SEM_final_project

Benjamin Šimsa

5/5/2022

This script is a part of the final assignment for the Structural Equations course at KU Leuven, academic year 2021/2022. Lecturers: Prof. Bart Meuleman and Alberto Stefanelli.

Acknowledgment: Large portions of the following code were adapted from scripts written by Alberto Stefanelli; https://albertostefanelli.github.io/SEM_labs/

Part 1: Preliminaries and Descriptives

Import the packages, install if necessary

```
packages <-lapply (c('tidyverse',
                     'lavaan',
                     'psych',
                     'gridExtra',
                     'mvnTest',
                     'semTools',
                     'tidySEM',
                     'haven',
                     'ggpubr',
                     'prettydoc'),
        FUN = function (x) {
          if (! require (x, character.only = T)) {
            install.packages(x, dependencies = T)
            library(x, character.only = T)
          }
        }
        )
```

Import data

```
data = read.csv('https://raw.githubusercontent.com/benjsimsa/SEM-assignment/main/ess_data.csv')
```

Data processing

1) Add a factor that tells us whether the country belongs to the Eastern/Central Europe (CEE) or Western Europe (WE) group
2) Transform missing data to NAs

```r
data_filtered = data %>%
  mutate(cntry_grp = case_when(
    cntry %in%
      c("BE", "DE", "FI", "FR", "DK", "GR", "IR", "NO", "NE", "PO", "ES", "CH", "UK", "SE") ~ "WE",
    cntry %in% c("BG", "CZ", "EE", "HR", "PL", "SI", "SK") ~ "CEE"
                                  )) %>%
  filter(cntry_grp != is.na(cntry_grp)) %>%
  mutate_all(~na_if(., 66)) %>%  # Transforming missing values to NAs
  mutate_all(~na_if(., 77)) %>%
  mutate_all(~na_if(., 88)) %>%
  mutate_all(~na_if(., 99)) %>%
  dplyr:::mutate(gndr = na_if(gndr, 9)) # Convert gender non-response to NAs

data_filtered$cntry_grp = as.factor(data_filtered$cntry_grp)
```

Descriptives

Get a table with numbers of participants for each country and country group:

```r
(cntry_samplesize = table(data_filtered[, "cntry"]))
```

```
##
##   BE   BG   CH   CZ   DE   DK   EE   ES   FI   FR   GR   HR   NO   PL   SE   SI
## 1704 2434 1506 2386 3031 1576 1793 1885 1878 1728 2715 1649 1548 1751 1497 1403
##   SK
## 1856
```

```r
(group_samplesize = table(data_filtered[, "cntry_grp"]))
```

```
##
##   CEE    WE
## 13272 19068
```

```r
# save the table
# write.table(cntry_samplesize,
#             "participants_countries.csv",
#             sep = ",")
```

Get an insight into the structure of the data

```r
data_full_todescribe = data_filtered %>%
  select(ppltrst, pplfair, pplhlp, trstprl, trstlgl, trstplc, trstplt, trstprt, happy, gndr, hinctnta)

descriptive_ess <- as.data.frame(psych::describe(data_full_todescribe))

(descriptive_ess <- descriptive_ess %>% select(n,
                                  mean,
                                  sd,
                                  median,
                                  min,
```

```
                                max,
                                skew,
                                kurtosis) %>%
                    mutate(across(where(is.numeric), ~ round(., 1))))
```

```
##              n mean  sd median min max skew kurtosis
## ppltrst  32242  4.9 2.5      5   0  10 -0.2     -0.7
## pplfair  32146  5.5 2.4      6   0  10 -0.4     -0.4
## pplhlp   32203  4.7 2.4      5   0  10 -0.1     -0.6
## trstprl  31583  4.0 2.6      4   0  10  0.1     -0.9
## trstlgl  31545  4.8 2.7      5   0  10 -0.2     -0.9
## trstplc  31955  5.8 2.6      6   0  10 -0.5     -0.5
## trstplt  31781  3.2 2.4      3   0  10  0.3     -0.8
## trstprt  31616  3.2 2.4      3   0  10  0.3     -0.8
## happy    32123  7.1 2.0      8   0  10 -0.9      0.9
## gndr     32327  1.5 0.5      2   1   2 -0.1     -2.0
## hinctnta 26057  5.3 2.8      5   1  10  0.1     -1.1
```

```
# save the table

#write.table(descriptive_ess,
#            "descriptives.csv",
#            sep = ",")
```

How many missing datapoints are there for each variable / country group

```
data_w = data_filtered %>%
  filter(cntry_grp == "WE") %>%
  select(ppltrst, pplfair, pplhlp, trstprl, trstlgl, trstplc, trstplt, trstprt, happy, gndr, hinctnta)

data_cee = data_filtered %>%
  filter(cntry_grp == "CEE") %>%
  select(ppltrst, pplfair, pplhlp, trstprl, trstlgl, trstplc, trstplt, trstprt, happy, gndr, hinctnta)

data_full_na = data_filtered %>%
  select(ppltrst, pplfair, pplhlp, trstprl, trstlgl, trstplc, trstplt, trstprt, happy, gndr, hinctnta)

na_count_w = sapply(data_w, function(y) sum(length(which(is.na(y)))))
na_count_ce = sapply(data_cee, function(y) sum(length(which(is.na(y)))))
na_count_total = sapply(data_full_na, function(y) sum(length(which(is.na(y)))))

na_count_w = data.frame(na_count_w)
na_count_ce = data.frame(na_count_ce)
na_count_total = data.frame(na_count_total)

(missing_table = cbind(na_count_w, na_count_ce, na_count_total))
```

```
##          na_count_w na_count_ce na_count_total
## ppltrst          24          74             98
## pplfair          61         133            194
```

```
## pplhlp           35         102            137
## trstprl         358         399            757
## trstlgl         265         530            795
## trstplc          78         307            385
## trstplt         224         335            559
## trstprt         293         431            724
## happy            64         153            217
## gndr              0          13             13
## hinctnta       3111        3172           6283
```

```
(missing_table = missing_table %>% mutate(
  na_perc_w = (na_count_w/nrow(data_w))*100,
  na_perc_cee = na_count_ce/nrow(data_cee)*100,
  na_perc_total = na_count_total/nrow(data_full_na)*100
) %>%
  mutate(across(where(is.numeric), ~ round(., 1))) 
  %>%
  relocate(na_count_w, na_perc_w, na_count_ce, na_perc_cee, na_count_total, na_perc_total)
  )
```

```
##          na_count_w na_perc_w na_count_ce na_perc_cee na_count_total
## ppltrst          24       0.1          74         0.6             98
## pplfair          61       0.3         133         1.0            194
## pplhlp           35       0.2         102         0.8            137
## trstprl         358       1.9         399         3.0            757
## trstlgl         265       1.4         530         4.0            795
## trstplc          78       0.4         307         2.3            385
## trstplt         224       1.2         335         2.5            559
## trstprt         293       1.5         431         3.2            724
## happy            64       0.3         153         1.2            217
## gndr              0       0.0          13         0.1             13
## hinctnta       3111      16.3        3172        23.9           6283
##          na_perc_total
## ppltrst            0.3
## pplfair            0.6
## pplhlp             0.4
## trstprl            2.3
## trstlgl            2.5
## trstplc            1.2
## trstplt            1.7
## trstprt            2.2
## happy              0.7
## gndr               0.0
## hinctnta          19.4
```

```
# export the table to csv
#write.table(missing_table,
#            "na_table.csv",
#            sep = ",")
```

Assumption check: Univariate normality of endogenous variables

Plot each variable to visually assess univariate normality

```
lgl_plot <- ggplot(data_filtered, aes(trstlgl)) +
        geom_blank() +
        geom_histogram(aes(y = ..density..), binwidth = 1, colour = "black", alpha=0.3) + theme_minimal

plc_plot <- ggplot(data_filtered, aes(trstplc)) +
        geom_blank() +
        geom_histogram(aes(y = ..density..), binwidth = 1, colour = "black", alpha=0.3) + theme_minimal

plt_plot <- ggplot(data_filtered, aes(trstplt)) +
        geom_blank() +
        geom_histogram(aes(y = ..density..), binwidth = 1, colour = "black", alpha=0.3) + theme_minimal

prt_plot <- ggplot(data_filtered, aes(trstprt)) +
        geom_blank() +
        geom_histogram(aes(y = ..density..), binwidth = 1, colour = "black", alpha=0.3) + theme_minimal

soc_trst_plot <- ggplot(data_filtered, aes(ppltrst)) +
        geom_blank() +
        geom_histogram(aes(y = ..density..), binwidth = 1, colour = "black", alpha=0.3) + theme_minimal

soc_fair_plot <- ggplot(data_filtered, aes(pplfair)) +
        geom_blank() +
        geom_histogram(aes(y = ..density..), binwidth = 1, colour = "black", alpha=0.3) + theme_minimal

soc_help_plot <- ggplot(data_filtered, aes(pplhlp)) +
        geom_blank() +
        geom_histogram(aes(y = ..density..), binwidth = 1, colour = "black", alpha=0.3) + theme_minimal

normality_plots = ggarrange(lgl_plot, plc_plot, plt_plot, prt_plot, soc_trst_plot, soc_fair_plot, soc_h
          ncol = 3, nrow = 3)


## Save the plots
# ggsave("normplots.png",
#      normality_plots)
```

Most of the variables appear to be highly skewed / deviate from univariate normality.

The data violate the univariate normality assumption (hence also the multivariate normality assumption). The robust ML estimator will thus be used for subsequent analyses.

Part 2: Evaluating measurement models

Simultaneous measurement model: Trust in institutions and Social trust

```
measurement_model <- 'trust_inst =~ trstlgl + trstplc + trstplt + trstprt
                      trust_soc =~ ppltrst + pplfair + pplhlp'

measurement_fit <- cfa(measurement_model,
              data = data_filtered,
              estimator = "MLR",
```

```
            missing = 'direct'
)
summary(measurement_fit)
```

```
## lavaan 0.6-10 ended normally after 48 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        22
##
##                                                   Used       Total
##   Number of observations                          32327       32340
##   Number of missing patterns                         55
##
## Model Test User Model:
##                                                Standard      Robust
##   Test Statistic                              12352.599    9164.592
##   Degrees of freedom                                 13          13
##   P-value (Chi-square)                            0.000       0.000
##   Scaling correction factor                                   1.348
##       Yuan-Bentler correction (Mplus variant)
##
## Parameter Estimates:
##
##   Standard errors                             Sandwich
##   Information bread                           Observed
##   Observed information based on                Hessian
##
## Latent Variables:
##                   Estimate  Std.Err  z-value  P(>|z|)
##   trust_inst =~
##     trstlgl          1.000
##     trstplc          0.810    0.006  137.386    0.000
##     trstplt          1.213    0.009  134.908    0.000
##     trstprt          1.174    0.009  133.560    0.000
##   trust_soc =~
##     ppltrst          1.000
##     pplfair          0.975    0.008  116.005    0.000
##     pplhlp           0.860    0.008  102.722    0.000
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)
##   trust_inst ~~
##     trust_soc        1.898    0.031   61.201    0.000
##
## Intercepts:
##                   Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl          4.830    0.015  313.718    0.000
##    .trstplc          5.824    0.015  395.145    0.000
##    .trstplt          3.204    0.013  237.881    0.000
##    .trstprt          3.197    0.013  240.378    0.000
##    .ppltrst          4.937    0.014  358.982    0.000
##    .pplfair          5.500    0.013  419.333    0.000
```

```
##    .pplhlp           4.683    0.013  356.992    0.000
##     trust_inst       0.000
##     trust_soc        0.000
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl          4.024    0.040  100.167    0.000
##    .trstplc          4.637    0.040  115.401    0.000
##    .trstplt          0.621    0.018   34.379    0.000
##    .trstprt          0.800    0.019   42.645    0.000
##    .ppltrst          2.471    0.040   62.324    0.000
##    .pplfair          2.086    0.038   54.979    0.000
##    .pplhlp           2.858    0.036   79.179    0.000
##     trust_inst       3.520    0.050   69.832    0.000
##     trust_soc        3.633    0.049   74.840    0.000
```

```
fitMeasures(measurement_fit, c("chisq", "df", "pvalue", "cfi", "rmsea", "srmr", "rmsea.ci.lower", "rmsea
```

```
##
## chisq          12352.599
## df                13.000
## pvalue             0.000
## cfi                0.901
## rmsea              0.171
## srmr               0.070
## rmsea.ci.lower     0.169
## rmsea.ci.upper     0.174
```

Modify the model (drawing from theoretical considerations and modification indices)

```
mi_measurement <- modificationIndices(measurement_fit)
(mi_measurement_sorted <- mi_measurement[order(-mi_measurement$mi),])
```

```
##           lhs op     rhs        mi    epc sepc.lv sepc.all sepc.nox
## 45    trstplt ~~ trstprt 11265.124  3.546   3.546    5.028    5.028
## 34    trstlgl ~~ trstplc  9095.149  2.445   2.445    0.566    0.566
## 30  trust_soc =~ trstlgl  1262.608  0.300   0.572    0.208    0.208
## 31  trust_soc =~ trstplc  1158.608  0.303   0.577    0.219    0.219
## 41    trstplc ~~ trstprt  1107.714 -0.544  -0.544   -0.282   -0.282
## 35    trstlgl ~~ trstplt   976.436 -0.594  -0.594   -0.376   -0.376
## 36    trstlgl ~~ trstprt   877.812 -0.545  -0.545   -0.304   -0.304
## 40    trstplc ~~ trstplt   850.037 -0.486  -0.486   -0.286   -0.286
## 32  trust_soc =~ trstplt   272.178 -0.084  -0.160   -0.066   -0.066
## 37    trstlgl ~~ ppltrst   232.680  0.327   0.327    0.104    0.104
## 43    trstplc ~~ pplfair   226.612  0.320   0.320    0.103    0.103
## 33  trust_soc =~ trstprt   208.540 -0.073  -0.138   -0.058   -0.058
## 38    trstlgl ~~ pplfair   179.063  0.270   0.270    0.093    0.093
## 42    trstplc ~~ ppltrst    91.708  0.216   0.216    0.064    0.064
## 29 trust_inst =~  pplhlp    87.991  0.073   0.136    0.058    0.058
## 52    ppltrst ~~ pplfair    87.987  0.399   0.399    0.176    0.176
## 44    trstplc ~~  pplhlp    87.266  0.213   0.213    0.059    0.059
```

```
## 50    trstprt ~~ pplfair    75.373 -0.096  -0.096   -0.075   -0.075
## 28 trust_inst =~ pplfair    66.330 -0.065  -0.123   -0.052   -0.052
## 53    ppltrst ~~  pplhlp    66.316 -0.280  -0.280   -0.105   -0.105
## 46    trstplt ~~ ppltrst    51.194 -0.084  -0.084   -0.067   -0.067
## 47    trstplt ~~ pplfair    42.638 -0.072  -0.072   -0.063   -0.063
## 39    trstlgl ~~  pplhlp    40.321  0.137   0.137    0.041    0.041
## 49    trstprt ~~ ppltrst    11.099 -0.039  -0.039   -0.028   -0.028
## 51    trstprt ~~  pplhlp     0.547 -0.009  -0.009   -0.006   -0.006
## 48    trstplt ~~  pplhlp     0.498 -0.008  -0.008   -0.006   -0.006
## 54    pplfair ~~  pplhlp     0.194 -0.015  -0.015   -0.006   -0.006
## 27 trust_inst =~ ppltrst     0.194 -0.004  -0.007   -0.003   -0.003
```

Out of the three modification suggestions with highest MI, allowing for trust in politicians and trust in political parties makes the most theoretical sense. Let us modify the model and assess model fit

```
measurement_model_mod <-'trust_inst =~ trstlgl + trstplc + trstplt + trstprt
                         trust_soc =~ ppltrst + pplfair + pplhlp
                         trstplt ~~ trstprt'

fit_measurement_mod <- cfa(measurement_model_mod,
              data = data_filtered,
              estimator = "MLR",
              missing = "direct"
)


# Get summary, fit measures and parameter estimates

summary(fit_measurement_mod)
```

```
## lavaan 0.6-10 ended normally after 55 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        23
##
##                                                   Used       Total
##   Number of observations                         32327       32340
##   Number of missing patterns                        55
##
## Model Test User Model:
##                                                Standard      Robust
##   Test Statistic                                788.650     604.551
##   Degrees of freedom                                 12          12
##   P-value (Chi-square)                            0.000       0.000
##   Scaling correction factor                                   1.305
##         Yuan-Bentler correction (Mplus variant)
##
## Parameter Estimates:
##
##   Standard errors                             Sandwich
##   Information bread                           Observed
##   Observed information based on                Hessian
```

```
##
## Latent Variables:
##                   Estimate  Std.Err  z-value  P(>|z|)
##   trust_inst =~
##     trstlgl          1.000
##     trstplc          0.832    0.006  149.812    0.000
##     trstplt          0.686    0.005  135.338    0.000
##     trstprt          0.654    0.005  130.652    0.000
##   trust_soc =~
##     ppltrst          1.000
##     pplfair          0.976    0.008  117.178    0.000
##     pplhlp           0.856    0.008  104.275    0.000
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)
##  .trstplt ~~
##    .trstprt          2.356    0.030   79.402    0.000
##   trust_inst ~~
##     trust_soc        2.750    0.036   77.291    0.000
##
## Intercepts:
##                   Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl          4.833    0.015  314.278    0.000
##    .trstplc          5.821    0.015  394.928    0.000
##    .trstplt          3.204    0.013  237.903    0.000
##    .trstprt          3.197    0.013  240.372    0.000
##    .ppltrst          4.937    0.014  358.967    0.000
##    .pplfair          5.500    0.013  419.309    0.000
##    .pplhlp           4.683    0.013  356.976    0.000
##     trust_inst       0.000
##     trust_soc        0.000
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl          1.513    0.038   39.977    0.000
##    .trstplc          2.784    0.037   74.640    0.000
##    .trstplt          2.963    0.032   91.567    0.000
##    .trstprt          3.069    0.032   94.537    0.000
##    .ppltrst          2.463    0.039   62.553    0.000
##    .pplfair          2.073    0.037   55.352    0.000
##    .pplhlp           2.878    0.036   80.807    0.000
##     trust_inst       6.024    0.055  108.624    0.000
##     trust_soc        3.641    0.048   75.445    0.000
```

```
fitMeasures(fit_measurement_mod, c("chisq", "df", "pvalue", "cfi", "rmsea", "rmsea.ci.lower", "rmsea.ci
```

```
##
## chisq          788.650
## df              12.000
## pvalue           0.000
## cfi              0.994
## rmsea            0.045
## rmsea.ci.lower   0.042
## rmsea.ci.upper   0.047
```

```
## srmr              0.026
```

```
parameters_measurement_std = as.data.frame(parameterEstimates(fit_measurement_mod,
                                                              standardized = TRUE))

parameters_measurement_std = parameters_measurement_std %>% mutate(across(where(is.numeric), ~ round(.,

# write.table(parameters_measurement_std,
#             "parameters_measurement_std.csv",
#             sep = ",")
```

The model has acceptable fit according to the pre-set cut-off values.

All standardized factors loadings are above 0.50.

Measurement invariance: Trust in institutions

Let us assess measurement invariance of trust in institutions between CEE / Western countries

```
# Measurement model for institutional trust
inst_model_mod <- 'trust_inst =~ trstlgl + trstplc + trstplt + trstprt
                     trstplt ~~ trstprt'


# configural invariance
  fit_inst_con <- cfa(inst_model_mod,
                      data = data_filtered,
                      group = "cntry_grp",
                      estimator = "MLR",
                      missing = "direct"
  )


  # metric

  fit_inst_met <- cfa(inst_model_mod,
                      data = data_filtered,
                      group = "cntry_grp",
                      group.equal = c("loadings"),
                      estimator = "MLR",
                      missing = "direct"
  )

  # scalar
  fit_inst_sca <- cfa(inst_model_mod,
                      data = data_filtered,
                      group = "cntry_grp",
                      group.equal = c("loadings",
                                      "intercepts"),
                      estimator = "MLR",
                      missing = "direct"
  )
```

```r
# strict
fit_inst_stri <- cfa(inst_model_mod,
                     data = data_filtered,
                     group = "cntry_grp",
                     group.equal = c("loadings",
                                     "intercepts",
                                     "residuals"),
                     estimator = "MLR",
                     missing = "direct"
)

# structural
fit_inst_stru <- cfa(inst_model_mod,
                     data = data_filtered,
                     group = "cntry_grp",
                     group.equal = c("loadings",
                                     "intercepts",
                                     "residuals",
                                     "lv.variances",
                                     "lv.covariances"),
                     estimator = "MLR",
                     missing = "direct"
)

# compare
model_fit <-  function(lavobject) {
  vars <- c("df", "cfi", "tli", "rmsea", "rmsea.ci.lower", "rmsea.ci.upper", "srmr")
  return(fitmeasures(lavobject)[vars] %>% data.frame() %>% t())
}

table_fit <-
  list(model_fit(fit_inst_con),
       model_fit(fit_inst_met),
       model_fit(fit_inst_sca),
       model_fit(fit_inst_stri),
       model_fit(fit_inst_stru)) %>%
  reduce(rbind)


rownames(table_fit) <- c("Configural", "Metric", "Scalar", "Strict", "Structural")
table_fit = as.data.frame(table_fit)
(table_fit = table_fit %>% mutate(across(where(is.numeric), ~ round(., 3))))
```

```
##            df   cfi   tli rmsea rmsea.ci.lower rmsea.ci.upper  srmr
## Configural  2 1.000 0.998 0.026          0.018          0.036 0.002
## Metric      5 0.999 0.998 0.031          0.025          0.037 0.017
## Scalar      8 0.998 0.996 0.039          0.034          0.044 0.022
## Strict     12 0.993 0.993 0.052          0.049          0.056 0.048
## Structural 13 0.993 0.994 0.050          0.047          0.054 0.049
```

```r
#  write.table(table_fit,
#              "invariance_fit.csv",
#              sep = ",")
```

```r
table_anova <- list(anova(fit_inst_con, fit_inst_met),
                    anova(fit_inst_met, fit_inst_sca),
                    anova(fit_inst_sca, fit_inst_stri),
                    anova(fit_inst_stri, fit_inst_stru)
) %>%
  reduce(rbind) %>%
  .[-c(3, 5, 7),]

table_anova
```

```
## Scaled Chi-Squared Difference Test (method = "satorra.bentler.2001")
##
## lavaan NOTE:
##     The "Chisq" column contains standard test statistics, not the
##     robust test that should be reported per model. A robust difference
##     test is a function of two standard (not robust) statistics.
##
##               Df    AIC    BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## fit_inst_con   2 505523 505741  24.330
## fit_inst_met   5 505573 505766  80.701     52.460       3   2.39e-11 ***
## fit_inst_sca   8 505690 505858 203.471    111.432       3  < 2.2e-16 ***
## fit_inst_stri 12 506018 506152 539.044    233.083       4  < 2.2e-16 ***
## fit_inst_stru 13 506016 506142 539.290      0.325       1     0.5685
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#  table_anova = as.data.frame(table_anova)
#  table_anova = table_anova %>% mutate(across(where(is.numeric), ~ round(., 3)))
#  write.table(table_anova,
#              "chisq_fit.csv",
#              sep = ",")
```

Chi-square test results: metric measurement invariance rejected

Results using Chen's (2007) cut-off values: scalar invariance not rejected

Still, we can see whether we can achieve at least partial structural invariance using the chi-square test by releasing the constraints for factor loadings, parameter-by-parameter

Partial measurement invariance: Trust in institutions

```r
lavTestScore(fit_inst_met)
```

```
## $test
##
## total score test:
##
##    test     X2 df p.value
## 1 score 55.855  3       0
##
## $uni
```

```
##
## univariate score tests:
##
##    lhs op   rhs     X2 df p.value
## 1 .p2. == .p17. 38.103  1   0.000
## 2 .p3. == .p18.  0.389  1   0.533
## 3 .p4. == .p19.  9.410  1   0.002
```

```
parTable(fit_inst_met)
```

```
##    id        lhs op        rhs user block group free ustart exo label plabel
## 1   1 trust_inst =~    trstlgl    1     1     1    0      1   0         .p1.
## 2   2 trust_inst =~    trstplc    1     1     1    1     NA   0  .p2.   .p2.
## 3   3 trust_inst =~    trstplt    1     1     1    2     NA   0  .p3.   .p3.
## 4   4 trust_inst =~    trstprt    1     1     1    3     NA   0  .p4.   .p4.
## 5   5    trstplt ~~    trstprt    1     1     1    4     NA   0         .p5.
## 6   6    trstlgl ~~    trstlgl    0     1     1    5     NA   0         .p6.
## 7   7    trstplc ~~    trstplc    0     1     1    6     NA   0         .p7.
## 8   8    trstplt ~~    trstplt    0     1     1    7     NA   0         .p8.
## 9   9    trstprt ~~    trstprt    0     1     1    8     NA   0         .p9.
## 10 10 trust_inst ~~ trust_inst    0     1     1    9     NA   0        .p10.
## 11 11    trstlgl ~1               0     1     1   10     NA   0        .p11.
## 12 12    trstplc ~1               0     1     1   11     NA   0        .p12.
## 13 13    trstplt ~1               0     1     1   12     NA   0        .p13.
## 14 14    trstprt ~1               0     1     1   13     NA   0        .p14.
## 15 15 trust_inst ~1               0     1     1    0      0   0        .p15.
## 16 16 trust_inst =~    trstlgl    1     2     2    0      1   0        .p16.
## 17 17 trust_inst =~    trstplc    1     2     2   14     NA   0  .p2.   .p17.
## 18 18 trust_inst =~    trstplt    1     2     2   15     NA   0  .p3.   .p18.
## 19 19 trust_inst =~    trstprt    1     2     2   16     NA   0  .p4.   .p19.
## 20 20    trstplt ~~    trstprt    1     2     2   17     NA   0         .p20.
## 21 21    trstlgl ~~    trstlgl    0     2     2   18     NA   0         .p21.
## 22 22    trstplc ~~    trstplc    0     2     2   19     NA   0         .p22.
## 23 23    trstplt ~~    trstplt    0     2     2   20     NA   0         .p23.
## 24 24    trstprt ~~    trstprt    0     2     2   21     NA   0         .p24.
## 25 25 trust_inst ~~ trust_inst    0     2     2   22     NA   0         .p25.
## 26 26    trstlgl ~1               0     2     2   23     NA   0         .p26.
## 27 27    trstplc ~1               0     2     2   24     NA   0         .p27.
## 28 28    trstplt ~1               0     2     2   25     NA   0         .p28.
## 29 29    trstprt ~1               0     2     2   26     NA   0         .p29.
## 30 30 trust_inst ~1               0     2     2    0      0   0         .p30.
## 31 31       .p2. ==      .p17.    2     0     0    0     NA   0
## 32 32       .p3. ==      .p18.    2     0     0    0     NA   0
## 33 33       .p4. ==      .p19.    2     0     0    0     NA   0
##    start   est    se
## 1  1.000 1.000 0.000
## 2  0.758 0.778 0.007
## 3  0.958 0.651 0.006
## 4  0.923 0.617 0.006
## 5  0.000 2.798 0.040
## 6  3.390 1.283 0.053
## 7  2.939 2.672 0.045
## 8  2.938 3.426 0.042
## 9  2.855 3.480 0.042
```

```
## 10 0.050 5.489 0.076
## 11 5.597 5.597 0.019
## 12 6.510 6.510 0.018
## 13 3.668 3.668 0.018
## 14 3.661 3.661 0.017
## 15 0.000 0.000 0.000
## 16 1.000 1.000 0.000
## 17 0.808 0.778 0.007
## 18 0.874 0.651 0.006
## 19 0.840 0.617 0.006
## 20 0.000 2.085 0.043
## 21 3.268 0.988 0.061
## 22 3.411 3.258 0.058
## 23 2.444 2.680 0.048
## 24 2.374 2.823 0.049
## 25 0.050 5.561 0.083
## 26 3.724 3.723 0.023
## 27 4.824 4.824 0.023
## 28 2.532 2.532 0.019
## 29 2.526 2.526 0.019
## 30 0.000 0.000 0.000
## 31 0.000 0.000 0.000
## 32 0.000 0.000 0.000
## 33 0.000 0.000 0.000
```

```
# We see that setting factor loading of trust in police equal across the two groups causes the most tro

  fit_inst_met_partial <- cfa(inst_model_mod,
                      data = data_filtered,
                      group = "cntry_grp",
                      group.equal = c("loadings"),
                      estimator = "MLR",
                      missing = "direct",
                      group.partial = c("trust_inst =~ trstplc",
                                        "trust_inst =~ trstplt")
  )


  lavTestScore(fit_inst_met_partial)
```

```
## $test
##
## total score test:
##
##    test     X2 df p.value
## 1 score 17.596  1       0
##
## $uni
##
## univariate score tests:
##
##    lhs op   rhs     X2 df p.value
## 1 .p4. == .p19. 17.596  1       0
```

```r
  table_anova <- list(anova(fit_inst_con, fit_inst_met_partial),
                      anova(fit_inst_met_partial, fit_inst_sca),
                      anova(fit_inst_sca, fit_inst_stri),
                      anova(fit_inst_stri, fit_inst_stru)
  ) %>%
    reduce(rbind) %>%
    .[-c(3, 5, 7),]

  table_anova
```

```
## Scaled Chi-Squared Difference Test (method = "satorra.bentler.2001")
##
## lavaan NOTE:
##     The "Chisq" column contains standard test statistics, not the
##     robust test that should be reported per model. A robust difference
##     test is a function of two standard (not robust) statistics.
##
##                       Df    AIC    BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## fit_inst_con           2 505523 505741  24.330
## fit_inst_met_partial   3 505539 505748  41.816     16.630       1  4.542e-05 ***
## fit_inst_sca           8 505690 505858 203.471    147.565       5  < 2.2e-16 ***
## fit_inst_stri         12 506018 506152 539.044    233.083       4  < 2.2e-16 ***
## fit_inst_stru         13 506016 506142 539.290      0.325       1     0.5685
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Partial metric invariance (according to chi-square test) was not established.

Part 3: Is institutional trust higher in Western Europe than in ECE?

```r
fit_inst_sca <- cfa(inst_model_mod,
                    data = data_filtered,
                    group = "cntry_grp",
                    group.equal = c("loadings",
                                    "intercepts"),
                    estimator = "MLR",
                    missing = "direct",
                    meanstructure = TRUE
  )

summary(fit_inst_sca)
```

```
## lavaan 0.6-10 ended normally after 69 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        27
##   Number of equality constraints                     7
```

```
##
##   Number of observations per group:                Used       Total
##     WE                                            19031       19068
##     CEE                                           13114       13272
##   Number of missing patterns per group:
##     WE                                               15
##     CEE                                              15
##
## Model Test User Model:
##                                                   Standard      Robust
##   Test Statistic                                   203.471     169.043
##   Degrees of freedom                                     8           8
##   P-value (Chi-square)                               0.000       0.000
##   Scaling correction factor                                      1.204
##         Yuan-Bentler correction (Mplus variant)
##   Test statistic for each group:
##     WE                                             120.867     100.416
##     CEE                                             82.604      68.627
##
## Parameter Estimates:
##
##   Standard errors                                Sandwich
##   Information bread                              Observed
##   Observed information based on                   Hessian
##
##
## Group 1 [WE]:
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   trust_inst =~
##     trstlgl           1.000
##     trstplc (.p2.)    0.805    0.006  132.165    0.000
##     trstplt (.p3.)    0.649    0.005  131.357    0.000
##     trstprt (.p4.)    0.620    0.005  127.291    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .trstplt ~~
##    .trstprt           2.796    0.039   72.079    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl (.11.)    5.610    0.019  300.591    0.000
##    .trstplc (.12.)    6.454    0.017  388.644    0.000
##    .trstplt (.13.)    3.714    0.016  227.288    0.000
##    .trstprt (.14.)    3.684    0.016  230.458    0.000
##     trst_ns           0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl           1.369    0.051   27.096    0.000
##    .trstplc           2.616    0.045   58.278    0.000
##    .trstplt           3.428    0.041   83.196    0.000
```

```
##    .trstprt           3.476    0.041   84.348    0.000
##     trust_inst        5.372    0.073   73.638    0.000
##
##
## Group 2 [CEE]:
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   trust_inst =~
##     trstlgl           1.000
##     trstplc (.p2.)    0.805    0.006  132.165    0.000
##     trstplt (.p3.)    0.649    0.005  131.357    0.000
##     trstprt (.p4.)    0.620    0.005  127.291    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .trstplt ~~
##    .trstprt           2.076    0.042   49.621    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl (.11.)    5.610    0.019  300.591    0.000
##    .trstplc (.12.)    6.454    0.017  388.644    0.000
##    .trstplt (.13.)    3.714    0.016  227.288    0.000
##    .trstprt (.14.)    3.684    0.016  230.458    0.000
##     trst_ns          -1.903    0.029  -66.751    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl           1.053    0.055   19.101    0.000
##    .trstplc           3.215    0.059   54.173    0.000
##    .trstplt           2.674    0.047   56.982    0.000
##    .trstprt           2.812    0.048   58.163    0.000
##     trust_inst        5.472    0.077   70.783    0.000
```

The latent mean of Trust in institution is lower for ECE countries than for WE.

Part 4: MIMIC model

Base MIMIC model

```
mimic_model = 'trust_inst =~ trstlgl + trstplc + trstplt + trstprt
                        trstplt ~~ trstprt
                        trust_soc =~ ppltrst + pplfair + pplhlp
                        trust_inst ~ hinctnta   + happy + gndr + trust_soc'

mimic_fit <- sem(mimic_model,
              data = data_filtered ,
              estimator = "MLR",
              missing = "direct"
)
```

```r
summary(mimic_fit)
```

```
## lavaan 0.6-10 ended normally after 58 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        26
##
##                                                   Used       Total
##   Number of observations                          25925       32340
##   Number of missing patterns                         48
##
## Model Test User Model:
##                                                Standard      Robust
##   Test Statistic                               4898.907    4208.551
##   Degrees of freedom                                 30          30
##   P-value (Chi-square)                            0.000       0.000
##   Scaling correction factor                                   1.164
##         Yuan-Bentler correction (Mplus variant)
##
## Parameter Estimates:
##
##   Standard errors                             Sandwich
##   Information bread                           Observed
##   Observed information based on                Hessian
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   trust_inst =~
##     trstlgl           1.000
##     trstplc           0.829    0.006  136.311    0.000
##     trstplt           0.687    0.006  124.410    0.000
##     trstprt           0.654    0.006  118.867    0.000
##   trust_soc =~
##     ppltrst           1.000
##     pplfair           0.956    0.010  100.123    0.000
##     pplhlp            0.851    0.009   91.445    0.000
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   trust_inst ~
##     hinctnta          0.025    0.005    4.700    0.000
##     happy             0.247    0.008   29.756    0.000
##     gndr             -0.097    0.028   -3.476    0.001
##     trust_soc         0.662    0.010   63.250    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .trstplt ~~
##    .trstprt           2.330    0.033   71.241    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
```

```
##    .trstlgl           3.199    0.075   42.565    0.000
##    .trstplc           4.485    0.065   68.986    0.000
##    .trstplt           2.096    0.053   39.902    0.000
##    .trstprt           2.141    0.050   42.706    0.000
##    .ppltrst           5.057    0.015  330.754    0.000
##    .pplfair           5.624    0.014  389.207    0.000
##    .pplhlp            4.782    0.015  328.010    0.000
##    .trust_inst        0.000
##     trust_soc         0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .trstlgl           1.520    0.041   37.333    0.000
##    .trstplc           2.700    0.041   66.527    0.000
##    .trstplt           2.932    0.035   83.015    0.000
##    .trstprt           3.056    0.036   85.070    0.000
##    .ppltrst           2.423    0.045   54.331    0.000
##    .pplfair           2.079    0.042   49.338    0.000
##    .pplhlp            2.869    0.040   71.898    0.000
##    .trust_inst        3.679    0.059   62.806    0.000
##     trust_soc         3.629    0.055   66.509    0.000
```

```
fitMeasures(mimic_fit, c("chisq", "df", "pvalue", "cfi", "rmsea", "rmsea.ci.lower", "rmsea.ci.upper", "s
```

```
##
## chisq          4898.907
## df               30.000
## pvalue            0.000
## cfi               0.954
## rmsea             0.079
## rmsea.ci.lower    0.077
## rmsea.ci.upper    0.081
## srmr              0.091
```

```
# parameter estimates
parameters_mimic_std = as.data.frame(parameterEstimates(mimic_fit,
                                                        standardized = TRUE))
(parameters_mimic_std = parameters_mimic_std %>% mutate(across(where(is.numeric), ~ round(., 3))))
```

```
##           lhs op        rhs    est    se       z pvalue ci.lower ci.upper
## 1  trust_inst =~    trstlgl  1.000 0.000      NA     NA    1.000    1.000
## 2  trust_inst =~    trstplc  0.829 0.006 136.311  0.000    0.817    0.841
## 3  trust_inst =~    trstplt  0.687 0.006 124.410  0.000    0.676    0.698
## 4  trust_inst =~    trstprt  0.654 0.006 118.867  0.000    0.643    0.665
## 5     trstplt ~~    trstprt  2.330 0.033  71.241  0.000    2.266    2.394
## 6   trust_soc =~    ppltrst  1.000 0.000      NA     NA    1.000    1.000
## 7   trust_soc =~    pplfair  0.956 0.010 100.123  0.000    0.937    0.975
## 8   trust_soc =~     pplhlp  0.851 0.009  91.445  0.000    0.833    0.869
## 9  trust_inst  ~   hinctnta  0.025 0.005   4.700  0.000    0.015    0.035
## 10 trust_inst  ~      happy  0.247 0.008  29.756  0.000    0.231    0.264
## 11 trust_inst  ~       gndr -0.097 0.028  -3.476  0.001   -0.152   -0.042
## 12 trust_inst  ~ trust_soc  0.662 0.010  63.250  0.000    0.641    0.682
## 13    trstlgl ~~    trstlgl  1.520 0.041  37.333  0.000    1.440    1.600
```

```
## 14    trstplc ~~    trstplc  2.700 0.041  66.527  0.000    2.620    2.779
## 15    trstplt ~~    trstplt  2.932 0.035  83.015  0.000    2.863    3.001
## 16    trstprt ~~    trstprt  3.056 0.036  85.070  0.000    2.985    3.126
## 17    ppltrst ~~    ppltrst  2.423 0.045  54.331  0.000    2.335    2.510
## 18    pplfair ~~    pplfair  2.079 0.042  49.338  0.000    1.996    2.161
## 19     pplhlp ~~     pplhlp  2.869 0.040  71.898  0.000    2.791    2.948
## 20 trust_inst ~~ trust_inst  3.679 0.059  62.806  0.000    3.564    3.793
## 21  trust_soc ~~  trust_soc  3.629 0.055  66.509  0.000    3.522    3.736
## 22   hinctnta ~~   hinctnta  7.766 0.000      NA     NA    7.766    7.766
## 23   hinctnta ~~      happy  1.588 0.000      NA     NA    1.588    1.588
## 24   hinctnta ~~       gndr -0.131 0.000      NA     NA   -0.131   -0.131
## 25      happy ~~      happy  4.140 0.000      NA     NA    4.140    4.140
## 26      happy ~~       gndr -0.015 0.000      NA     NA   -0.015   -0.015
## 27       gndr ~~       gndr  0.249 0.000      NA     NA    0.249    0.249
## 28    trstlgl ~1             3.199 0.075  42.565  0.000    3.052    3.347
## 29    trstplc ~1             4.485 0.065  68.986  0.000    4.358    4.613
## 30    trstplt ~1             2.096 0.053  39.902  0.000    1.993    2.198
## 31    trstprt ~1             2.141 0.050  42.706  0.000    2.043    2.239
## 32    ppltrst ~1             5.057 0.015 330.754  0.000    5.027    5.087
## 33    pplfair ~1             5.624 0.014 389.207  0.000    5.596    5.652
## 34     pplhlp ~1             4.782 0.015 328.010  0.000    4.753    4.810
## 35   hinctnta ~1             5.276 0.000      NA     NA    5.276    5.276
## 36      happy ~1             7.141 0.000      NA     NA    7.141    7.141
## 37       gndr ~1             1.526 0.000      NA     NA    1.526    1.526
## 38 trust_inst ~1             0.000 0.000      NA     NA    0.000    0.000
## 39  trust_soc ~1             0.000 0.000      NA     NA    0.000    0.000
##    std.lv std.all std.nox
## 1   2.356   0.886   0.886
## 2   1.952   0.765   0.765
## 3   1.619   0.687   0.687
## 4   1.541   0.661   0.661
## 5   2.330   0.778   0.778
## 6   1.905   0.774   0.774
## 7   1.821   0.784   0.784
## 8   1.621   0.691   0.691
## 9   0.011   0.030   0.011
## 10  0.105   0.214   0.105
## 11 -0.041  -0.021  -0.041
## 12  0.535   0.535   0.535
## 13  1.520   0.215   0.215
## 14  2.700   0.415   0.415
## 15  2.932   0.528   0.528
## 16  3.056   0.563   0.563
## 17  2.423   0.400   0.400
## 18  2.079   0.385   0.385
## 19  2.869   0.522   0.522
## 20  0.663   0.663   0.663
## 21  1.000   1.000   1.000
## 22  7.766   1.000   7.766
## 23  1.588   0.280   1.588
## 24 -0.131  -0.094  -0.131
## 25  4.140   1.000   4.140
## 26 -0.015  -0.015  -0.015
## 27  0.249   1.000   0.249
```

```
## 28  3.199   1.203   1.203
## 29  4.485   1.758   1.758
## 30  2.096   0.889   0.889
## 31  2.141   0.919   0.919
## 32  5.057   2.056   2.056
## 33  5.624   2.421   2.421
## 34  4.782   2.039   2.039
## 35  5.276   1.893   5.276
## 36  7.141   3.510   7.141
## 37  1.526   3.057   1.526
## 38  0.000   0.000   0.000
## 39  0.000   0.000   0.000
```

```
# write.table(parameters_mimic_std,
#             "parameters_mimic_std.csv",
#             sep = ",")
```

Part 5: Interaction between predictors and CEE

```
mimic_model = 'trust_inst =~ trstlgl + trstplc + trstplt + trstprt
               trstplt ~~ trstprt
               trust_soc =~ ppltrst + pplfair + pplhlp
               trust_inst ~ hinctnta    + happy + gndr + trust_soc'

mimic_fit_moder <- sem(mimic_model,
              data = data_filtered ,
              estimator = "MLR",
              group = "cntry_grp",
              missing = "direct"
)

summary(mimic_fit_moder,
        standardized = TRUE)
```

```
## lavaan 0.6-10 ended normally after 120 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        52
##
##   Number of observations per group:               Used       Total
##     WE                                           15920       19068
##     CEE                                          10005       13272
##   Number of missing patterns per group:
##     WE                                              34
##     CEE                                             45
##
## Model Test User Model:
##                                                   Standard      Robust
##   Test Statistic                                4551.629    3933.987
##   Degrees of freedom                                  60          60
```

```
##   P-value (Chi-square)                            0.000       0.000
##   Scaling correction factor                                   1.157
##        Yuan-Bentler correction (Mplus variant)
##   Test statistic for each group:
##     WE                                         3446.386    2978.722
##     CEE                                        1105.243     955.265
##
## Parameter Estimates:
##
##   Standard errors                             Sandwich
##   Information bread                           Observed
##   Observed information based on                Hessian
##
##
## Group 1 [WE]:
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   trust_inst =~
##     trstlgl           1.000                               2.144    0.855
##     trstplc           0.796    0.009   89.532    0.000    1.706    0.732
##     trstplt           0.736    0.009   81.017    0.000    1.578    0.669
##     trstprt           0.702    0.009   77.959    0.000    1.505    0.647
##   trust_soc =~
##     ppltrst           1.000                               1.711    0.738
##     pplfair           0.967    0.014   71.283    0.000    1.654    0.763
##     pplhlp            0.873    0.014   64.117    0.000    1.495    0.673
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   trust_inst ~
##     hinctnta          0.043    0.006    6.890    0.000    0.020    0.055
##     happy             0.194    0.011   17.073    0.000    0.090    0.164
##     gndr             -0.110    0.033   -3.330    0.001   -0.051   -0.026
##     trust_soc         0.718    0.015   49.265    0.000    0.573    0.573
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .trstplt ~~
##    .trstprt           2.458    0.045   54.508    0.000    2.458    0.790
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .trstlgl           4.173    0.101   41.288    0.000    4.173    1.664
##    .trstplc           5.391    0.084   63.827    0.000    5.391    2.312
##    .trstplt           2.655    0.076   34.825    0.000    2.655    1.125
##    .trstprt           2.694    0.073   36.919    0.000    2.694    1.157
##    .ppltrst           5.472    0.018  297.780    0.000    5.472    2.361
##    .pplfair           6.033    0.017  351.026    0.000    6.033    2.785
##    .pplhlp            5.155    0.018  292.651    0.000    5.155    2.320
##    .trust_inst        0.000                               0.000    0.000
##     trust_soc         0.000                               0.000    0.000
##
## Variances:
```

```
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     .trstlgl          1.693    0.053   31.962    0.000    1.693    0.269
##     .trstplc          2.528    0.048   52.232    0.000    2.528    0.465
##     .trstplt          3.075    0.048   63.802    0.000    3.075    0.553
##     .trstprt          3.152    0.048   65.921    0.000    3.152    0.582
##     .ppltrst          2.445    0.054   45.080    0.000    2.445    0.455
##     .pplfair          1.958    0.049   39.988    0.000    1.958    0.417
##     .pplhlp           2.701    0.047   57.274    0.000    2.701    0.547
##     .trust_inst       2.927    0.071   41.335    0.000    0.637    0.637
##      trust_soc        2.928    0.064   46.043    0.000    1.000    1.000
##
##
## Group 2 [CEE]:
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   trust_inst =~
##     trstlgl           1.000                               2.272    0.897
##     trstplc           0.844    0.012   73.106    0.000    1.918    0.741
##     trstplt           0.645    0.010   64.960    0.000    1.465    0.672
##     trstprt           0.597    0.010   60.926    0.000    1.357    0.629
##   trust_soc =~
##     ppltrst           1.000                               2.017    0.796
##     pplfair           0.932    0.016   59.754    0.000    1.879    0.778
##     pplhlp            0.813    0.015   54.416    0.000    1.640    0.680
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   trust_inst ~
##     hinctnta          0.022    0.009    2.464    0.014    0.010    0.027
##     happy             0.195    0.012   16.624    0.000    0.086    0.190
##     gndr              0.112    0.046    2.431    0.015    0.049    0.024
##     trust_soc         0.463    0.015   30.386    0.000    0.411    0.411
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .trstplt ~~
##     .trstprt          2.034    0.049   41.596    0.000    2.034    0.751
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     .trstlgl          2.193    0.108   20.348    0.000    2.193    0.867
##     .trstplc          3.551    0.095   37.548    0.000    3.551    1.372
##     .trstplt          1.529    0.071   21.633    0.000    1.529    0.701
##     .trstprt          1.589    0.066   24.176    0.000    1.589    0.737
##     .ppltrst          4.396    0.025  173.274    0.000    4.396    1.734
##     .pplfair          4.973    0.024  205.536    0.000    4.973    2.060
##     .pplhlp           4.187    0.024  173.277    0.000    4.187    1.736
##     .trust_inst       0.000                               0.000    0.000
##      trust_soc        0.000                               0.000    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     .trstlgl          1.247    0.070   17.895    0.000    1.247    0.195
```

```
##    .trstplc           3.020    0.072   41.718    0.000    3.020    0.451
##    .trstplt           2.610    0.054   48.467    0.000    2.610    0.549
##    .trstprt           2.811    0.057   49.655    0.000    2.811    0.604
##    .ppltrst           2.357    0.079   29.829    0.000    2.357    0.367
##    .pplfair           2.297    0.077   29.731    0.000    2.297    0.394
##    .pplhlp            3.130    0.072   43.643    0.000    3.130    0.538
##    .trust_inst        4.079    0.097   41.927    0.000    0.791    0.791
##     trust_soc         4.067    0.096   42.276    0.000    1.000    1.000
```

```r
fitMeasures(mimic_fit_moder, c("chisq", "df", "pvalue", "cfi", "rmsea", "rmsea.ci.lower", "rmsea.ci.upp
```

```
##
## chisq          4551.629
## df               60.000
## pvalue            0.000
## cfi               0.953
## rmsea             0.076
## rmsea.ci.lower    0.074
## rmsea.ci.upper    0.078
## srmr              0.082
```

```r
# parameter estimates
parameters_mimic_multigroup = as.data.frame(parameterEstimates(mimic_fit_moder,
                                                    standardized = TRUE))

(parameters_mimic_multigroup = parameters_mimic_multigroup %>% mutate(across(where(is.numeric), ~ round
)
```

```
##           lhs op        rhs block group    est    se       z pvalue ci.lower
## 1  trust_inst =~    trstlgl     1     1  1.000 0.000      NA     NA    1.000
## 2  trust_inst =~    trstplc     1     1  0.796 0.009  89.532  0.000    0.778
## 3  trust_inst =~    trstplt     1     1  0.736 0.009  81.017  0.000    0.718
## 4  trust_inst =~    trstprt     1     1  0.702 0.009  77.959  0.000    0.684
## 5     trstplt ~~    trstprt     1     1  2.458 0.045  54.508  0.000    2.370
## 6   trust_soc =~    ppltrst     1     1  1.000 0.000      NA     NA    1.000
## 7   trust_soc =~    pplfair     1     1  0.967 0.014  71.283  0.000    0.940
## 8   trust_soc =~     pplhlp     1     1  0.873 0.014  64.117  0.000    0.847
## 9  trust_inst  ~   hinctnta     1     1  0.043 0.006   6.890  0.000    0.031
## 10 trust_inst  ~      happy     1     1  0.194 0.011  17.073  0.000    0.171
## 11 trust_inst  ~       gndr     1     1 -0.110 0.033  -3.330  0.001   -0.175
## 12 trust_inst  ~  trust_soc     1     1  0.718 0.015  49.265  0.000    0.689
## 13    trstlgl ~~    trstlgl     1     1  1.693 0.053  31.962  0.000    1.589
## 14    trstplc ~~    trstplc     1     1  2.528 0.048  52.232  0.000    2.433
## 15    trstplt ~~    trstplt     1     1  3.075 0.048  63.802  0.000    2.980
## 16    trstprt ~~    trstprt     1     1  3.152 0.048  65.921  0.000    3.058
## 17    ppltrst ~~    ppltrst     1     1  2.445 0.054  45.080  0.000    2.338
## 18    pplfair ~~    pplfair     1     1  1.958 0.049  39.988  0.000    1.862
## 19     pplhlp ~~     pplhlp     1     1  2.701 0.047  57.274  0.000    2.609
## 20 trust_inst ~~ trust_inst     1     1  2.927 0.071  41.335  0.000    2.788
## 21  trust_soc ~~  trust_soc     1     1  2.928 0.064  46.043  0.000    2.803
## 22   hinctnta ~~   hinctnta     1     1  7.582 0.000      NA     NA    7.582
## 23   hinctnta ~~      happy     1     1  1.249 0.000      NA     NA    1.249
## 24   hinctnta ~~       gndr     1     1 -0.122 0.000      NA     NA   -0.122
```

```
## 25      happy ~~      happy     1     1  3.283 0.000      NA     NA    3.283
## 26      happy ~~       gndr     1     1 -0.007 0.000      NA     NA   -0.007
## 27       gndr ~~       gndr     1     1  0.250 0.000      NA     NA    0.250
## 28    trstlgl ~1             1     1  4.173 0.101  41.288  0.000    3.975
## 29    trstplc ~1             1     1  5.391 0.084  63.827  0.000    5.225
## 30    trstplt ~1             1     1  2.655 0.076  34.825  0.000    2.505
## 31    trstprt ~1             1     1  2.694 0.073  36.919  0.000    2.551
## 32    ppltrst ~1             1     1  5.472 0.018 297.780  0.000    5.436
## 33    pplfair ~1             1     1  6.033 0.017 351.026  0.000    5.999
## 34     pplhlp ~1             1     1  5.155 0.018 292.651  0.000    5.120
## 35   hinctnta ~1             1     1  5.386 0.000      NA     NA    5.386
## 36      happy ~1             1     1  7.526 0.000      NA     NA    7.526
## 37       gndr ~1             1     1  1.505 0.000      NA     NA    1.505
## 38 trust_inst ~1             1     1  0.000 0.000      NA     NA    0.000
## 39  trust_soc ~1             1     1  0.000 0.000      NA     NA    0.000
## 40 trust_inst =~    trstlgl     2     2  1.000 0.000      NA     NA    1.000
## 41 trust_inst =~    trstplc     2     2  0.844 0.012  73.106  0.000    0.822
## 42 trust_inst =~    trstplt     2     2  0.645 0.010  64.960  0.000    0.625
## 43 trust_inst =~    trstprt     2     2  0.597 0.010  60.926  0.000    0.578
## 44    trstplt ~~    trstprt     2     2  2.034 0.049  41.596  0.000    1.939
## 45  trust_soc =~    ppltrst     2     2  1.000 0.000      NA     NA    1.000
## 46  trust_soc =~    pplfair     2     2  0.932 0.016  59.754  0.000    0.901
## 47  trust_soc =~     pplhlp     2     2  0.813 0.015  54.416  0.000    0.784
## 48 trust_inst  ~   hinctnta     2     2  0.022 0.009   2.464  0.014    0.004
## 49 trust_inst  ~      happy     2     2  0.195 0.012  16.624  0.000    0.172
## 50 trust_inst  ~       gndr     2     2  0.112 0.046   2.431  0.015    0.022
## 51 trust_inst  ~  trust_soc     2     2  0.463 0.015  30.386  0.000    0.433
## 52    trstlgl ~~    trstlgl     2     2  1.247 0.070  17.895  0.000    1.110
## 53    trstplc ~~    trstplc     2     2  3.020 0.072  41.718  0.000    2.878
## 54    trstplt ~~    trstplt     2     2  2.610 0.054  48.467  0.000    2.505
## 55    trstprt ~~    trstprt     2     2  2.811 0.057  49.655  0.000    2.700
## 56    ppltrst ~~    ppltrst     2     2  2.357 0.079  29.829  0.000    2.202
## 57    pplfair ~~    pplfair     2     2  2.297 0.077  29.731  0.000    2.145
## 58     pplhlp ~~     pplhlp     2     2  3.130 0.072  43.643  0.000    2.990
## 59 trust_inst ~~ trust_inst     2     2  4.079 0.097  41.927  0.000    3.888
## 60  trust_soc ~~  trust_soc     2     2  4.067 0.096  42.276  0.000    3.879
## 61   hinctnta ~~   hinctnta     2     2  8.008 0.000      NA     NA    8.008
## 62   hinctnta ~~      happy     2     2  1.950 0.000      NA     NA    1.950
## 63   hinctnta ~~       gndr     2     2 -0.136 0.000      NA     NA   -0.136
## 64      happy ~~      happy     2     2  4.892 0.000      NA     NA    4.892
## 65      happy ~~       gndr     2     2  0.005 0.000      NA     NA    0.005
## 66       gndr ~~       gndr     2     2  0.246 0.000      NA     NA    0.246
## 67    trstlgl ~1             2     2  2.193 0.108  20.348  0.000    1.982
## 68    trstplc ~1             2     2  3.551 0.095  37.548  0.000    3.366
## 69    trstplt ~1             2     2  1.529 0.071  21.633  0.000    1.391
## 70    trstprt ~1             2     2  1.589 0.066  24.176  0.000    1.460
## 71    ppltrst ~1             2     2  4.396 0.025 173.274  0.000    4.346
## 72    pplfair ~1             2     2  4.973 0.024 205.536  0.000    4.926
## 73     pplhlp ~1             2     2  4.187 0.024 173.277  0.000    4.140
## 74   hinctnta ~1             2     2  5.099 0.000      NA     NA    5.099
## 75      happy ~1             2     2  6.529 0.000      NA     NA    6.529
## 76       gndr ~1             2     2  1.560 0.000      NA     NA    1.560
## 77 trust_inst ~1             2     2  0.000 0.000      NA     NA    0.000
## 78  trust_soc ~1             2     2  0.000 0.000      NA     NA    0.000
```

```
##    ci.upper std.lv std.all std.nox
## 1     1.000  2.144   0.855   0.855
## 2     0.813  1.706   0.732   0.732
## 3     0.754  1.578   0.669   0.669
## 4     0.720  1.505   0.647   0.647
## 5     2.546  2.458   0.790   0.790
## 6     1.000  1.711   0.738   0.738
## 7     0.993  1.654   0.763   0.763
## 8     0.900  1.495   0.673   0.673
## 9     0.055  0.020   0.055   0.020
## 10    0.216  0.090   0.164   0.090
## 11   -0.045 -0.051  -0.026  -0.051
## 12    0.747  0.573   0.573   0.573
## 13    1.796  1.693   0.269   0.269
## 14    2.623  2.528   0.465   0.465
## 15    3.169  3.075   0.553   0.553
## 16    3.246  3.152   0.582   0.582
## 17    2.551  2.445   0.455   0.455
## 18    2.054  1.958   0.417   0.417
## 19    2.793  2.701   0.547   0.547
## 20    3.066  0.637   0.637   0.637
## 21    3.052  1.000   1.000   1.000
## 22    7.582  7.582   1.000   7.582
## 23    1.249  1.249   0.250   1.249
## 24   -0.122 -0.122  -0.088  -0.122
## 25    3.283  3.283   1.000   3.283
## 26   -0.007 -0.007  -0.007  -0.007
## 27    0.250  0.250   1.000   0.250
## 28    4.371  4.173   1.664   1.664
## 29    5.556  5.391   2.312   2.312
## 30    2.804  2.655   1.125   1.125
## 31    2.837  2.694   1.157   1.157
## 32    5.508  5.472   2.361   2.361
## 33    6.066  6.033   2.785   2.785
## 34    5.189  5.155   2.320   2.320
## 35    5.386  5.386   1.956   5.386
## 36    7.526  7.526   4.154   7.526
## 37    1.505  1.505   3.010   1.505
## 38    0.000  0.000   0.000   0.000
## 39    0.000  0.000   0.000   0.000
## 40    1.000  2.272   0.897   0.897
## 41    0.867  1.918   0.741   0.741
## 42    0.664  1.465   0.672   0.672
## 43    0.617  1.357   0.629   0.629
## 44    2.130  2.034   0.751   0.751
## 45    1.000  2.017   0.796   0.796
## 46    0.962  1.879   0.778   0.778
## 47    0.842  1.640   0.680   0.680
## 48    0.039  0.010   0.027   0.010
## 49    0.218  0.086   0.190   0.086
## 50    0.202  0.049   0.024   0.049
## 51    0.493  0.411   0.411   0.411
## 52    1.383  1.247   0.195   0.195
## 53    3.161  3.020   0.451   0.451
```

```
## 54    2.716  2.610   0.549   0.549
## 55    2.922  2.811   0.604   0.604
## 56    2.512  2.357   0.367   0.367
## 57    2.448  2.297   0.394   0.394
## 58    3.271  3.130   0.538   0.538
## 59    4.270  0.791   0.791   0.791
## 60    4.256  1.000   1.000   1.000
## 61    8.008  8.008   1.000   8.008
## 62    1.950  1.950   0.312   1.950
## 63   -0.136 -0.136  -0.097  -0.136
## 64    4.892  4.892   1.000   4.892
## 65    0.005  0.005   0.005   0.005
## 66    0.246  0.246   1.000   0.246
## 67    2.405  2.193   0.867   0.867
## 68    3.737  3.551   1.372   1.372
## 69    1.668  1.529   0.701   0.701
## 70    1.718  1.589   0.737   0.737
## 71    4.446  4.396   1.734   1.734
## 72    5.021  4.973   2.060   2.060
## 73    4.235  4.187   1.736   1.736
## 74    5.099  5.099   1.802   5.099
## 75    6.529  6.529   2.952   6.529
## 76    1.560  1.560   3.142   1.560
## 77    0.000  0.000   0.000   0.000
## 78    0.000  0.000   0.000   0.000
```

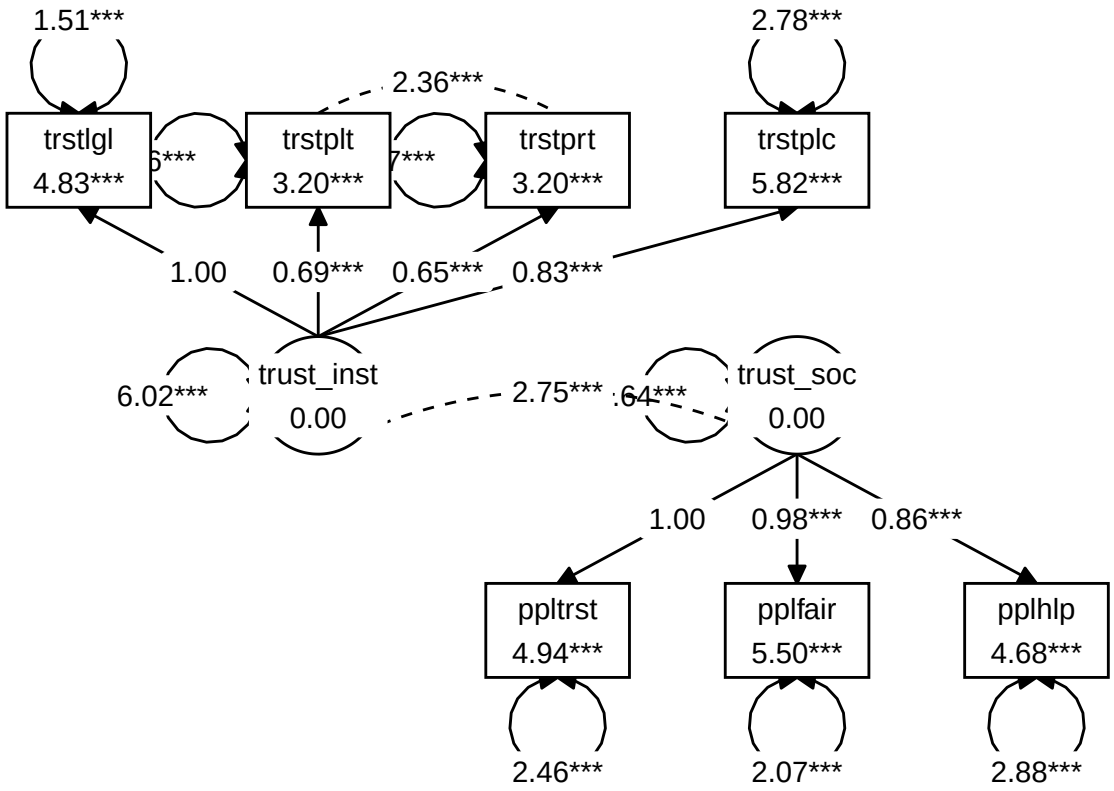
```
# write.table(parameters_mimic_multigroup,
#             "parameters_mimic_std_mult.csv",
#             sep = ",")
```

Part 5: Plot the models

Modified measurement model

```
lay_measurement = get_layout("trstlgl", "trstplt", "trstprt", "trstplc", "",
                             "", "trust_inst", "", "trust_soc", "",
                             "", "", "ppltrst", "pplfair", "pplhlp",
                             "", "", "", "", "",
                             rows = 4)

(measurement_plot = graph_sem(fit_measurement_mod,
                              layout = lay_measurement,
                              angle = 170)
)
```



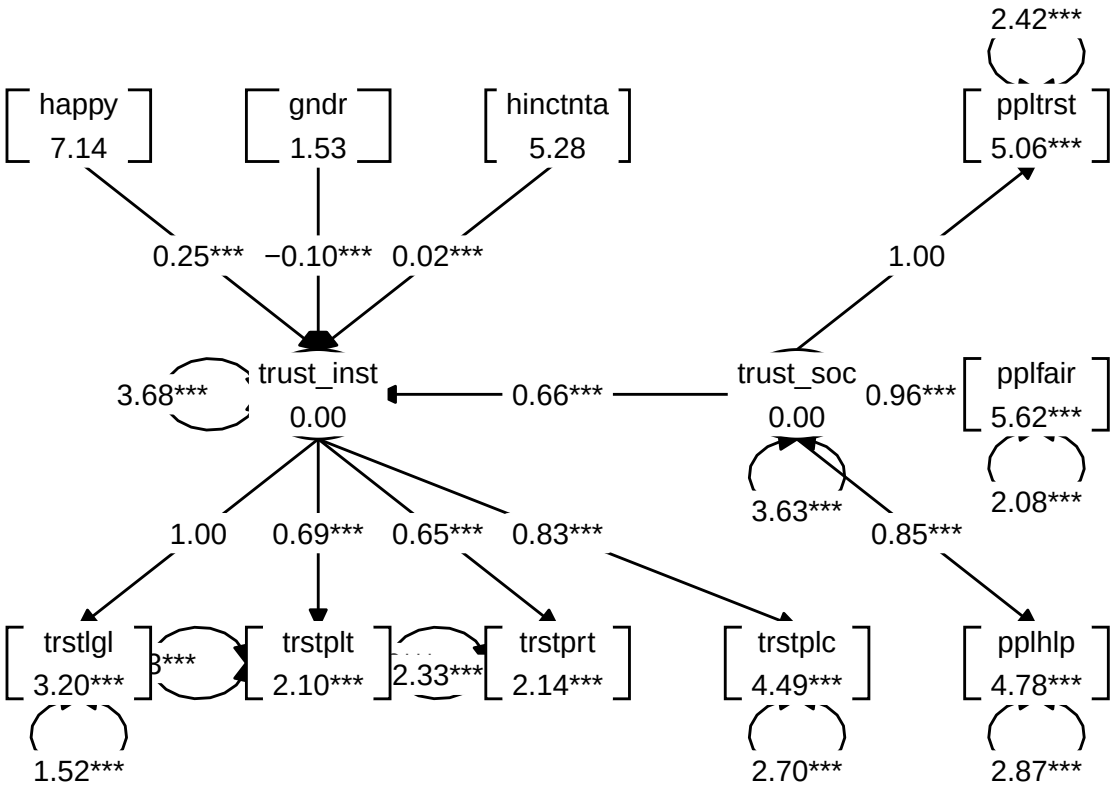

```
# Save the plot
#library(Cairo)
#Cairo(600, 600, file="measurement_plot.png", type="png", bg="white")
#graph_sem(fit_measurement_mod,
#          layout = lay_measurement,
#          angle = 170,
#        spacing_y = 3,
#          text_size = 4,
#          ellipses_width = 1,
#          ellipses_height = 1)
#dev.off()
```

MIMIC model

```
lay <- get_layout("happy", "gndr", "hinctnta", "", "ppltrst",
                  "", "trust_inst", "", "trust_soc",  "pplfair",
                  "trstlgl", "trstplt", "trstprt", "trstplc", "pplhlp",
                  rows = 3)


(mimic_plot = graph_sem(mimic_fit,
          layout = lay,
          angle = 170,
          spacing_y = 3,
```

```
          text_size = 4,
          ellipses_width = 1,
          ellipses_height = 1))
```




```
# Save the plot
# library(Cairo)
#Cairo(600, 600, file="mimic_plot.png", type="png", bg="white")
#graph_sem(mimic_fit,
#          layout = lay,
#          angle = 170,
#          spacing_y = 3,
#          text_size = 4,
#          ellipses_width = 1,
#          ellipses_height = 1.5)
#dev.off()
```