Analysis of The Political Democracy Example and A Fine Dexterity Model: A Structural Equation Modelling Approach

Shahryar Ebrahimi[1]

[1]Department of Psychology, McGill University

Introduction

The ability to coordinate arm, hand, and finger movements in order to manipulate objects is referred as dexterity (Heus et al., 1995). It involves both neural and musculoskeletal functions simultaneously to produce small and precise movements. Dexterity can be classified into gross dexterity and fine dexterity based on movement accuracy (Ives, 2013). Fine dexterity is defined as precise motions associated with controlling small objects using arm, hand, and fingers with eye-hand coordination (Gentier et al., 2013).

The model in Figure 1 is proposed to predict fine dexterity. (Prasetyo, 2020) showed that fine dexterity could be measured by several factors. Hand grip strength is widely known as an indication of performing fine dexterity. As an evidence, based on muscle activation, intrinsic hand muscles have insertions into the proximal phalanx of individual fingers and reported to have a major role in fine motor control, especially the activation of abductor pollicis brevis (APB) (Li et al., 2015). Hand skin temperature is another important factor which influences fine dexterity (Brajkovic & Ducharme, 2003). It is well known that cooling decreases fine dexterity and there is also a significant relationship between the loss of fine dexterity and decrease in finger skin temperature.

Movement properties, such as dexterity, can be influenced directly by arm length, room temperature, and humidity, and indirectly by age and gender that can affect the grip strength therefore, it is valuable to investigate all the affecting factors in one model.

Structural Equation Modeling (SEM) is multivariate statistical models that allow complex modeling of relational structures between independent and dependent variables (Boslaugh, 2007). SEM also allows the estimation of all effects in a model whereas regression approaches often require the specification of numerous regression model to estimate different parts of an overall model (Fabrigar et al., 2010). Therefore, by combining all possible influencing factors into one model, SEM can be applied to predict fine dexterity.

In the present study I aimed to investigate the factors affecting fine dexterity. Gender, age, hand length, grip strength, hand skin temperature, room temperature, and room humidity served as factors to predict fine dexterity by using SEM. To conduct SEM, I used a self-written R script (See Methods) instead of lavaan package. The performance of the script was evaluated before implementing it on the dexterity model.

Method

A new self-written R function called "semCustom" was used to investigate the dexterity model. This function tries to replicate SEM package in R by estimating the model parameters. It uses a general structural equation modelling approach and consists of a measurement model that specifies the relation of observed to latent variables and a latent variable model that shows the influence of latent variables on each other. As described in (Bollen, 1989), the first component of the structural equations is the latent variable model:

$$\eta = \mathbf{B}\eta + \mathbf{\Gamma}\xi + \zeta \qquad \text{Eq.1}$$

In Eq.1, η, the vector of latent endogenous random variables, is $m \times 1$; ξ, the latent exogenous random variables, is $n \times 1$; **B** is the $m \times m$ coefficient matrix showing the influence of the latent endogenous variables on each other; $\mathbf{\Gamma}$ is the $m \times n$ coefficient matrix for the effects of ξ on η. ζ is the disturbance vector that is assumed ($\mathrm{E}(\zeta) = 0$), and which is uncorrelated with ξ. The second component of the general system is the measurement model:

$$y = \Lambda_y \eta + \epsilon \qquad \text{Eq.2}$$

$$x = \Lambda_x \xi + \delta \qquad \text{Eq.3}$$

The $y_{p\times 1}$ and $\mathrm{x}_{q\times 1}$ vectors are observed variables, Λ_y and Λ_x are the coefficient matrices that show the relation of y to η, and x to ξ respectively, and $\epsilon_{p\times 1}$ and $\delta_{q\times 1}$ are the errors of measurement for y and x, respectively. For this system of equations that defines the general structural equation model, a covariance matrix can be defined as:

$$\Sigma(\theta) = \begin{bmatrix} \Sigma_{yy}(\theta) & \Sigma_{yx}(\theta) \\ \Sigma_{xy}(\theta) & \Sigma_{xx}(\theta) \end{bmatrix}$$
$$= \begin{bmatrix} \Lambda_y(\mathrm{I}-\mathrm{B})^{-1}(\Gamma\Phi\Gamma' + \Psi)[(\mathrm{I}-\mathrm{B})^{-1}]'\Lambda_y{}' + \Theta_\epsilon & \Lambda_y(\mathrm{I}-\mathrm{B})^{-1}\Gamma\Phi\Lambda_x{}' \\ \Lambda_x\Phi\Gamma'[(\mathrm{I}-\mathrm{B})^{-1}]'\Lambda_y{}' & \Lambda_x\Phi\Lambda_x' + \Theta_\delta \end{bmatrix} \qquad \text{Eq.5}$$

In Eq.5, Σ_{yy} is the covariance matrix of the observed y variables, and $\Sigma_{yy}(\theta)$ contains the covariances of y written as a function of the unknown model parameters that are stacked in the vector θ. The covariance matrix of y with x is $\Sigma_{yx}(\theta)$ as a function of the structural parameters, and $\Sigma_{xx}(\theta)$ is the covariance matrix of the observed x variables. $\Sigma_{xy}(\theta)$ is the transpose of $\Sigma_{yx}(\theta)$.

The estimation of each element in θ would lead into a complete identification of the model. Although it is possible to identify the model algebraically by equating each element of $\Sigma(\theta)$ with the sample covariance matrix, S, from observed variables y and x, it is not always the best approach, especially with complex models. A numerical solution as used in the present study is also available in which a cost function is defined and minimized using an optimizer. In this method, each parameter of θ is estimated such that the difference between $\Sigma(\theta)$ and S is minimal. In this study, a nonlinear optimizer "nlminb" was used to find the parameters in θ. Three different approaches can be used to define the cost function such that it results in $\Sigma(\theta) \approx \mathrm{S}$.

The most widely known approach is Maximum Likelihood (ML) in which the parameter values are found such that they maximize the likelihood that the process described by the model produced the data that were actually observed. The associated cost function with this approach is defined in Eq.6:

$$F_{ML}(\theta) = \ln(|\Sigma(\theta)|) - ln(|\mathrm{S}|) + tr\{\mathrm{S}\Sigma(\theta)^{-1}\} - p \qquad \text{Eq.6}$$

The other two methods are Unweighted Least Square (ULS) and General Least Square (GLS). The cost functions associated with these methods are defined in Eq.7 and Eq.8, respectively:

$$F_{ULS}(\theta) = \frac{1}{2}\|\mathrm{S} - \Sigma(\theta)\| \qquad \text{Eq.7}$$

$$F_{GLS}(\theta) = \frac{1}{2}\|\big(\mathrm{S} - \Sigma(\theta)\big)\mathrm{W}^{-1}\| \qquad \text{Eq.8}$$

In Eq.8, W is a random weight matrix but is typically chosen to be W = cΣ, where c is a constant and c = 1. As can be seen from Eq.7 and Eq.8, ULS is a special case of GLS where $\mathrm{W} = \mathrm{I}$.

The semCustom can estimate parameters of a given model using ML, ULS, and GLS estimators. It is also able to calculate the degrees of freedom, test statistics (chi-squared), and fit indices such as Root Mean Square of Approximation (RMSEA), Hoelter's Critical N (HCN), McDonals's Fit Index (MFI), Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Expected Cross Validation Index (ECVI), Relative Noncentrality Index (RNI), Bollen's Incremental Fit Index (BIFI), Bollen's Relative Fit Index (BRFI), Parsimony Normed Fit Index (PNFI), Bentler-Bonnet Normed Fit Index (BBNFI), Tucker-Lewis Index (TLI), and Comparative Fit Index (CFI). The details regarding how to calculate these indices can be found in (Bollen, 1989).

Result

Before applying semCustom function on the dexterity model in Figure 1, I evaluated it with a built-in R example called "Political Democracy". The path diagram for this example is shown in Figure 2. The system of equations based on SEM (see Methods) for this model is as Eq.9 to Eq.11:

$$\begin{bmatrix}\eta_1\\ \eta_2\end{bmatrix} = \begin{bmatrix}0 & 0\\ \beta_1 & 0\end{bmatrix}\begin{bmatrix}\eta_1\\ \eta_2\end{bmatrix} + \begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix}[\xi_1] + \begin{bmatrix}\zeta_1\\ \zeta_2\end{bmatrix} \quad \text{Eq.9}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \lambda_2 & 0 \\ \lambda_3 & 0 \\ \lambda_4 & 0 \\ 0 & 1 \\ 0 & \lambda_6 \\ 0 & \lambda_7 \\ 0 & \lambda_8 \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ \epsilon_7 \\ \epsilon_8 \end{bmatrix} \qquad \text{Eq.10}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \lambda_{10} \\ \lambda_{11} \end{bmatrix} [\xi_1] + \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} \qquad \text{Eq.11}$$

semCustom function forces all the first λ variables to 1 as lavaan does. Its output gives the estimates of all λ variables, variances of each observed variable and covariances specified in the model. The fit indices, degrees of freedom and test statistics are also available. The output of this function is the same as lavaan package in R (for lavaan output, please see Appendix), which verifies the performance of the self-written script. ML was used as the default estimator. However, ULS and GLS are also available and produced accurate results. The output of semCustom using ML estimator can be found in Table 1.

In the next step, I applied semCustom on the dexterity model in Figure 1 to estimate the model parameters. The system of equations for this model is as Eq.12 to Eq.14:

$$[\eta_1] = [\gamma_1][\xi_1] + [\zeta_1] \qquad \text{Eq.12}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} [\eta_1] + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix} \qquad \text{Eq.13}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \lambda_5 \\ \lambda_6 \end{bmatrix} [\xi_1] + \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} \qquad \text{Eq.14}$$

I used predefined parameters to generate a sample covariance matrix for the dexterity model. Using this matrix, and multivariate normal distribution, I generated simulated data for y and x variables with N=100 and expected that semCustom function can estimate these predefined parameters properly. The predefined parameters for this model are as follow:

$$\boldsymbol{\lambda} = [\mathbf{0.102}, \mathbf{0.7623}, \mathbf{-0.0923}, \mathbf{-0.382}, \mathbf{0.242}, \mathbf{0.5092}]$$

$$\boldsymbol{\sigma^2} = [\mathbf{3.12}, \mathbf{1.12}, \mathbf{1.1128}, \mathbf{3.0201}, \mathbf{6.312}, \mathbf{0.2525}]$$

$$\mathbf{Cov} = [\mathbf{0.8349}, \mathbf{1.812}, \mathbf{1.023}]$$

B = 0, Γ <- 0.2

The semCustom function was able to estimate each of these parameters correctly. The output of this function is listed in Table 2. In order to show that semCustom estimates λ values correctly, the step in which semCustom forces first loadings to 1 is removed from this part. However, adding this step, gives the same output of lavaan (see Appendix).

Discussion

In this study, first, I evaluated the self-written semCustom function which replicates lavaan package in R. It was applied to Political Democracy Example and was shown that its output is aligned with lavaan and can estimate the parameters of the model correctly. Then, simulated data was generated using the Dexterity model in Figure 1 with predefined parameters and associated sample covariance matrix. Then the data was feed to semCustom, which estimated all of them but two of the covariances correctly. However, lavaan was not able to estimate them either. The reason of this is still unknown to me.

In order to calculate the standard errors, I needed to calculate Fisher Information matrix. Calculating expected information matrix for the models introduced in this project seemed to be complicated. Thus, for this version of semCustom standard errors are not included.

Figures

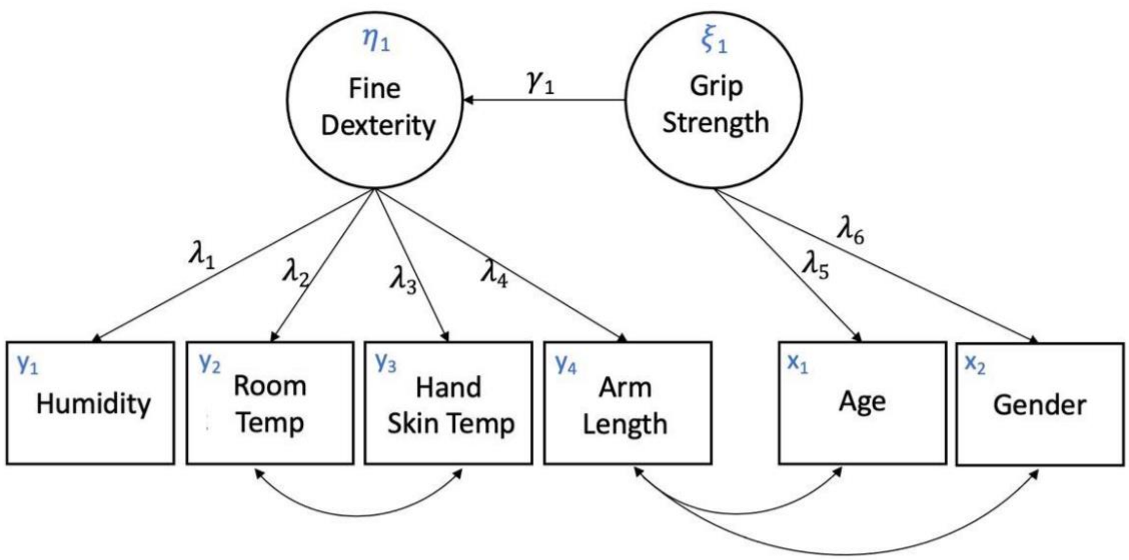

Figure 1. The Fine Dexterity Path Diagram

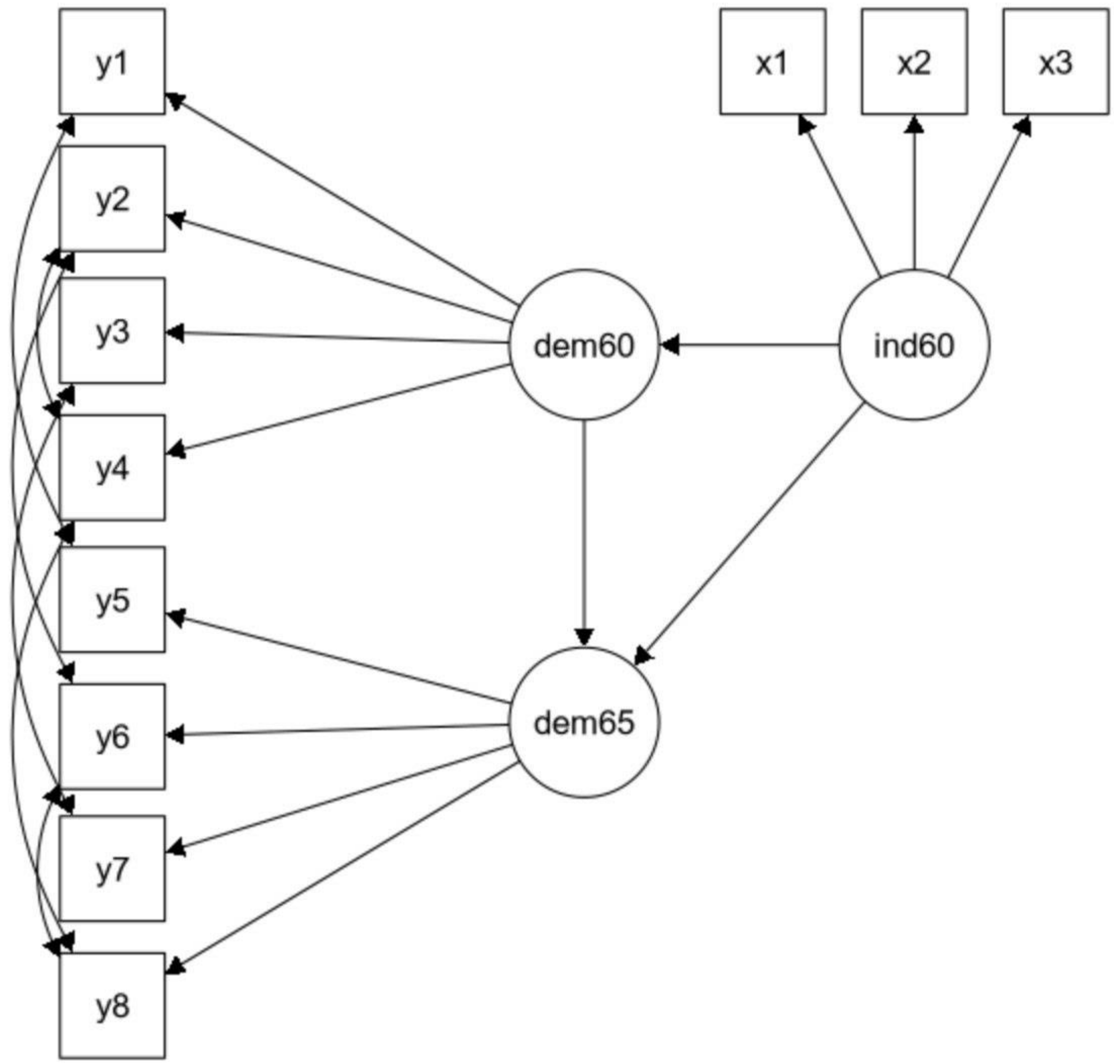

Figure 2. The path diagram for Political Democracy Example

Tables

Table 1. Estimated parameters of Political Democracy Example using semCustom function

loading 1 = 1 loading 2 = 1.257 loading 3 = 1.058 loading 4 = 1.265 loading 5 = 1 loading 6 = 1.186 loading 7 = 1.280 loading 8 = 1.266 loading 9 = 1 loading10 = 2.18 loading11 = 1.82	Beta 1 = 4.01 Gamma 1 = 0.50 Gamma 2 = 0.92	Var 1 = 1.89 Var 2 = 7.37 Var 3 = 5.07 Var 4 = 3.15 Var 5 = 2.35 Var 6 = 4.95 Var 7 = 3.43 Var 8 = 3.25 Var 9 = 0.08 Var 10 = 0.12 Var 11 = 0.47	Cov 1 = 0.62 Cov 2 = 1.31 Cov 3 = 2.15 Cov 4 = 0.79 Cov 5 = 0.35 Cov 6 = 1.36	Model Chi-Sq = 38.13 Null Chi-Sq = 730.65	Base DF = 66 Null DF = 55 Model DF = 35
Fit Indices					
SSABIC = 3159.7 BIC = 3333 AIC = 3205.6 MFI = 0.979	RMSEA = 0.0345 RNI = 0.9953 BIFI = 0.9955 BRFI = 0.9180	PNFI = 0.603 BBNFI = 0.947 TLI = 0.993	PNFI = 0.603 BBNFI = 0.948 TLI = 0.992	CFI = 0.9953 ECVI = 1.335	HCN01 = 113.803 HCN05 = 98.97

Table 2. Estimated parameters of dexterity model using semCustom function

loading1 = 0.101 loading2 = 0.758 loading3 = -0.091 loading4 = -0.380 loading5 = 0.240 loading6 = 0.507	Beta 1 = 4.01 Gamma 1 = 0.20	Var1 = 3.09 Var2 = 1.11 Var3 = 1.10 Var4 = 2.99 Var5 = 6.245 Var6 = 0.250	Cov1 = 0.826 Cov2 = 0.000 Cov3 = 0.000	Model Chi-Sq = 0.00 Null Chi-Sq = 40.413	Base DF = 21 Null DF = 15 Model DF = 5
Fit Indices					
SSABIC = 2091.7 BIC = 2139.07 AIC = 2099.99 MFI = 1.025	RMSEA = NaN RNI = 1.197 BIFI = 1.141 BRFI = 1.00	PNFI = 0.333 BBNFI = 1 TLI = 1.59	PNFI = 0.60 BBNFI = 0.95 TLI = 0.992	CFI = 1.197 ECVI = 0.32	HCN05 = 1.25e13 HCN01 = 1.71e13

Appendix

Lavaan output for Political Democracy Example is as follow:

```
lavaan 0.6-12 ended normally after 68 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        31

  Number of observations                            75

Model Test User Model:

  Test statistic                                38.125
  Degrees of freedom                                35
  P-value (Chi-square)                           0.329

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  dem60 =~
    y1                1.000                               2.223    0.850
    y2                1.257    0.182    6.889    0.000    2.794    0.717
    y3                1.058    0.151    6.987    0.000    2.351    0.722
    y4                1.265    0.145    8.722    0.000    2.812    0.846
  dem65 =~
    y5                1.000                               2.103    0.808
    y6                1.186    0.169    7.024    0.000    2.493    0.746
    y7                1.280    0.160    8.002    0.000    2.691    0.824
    y8                1.266    0.158    8.007    0.000    2.662    0.828
```

```
  ind60 =~
    x1                1.000                               0.670    0.920
    x2                2.180    0.139   15.742    0.000    1.460    0.973
    x3                1.819    0.152   11.967    0.000    1.218    0.872

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  dem60 ~
    ind60             1.483    0.399    3.715    0.000    0.447    0.447
  dem65 ~
    ind60             0.572    0.221    2.586    0.010    0.182    0.182
    dem60             0.837    0.098    8.514    0.000    0.885    0.885

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .y1 ~~
   .y5                0.624    0.358    1.741    0.082    0.624    0.296
 .y2 ~~
   .y4                1.313    0.702    1.871    0.061    1.313    0.273
   .y6                2.153    0.734    2.934    0.003    2.153    0.356
 .y3 ~~
   .y7                0.795    0.608    1.308    0.191    0.795    0.191
 .y4 ~~
   .y8                0.348    0.442    0.787    0.431    0.348    0.109
 .y6 ~~
   .y8                1.356    0.568    2.386    0.017    1.356    0.338
```

1

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .y1                1.891    0.444    4.256    0.000    1.891    0.277
   .y2                7.373    1.374    5.366    0.000    7.373    0.486
   .y3                5.067    0.952    5.324    0.000    5.067    0.478
   .y4                3.148    0.739    4.261    0.000    3.148    0.285
   .y5                2.351    0.480    4.895    0.000    2.351    0.347
   .y6                4.954    0.914    5.419    0.000    4.954    0.443
   .y7                3.431    0.713    4.814    0.000    3.431    0.322
   .y8                3.254    0.695    4.685    0.000    3.254    0.315
   .x1                0.082    0.019    4.184    0.000    0.082    0.154
   .x2                0.120    0.070    1.718    0.086    0.120    0.053
   .x3                0.467    0.090    5.177    0.000    0.467    0.239
   .dem60             3.956    0.921    4.295    0.000    0.800    0.800
   .dem65             0.172    0.215    0.803    0.422    0.039    0.039
    ind60             0.448    0.087    5.173    0.000    1.000    1.000
```

2

Lavaan output for the Dexterity model is as follow:

```
lavaan 0.6-12 ended normally after 237 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        16

  Number of observations                           100

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 5
  P-value (Chi-square)                           1.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  FD =~
    y1                1.000                               0.103    0.059
    y2                7.474   49.701    0.150    0.880    0.773    0.592
    y3               -0.905    7.140   -0.127    0.899   -0.094   -0.089
    y4               -3.745    8.529   -0.439    0.661   -0.388   -0.219
  GS =~
    x1                1.000                               0.241    0.096
    x2                2.104   14.507    0.145    0.885    0.507    0.712
```

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  FD ~
    GS                0.084    0.577    0.146    0.884    0.196    0.196

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .y2 ~~
   .y3                0.827    0.564    1.466    0.143    0.827    0.748
 .y4 ~~
   .x1                0.000    0.458    0.000    1.000    0.000    0.000
   .x2                0.000    0.281    0.000    1.000    0.000    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .y1                3.089    0.442    6.981    0.000    3.089    0.997
   .y2                1.109    3.961    0.280    0.780    1.109    0.650
   .y3                1.102    0.181    6.082    0.000    1.102    0.992
   .y4                2.990    1.075    2.781    0.005    2.990    0.952
   .x1                6.249    0.970    6.443    0.000    6.249    0.991
   .x2                0.250    1.769    0.141    0.888    0.250    0.493
   .FD                0.010    0.074    0.139    0.889    0.962    0.962
    GS                0.058    0.417    0.139    0.890    1.000    1.000
```

If the first loadings in the Dexterity model is forced to 1 in semCustom, the same λ values as lavaan are produced as follow:

loading 1 = 1.000000
loading 2 = 7.473532
loading 3 = -0.904902
loading 4 = -3.745098
loading 5 = 1.000000
loading 6 = 2.104132

Acknowledgement

I want to thank Jeremy Rappel for his contribution to semCustom function and this project.

References

Bollen, K. A. (1989). *Structural equations with latent variables* (Vol. 210). John Wiley & Sons.

Boslaugh, S. (2007). *Encyclopedia of epidemiology*. Sage Publications.

Brajkovic, D., & Ducharme, M. B. (2003). Finger dexterity, skin temperature, and blood flow during auxiliary heating in the cold. *Journal of Applied Physiology*, *95*(2), 758-770.

Fabrigar, L. R., Porter, R. D., & Norris, M. E. (2010). Some things you should know about structural equation modeling but never thought to ask. *Journal of Consumer Psychology*, *20*(2), 221-225.

Gentier, I., D'Hondt, E., Shultz, S., Deforche, B., Augustijn, M., Hoorne, S., Verlaecke, K., De Bourdeaudhuij, I., & Lenoir, M. (2013). Fine and gross motor skills differ between healthy-weight and obese children. *Research in Developmental Disabilities*, *34*(11), 4043-4051. https://doi.org/10.1016/j.ridd.2013.08.040

Heus, R., Daanen, H. A. M., & Havenith, G. (1995). Physiological Criteria for Functioning of Hands in the Cold - a Review. *Applied Ergonomics*, *26*(1), 5-13. https://doi.org/Doi 10.1016/0003-6870(94)00004-I

Ives, J. C. (2013). *Motor behavior: Connecting mind and body for optimal performance*. Lippincott Williams & Wilkins.

Li, T., Hua, X.-Y., Zheng, M.-X., Wang, W.-W., Xu, J.-G., Gu, Y.-D., & Xu, W.-D. (2015). Different cerebral plasticity of intrinsic and extrinsic hand muscles after peripheral neurotization in a patient with brachial plexus injury: A TMS and fMRI study. *Neuroscience Letters*, *604*, 140-144.

Prasetyo, Y. T. (2020). Factors Affecting Fine Dexterity: A Structural Equation Modeling Approach. Proceedings of the 2020 2nd International Conference on Management Science and Industrial Engineering,