ManifestVar1:a1\*b1+c1

’

Example:

model\_constrain\_wsOnIncomeEgali<-'

welSupp=~gvslvol+gvslvue+gvhlthc

Egali=~gincdif+dfincac+smdfslv

Egali~c(a1,a1)\*hinctnta

welSupp~c(b1,b1)\*Egali+c(c1,c1)\*hinctnta

IndirectIncome:=a1\*b1

TotalIncome:=a1\*b1+c1

'

* See the new model fit:

NameOfConstrainFit<-cfa(model=NameOfSEMconstrainModel, data = NameOfData, group = “CatVarName”, group.equal=c(“loadings”))

Example:

fit\_constrain\_wsOnIncomeEgali<-cfa(model = model\_constrain\_wsOnIncomeEgali,

data = ESS4\_BE\_clean,

group = "gndr",

group.equal=c("loadings"))

summary(fit\_constrain\_wsOnIncomeEgali,standardized=TRUE)

* Compare the fit between the non-constrain and constrain model

anova(NameOfSEMfit, NameOfSEMConstrainFit)

Example:

anova(fit\_ws\_onEgaliIncome,fit\_constrain\_wsOnIncomeEgali)

**Visualization/detection of Normality:**

* To ask for histogram to visualize the normality of a certain variable:

NameOfHistogram<-ggplot(NameOfData, aes(VarName)) +geom\_histogram(aes(y=after\_stat(density)), binwidth = 1, colour = “black”, alpha=0.3)

Example:

hist\_gvslvol<-ggplot(ESS4\_BE,aes(gvslvol)) + geom\_blank()+ geom\_histogram(aes(y=after\_stat(density)),

binwidth = 1,

colour="black",

alpha=0.3)

* THIS NEEDS PACKAGE: patchwork

To print the histograms for several variables at the same time to compare:

NameOfHistogram1 + NameOfHistogram2 + NameOfHistogram3 + …

Example:

hist\_gvslvol + hist\_gvhlthc + hist\_gvcldcr + hist\_gvpdlwk

🡪The output is 4 histograms in a 2X2 layout

* To perform Kolmogorov-Smirnov (KS) test to be more sure of the normality:

NameOfVarNameKSTest<-ks.test(x=NameOfData$VarName, y=”pnorm”, mean(NameOfData$VarName, na.rm=TRUE), sd(NameOfData$VarName, na.rm=TRUE))

Example:

ks\_gvslvol<-ks.test(x=ESS4\_BE$gvslvol,y="pnorm",mean(ESS4\_BE$gvslvol, na.rm=TRUE),

sd(ESS4\_BE$gvslvol,na.rm = TRUE))

* To bind variable KS test results together for comparison

KSCompare<-data.frame(VarName = c(“Var1”,”Var2”,”Var3”,…),

ksStat=round(c(ks\_Var1TestName$statistics, ks\_Var2TestName$statistics, ks\_Var3TestName$statistics, ...), digits =2),

pValue= c(ks\_Var1TestName$p.value, ks\_Var2TestName$p.value, ks\_Var3TestName$p.value, ...))

Example:

ks\_statCom2<-data.frame(VarName=c("gvslvol","gvhlthc","gvcldcr","gvpdlwk"),

ksStat=c(round(ks\_gvslvol$statistic,digits = 2),

round(ks\_gvhlthc$statistic,digits = 2),

round(ks\_gvcldcr$statistic,digits = 2),

round(ks\_gvpdlwk$statistic, digits = 2)),

pValue=c(ks\_gvslvol$p.value,ks\_gvhlthc$p.value,ks\_gvcldcr$p.value,ks\_gvpdlwk$p.value))

* To perform normality test for multivariate 🡪 we use Henze-Zirkler’s multivariate normality test. 🡪 For multivariate normality, the test p-value should be greater than 0.05

First we select the variables for this test out (they will be the same variables we use for the cfa):

NameOfData\_MVNselected<-select(NameOfData, Var1, Var2, Var3,…)

Then we will take out the NA

nameOfData\_MVNselected\_complete<-na.omit(NameOfData\_MVNselected)

We will put the data without NA into the test:

HZ\_testName<-mvn(data = nameOfData\_MVNselected\_complete, mvnTest=”hz”)

To check the statistics:

HZ\_testName$multivariateNormality

Example:

ESS4\_BE\_4Var<-select(ESS4\_BE,gvslvol, gvhlthc, gvcldcr, gvpdlwk)

ESS4\_BE\_4Var\_complete<-na.omit(ESS4\_BE\_4Var)

HZtest\_4var<-mvn(data = ESS4\_BE\_4Var\_complete, mvnTest = "hz")

HZtest\_4var$multivariateNormality

* To correct the normality and gives weight in the CFA model:

modelName<-‘LatentVar=~Var1+Var2+Var3+...’

NameOfCFAfit<-cfa(model=modelName, data=NameOfData, **estimator = “MLM”**)

summary(NameOfCFAfit, fit.measures=TRUE, standardized=TRUE)

!MLM is good if the missing values are not so much, but it is better to use MLR if there are a lot of missing values!

Example:

model\_welfareSup<-'welfareSup=~gvslvol+gvhlthc+gvcldcr+gvpdlwk'

fit\_welfareSup\_MLM<-cfa(model = model\_welfareSup,

data = ESS4\_BE,

estimator="MLM")

summary(fit\_welfareSup\_MLM,fit.measures=TRUE,standardized=TRUE)

**Categorical Variables using Threshold**

* Define the model as it is:

NameOfModel<-‘

LatentVar1=~Ind1+Ind2+Ind3

LatentVar2=~Ind4+Ind5+Ind6

‘

* Use argument ordered to cut threshold and transforming the ordinal variable to a normal distribution and argument estimator to change from ML to WLS:

NameOfCFAfit<-cfa(model = NameOfModel, data=NameOfData, ordered= c(“OrdVar1”,”OrdVar2”,”OrdVar3”,”OrdVar4”,…), estimator = “WLSMV”)

!!The order will directly break a numeric factor into categorical variable, even with the binary variable (0=male, 1=female)!!

Example:

model\_social\_moral<-'

SocioCri=~sbprvpv+sbeqsoc+sbcwkfm

MoralCri=~sblazy+sblwcoa+sblwlka

'

fit\_social\_moral<-cfa(model = model\_social\_moral,

data = ESS4\_BE,

ordered = c("sbprvpv", "sbeqsoc", "sbcwkfm","sblazy","sblwcoa","sblwlka"),

estimator="WLSMV")

summary(fit\_social\_moral,standardized=TRUE)

**Categorical Variable Threshold Invariance**

* To check invariance:
* Define the model as it is:

NameOfModel<-‘

LatentVar1=~Ind1+Ind2+Ind3

LatentVar2=~Ind4+Ind5+Ind6

‘

Example: model\_eg\_sc<-‘egali=~gincdif+dfincac+smdfslv’

* Fit model with metric invariance:

NameOffit\_metric\_cat<-cfa(model=NameOfModel, data= NameOfData, estimator=”WLSMV”, group=”CatVarNameForGrouping”,

ordered=c(“Ind1”,”Ind2”,”Ind3”,…), **group.equal=c(“loadings”)**)

Example:

fit\_eg\_sc\_metric <- cfa(model\_eg\_sc, data = ess\_df, estimator = "WLSMV”,

group = "gndr",

ordered = c("gincdif","dfincac", "smdfslv"), group.equal=c("loadings"))

* Fit model with scalar invariance:

NameOffit\_metric\_cat<-cfa(model=NameOfModel, data= NameOfData, estimator=”WLSMV”, group=”CatVarNameForGrouping”,

ordered=c(“Ind1”,”Ind2”,”Ind3”,…), **group.equal=c(“loadings”,”threshold”)**)

Example:

fit\_eg\_sc\_scalar <- cfa(model\_eg\_sc, data = ess\_df, estimator = "WLSMV”,

group = "gndr",

ordered = c("gincdif","dfincac", "smdfslv"), group.equal=c("loadings",”threshold”))

* After checking the fit statistics, if the scalar invariance cannot be reached, we use lavTestScore and parTable to check whether there are any parameters we can free:

Example:

lavTestScore(fit\_eg\_sc\_scalar)

parTable(fit\_eg\_sc\_scalar)

* If we need to set certain intercepts of a certain indicator free, then we shall first constrain the scale of this item to 1 in order for the model to be identified:

Example :

model\_eg\_sc\_p <- '

egual =~ gincdif + dfincac + smdfslv

dfincac ~\*~ c(1,1)\*dfincac’

* Then we can use group.partial argument to set the intercept free:

NameOfFit\_freePara<- cfa(model=NameOfModel, data= NameOfData, estimator=”WLSMV”, group=”CatVarNameForGrouping”,

ordered=c(“Ind1”,”Ind2”,”Ind3”,…), **group.equal=c(“loadings”,”threshold”)**,

**group.partial=c(“LatentVar=~Ind2”, “Ind2|t1”)**)

* To manually define the model to constrain certain intercepts or loadings to equal:

Example for two groups:

model\_egaliwelfare\_scalar<-'

egali=~c(L1,L1)\*gincdif+c(L2,L2)\*dfincac+c(L3,L3)\*smdfslv

gincdif|c(T1\_1,T1\_1)\*t1

gincdif|c(T1\_2,T1\_2)\*t2

gincdif|c(T1\_3,T1\_3)\*t3

gincdif|c(T1\_4,T1\_4)\*t4

dfincac|c(T2\_1,T2\_1)\*t1

dfincac|c(T2\_2,T2\_2)\*t2

dfincac|c(T2\_3,T2\_3)\*t3

dfincac|c(T2\_4,T2\_4)\*t4

smdfslv|c(T3\_1,T3\_1)\*t1

smdfslv|c(T3\_2,T3\_2)\*t2

smdfslv|c(T3\_3,T3\_3)\*t3

smdfslv|c(T3\_4,T3\_4)\*t4

'

If we do not specify it, then it will be freely estimated: (now smdfslv intercepts are freely estimated)

model\_egaliwelfare\_scalar<-'

egali=~c(L1,L1)\*gincdif+c(L2,L2)\*dfincac+c(L3,L3)\*smdfslv

gincdif|c(T1\_1,T1\_1)\*t1

gincdif|c(T1\_2,T1\_2)\*t2

gincdif|c(T1\_3,T1\_3)\*t3

gincdif|c(T1\_4,T1\_4)\*t4

dfincac|c(T2\_1,T2\_1)\*t1

dfincac|c(T2\_2,T2\_2)\*t2

dfincac|c(T2\_3,T2\_3)\*t3

dfincac|c(T2\_4,T2\_4)\*t4

'

**Missing Values - Listwise/Case-wise deletion/Imputation**

* To handle listwise in cfa model, use argument missing
* For listwise:

NameOfCFA\_listwise<-cfa(model=ModelName, data=NameOfData, **missing=”listwise”**)

Example:

fit\_criticism\_listwise<-cfa(model = model\_criticsm,

data = ESS4\_BE,

missing = "listwise")

summary(fit\_criticism\_listwise,standardized=TRUE)

* For case-wise (FIML):

NameOfCFA\_listwise<-cfa(model=ModelName, data=NameOfData, **missing=”direct”**)

Example:

fit\_criticism\_FIML<-cfa(model = model\_criticsm,

data = ESS4\_BE,

missing="direct")

summary(fit\_criticism\_FIML,standardized=TRUE)

* Case-wise combining with weighted for normality (MLR):

NameOfCFA\_listwise<-cfa(model=ModelName, data=NameOfData, **missing=”direct”, estimator=”MLR”**)

Example:

fit\_criticism\_FIML\_MLR<-cfa(model = model\_criticsm,

data = ESS4\_BE,

missing="direct",

estimator="MLR")

summary(fit\_criticism\_FIML\_MLR, fit.measures=TRUE, standardized=TRUE)

* Imputation:
* First select the variables we need for the imputation = the variables we need for the cfa model (If the data is read in through SAV file, we need to coerce it in a normal data frame)

Example:

ESS4\_wsCriticism<-select(ESS4\_BE,sbstrec,sbbsntx,sbprvpv, sbeqsoc, sbcwkfm,sblazy, sblwcoa, sblwlka)

ESS4\_wsCriticism<-as.data.frame(ESS4\_wsCriticism)

* Then, we feed the above data frame into imputation:

dataImputedName<-amelia(NameOfData, m=NumberOfImputation, seed=SpecificNumber)

Example:

wsCriticism\_imputeOutput<-amelia(ESS4\_wsCriticism,

m=10,

seed=100)

* To double check the NA is really computed:

cbind(NameOfData$VarName, ImputedDataName$imputations$imp1$VarName)

Example: cbind(ESS4\_wsCriticism$sbbsntx,wsCriticism\_imputeOutput$imputations$imp1$sbbsntx)[c(41:57),]

* Run CFA model with semTools:

NameOfCFAImpute<-semTools::runMI(model = NameOfModel, data=dataImputedName$imputations, fun=”cfa”, estimator=”MLR”)

Example:

fit\_criticism\_impute<-semTools::runMI(model = model\_criticsm,

data = wsCriticism\_imputeOutput$imputations,

fun = "cfa",

estimator="MLR")

summary(fit\_criticism\_impute,fit.measures=TRUE, standardized=TRUE)

Predicted Latent Score:

https://rdrr.io/cran/lavaan/src/R/lav\_predict.R

Extra:

* Chi-square difference test: pchisq(q=Chi2statistics, df=DegreeOfFreedom, lower.tail=FALSE)
* Chi-square difference test better way: anova(NameOfCFAfit1, NameOfCFAfit2)

To sink everything inside the a txt file in the working directory

* #sinks the data into text file

sink("sink.txt")

#prints numbers from 1 to 5

for (i in 1:5)

print(i)

sink()

* Alberto Stefanelli

**7:16 PM**

sink("sink.txt")

summary(lav\_mod)

sink()