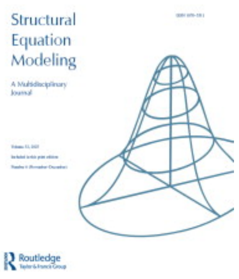
lavaangui: a web-based graphical interface for specifying lavaan models by drawing path diagrams

Karch, J.D.



Structural Equation Modeling: A Multidisciplinary Journal

ISSN: 1070-5511 (Print) 1532-8007 (Online) Journal homepage: www.tandfonline.com/journals/hsem20

lavaangui: A Web-Based Graphical Interface for Specifying Lavaan Models by Drawing Path Diagrams

J. D. Karch

To cite this article: J. D. Karch (2025) lavaangui: A Web-Based Graphical Interface for Specifying Lavaan Models by Drawing Path Diagrams, Structural Equation Modeling: A Multidisciplinary Journal, 32:6, 1077-1088, DOI: 10.1080/10705511.2024.2420678

To link to this article: https://doi.org/10.1080/10705511.2024.2420678

STRUCTURAL EQUATION MODELING: A MULTIDISCIPLINARY JOURNAL
2025, VOL. 32, NO. 6, 1077–1088
https://doi.org/10.1080/10705511.2024.2420678

TEACHER'S CORNER

OPEN ACCESS

`lavaangui`: A Web-Based Graphical Interface for Specifying Lavaan Models by Drawing Path Diagrams

J. D. Karch

Department of Methodology and Statistics, Institute of Psychology, Leiden University, Leiden, The Netherlands

ABSTRACT
Path diagrams facilitate the specification of Structural Equation Models because drawing them is often faster and less error-prone than specifying a model using equations or matrix algebra. Despite this, there are very few free and open-source programs that allow model specification through path diagrams. To address this gap, this paper introduces the web application `lavaangui`, which is available at https://lavaangui.org. `lavaangui` is a graphical user interface that allows specifying `lavaan` models by drawing path diagrams. Additionally, it can be installed as an R package and then also supports creating interactive path diagrams from `lavaan` models specified in R. This paper presents a tutorial on using `lavaangui`.

KEYWORDS
Graphical user interface; lavaan; path diagram; structural equation modeling; web application

1. Introduction

Path diagrams, the graphical representations of structural equation models (SEM), have significantly contributed to the popularity of SEM. They offer a clear and intuitive presentation of the model, detailing the expected relationships among all variables. Path diagrams facilitate model specification because drawing a path diagram is often faster and less error-prone than specifying a model using equations or matrix algebra. Additionally, path diagrams enhance communication, as they are much easier to comprehend than the corresponding matrix equations or result tables.

SEM analysis has traditionally been performed using closed-source commercial packages such as Mplus (Muthén & Muthén, 1998–2017), LISREL (Jöreskog & Sörbom, 1996), AMOS (Arbuckle, 2019), EQS (Bentler, 2006), SAS PROC CALIS (Yung, 2010), and Stata (StataCorp LLC, 2023). Originally, most of these programs supported only a syntax mode. For model specification, users needed to translate their path diagrams into software-specific syntax. For the first widely used SEM program, LISREL, users had to translate their diagrams into a set of model matrices. As noted in Rosseel (2012), this was an unpleasant experience for many first-time users. To remedy this, later SEM programs, such as EQS and Mplus, aimed to offer simpler model specification syntaxes that did not rely on matrix algebra. However, while easier, users still had to learn software-specific model syntaxes. To address this issue, current versions of most commercial software packages such as AMOS, Stata, Mplus, EQS, and LISREL support model specification through the drawing of path diagrams, complementary to the syntax mode, thereby simplifying the process. These graphical user interfaces are typically called diagrammers.

The SEM community is increasingly transitioning from commercial and closed-source software to free and open-source (FOSS) alternatives. This shift is primarily motivated by the desire for transparency, reproducibility, and accessibility. Consequently, a significant number of FOSS SEM packages have been and are still being developed. This includes the R packages `sem` (Fox et al., 2022), `lavaan` (Rosseel, 2012), `OpenMx` (Neale et al., 2016), and `tidySEM` (van Lissa, 2023); `semopy` (Igolkina & Meshcheryakov, 2020) for Python; `StructuralEquationModels.jl` (Ernst & Peikert, 2024) for Julia; the standalone software Ωnyx (von Oertzen et al., 2015); and the SEM modules within JASP (JASP Team, 2024) and Jamovi (The jamovi project, 2024). Crucially, the only FOSS software that includes a diagrammer, which allows model specification via drawing path diagrams, is Ωnyx. Thus, compared to commercial software, the availability of diagrammers in FOSS software is scarce.

Of the FOSS packages, `lavaan` is arguably the most popular[1] and thus constitutes a good candidate for an accompanying diagrammer. In fact, multiple graphical user interfaces for `lavaan` have already been developed with the goal of making it more accessible. The standalone applications JASP and Jamovi both include a graphical interface

CONTACT J.D. Karch j.d.karch@fsw.leidenuniv.nl Department of Methodology and Statistics, Institute of Psychology, Leiden University, Leiden, The Netherlands.

[1]Among the three established R packages (OpenMx, sem, lavaan), lavaan is downloaded most often (lavaan with 51k downloads, OpenMx with 18k downloads, sem with 25k downloads last month, according to https://cranlogs.r-pkg.org/ as of April 4th, 2024). Furthermore, the SEM modules in both JASP and Jamovi rely on lavaan for model estimation.

for lavaan. Additionally, multiple web applications for lavaan have been developed, such as lavaanGUI (Mayer, 2017), shinylavaan (Recka, 2024), shiny.lavaan (Hamilton, 2017), and ezsem (Iguchi, 2018). However, none of the existing interfaces includes a diagrammer.[2] Thus, none of the interfaces allow specifying the model through drawing a path diagram.

To fill this gap, this paper introduces the graphical user interface lavaangui for lavaan, which crucially includes a diagrammer and thus allows for model specification through drawing path diagrams. lavaangui is a web application accessible via http://lavaangui.org. It is designed as a web application to maximize accessibility, requiring no installation and capable of running on any computer with a browser. The main features of lavaangui are

1. Easy creation and modification of path diagrams through an interactive graphical user interface.
2. Auto-completion of path diagrams according to changeable rules.
3. Estimates are directly displayed in the path diagram.
4. Multiple features for obtaining visually pleasing path diagrams, such as automatic layout algorithms but also fined-grained customization via mouse gestures.
5. Support for most estimators available in lavaan.
6. Most additional results, such as modification indices and model residuals, are available.
7. The model, including the data and estimation results, can be saved locally to enable reproducibility.
8. The path diagram can be exported to most common file formats and can be readily used in publications.

lavaangui is also available as an R package. This allows running lavaangui locally, offering a solution when the web server is unreachable, for example, due to a lack of internet access. Another benefit is that computations are likely to be faster, as most personal computers possess more computing power than the web server allocates to one user. Additionally, the R package facilitates importing lavaan models. This enables the modification of lavaan models via lavaangui.

The lavaangui R package additionally provides utility to users who prefer to keep specifying their models in lavaan because it can generate path diagrams for lavaan models, which is commonly known as plotting an SEM.[3] lavaangui is thus an alternative to the existing plotting packages for lavaan: semPlot (Epskamp, 2015), lavaanPlot (Lishinski, 2024), and tidySEM (van Lissa, 2023). The key benefit is that the path diagrams lavaangui produces are interactive. That is, after they have been created, they can be easily customized via the graphical user interface, for example, by dragging variables and arrows into desired positions with the mouse. Figure 1 shows a path diagram generated by lavaangui for an example model, both without and with exemplary customization. The disadvantages of lavaangui compared to other plotting packages are that it currently does not offer as many customization options as some of the existing packages, and the steps for customizing the path diagram cannot be saved as R code.

lavaangui can also be used to only draw path diagrams without fitting a model. In this capacity, it is an alternative to semdiag (Mai et al., 2023), the only other web application for drawing path diagrams currently available. lavaangui offers several features not available in semdiag, including zooming, automatic placement of arrows to avoid overlap, automatic layout algorithms, dynamic guidelines for layout assistance, undo/redo functionality, and the ability to snap nodes to a grid. Conversely, semdiag uniquely supports the use of mathematical symbols, copying shapes, annotating the model with text fields, and offers more flexible placement of path labels.

The remainder of this paper contains a tutorial on lavaangui, specifically version 0.2.0. First, I briefly describe the basics of path diagrams and the specific style of path diagrams adopted by lavaangui. Second, I provide an overview of how to use the main features of the diagrammer based on an example analysis. Third, I explain how to use lavaangui for plotting lavaan models and drawing path diagrams without fitting a model. Fourth, I provide a short overview of the remaining features. Finally, I discuss current limitations, offer recommendations on when to use lavaangui versus lavaan, and outline plans.

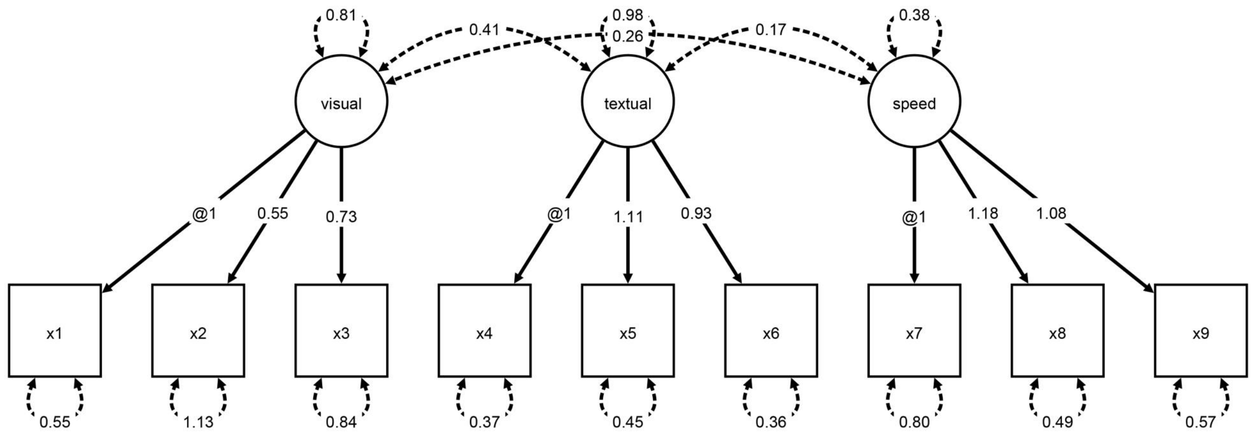
2. Path Diagrams in lavaangui

In addition to the textual description of path diagrams, as used in lavaangui, Figure 2 displays all components. In path diagrams, variables are depicted as nodes, and the relationships between these variables are represented by arrows. Variables can be of three types: observed, latent, and constant. Rectangles represent observed variables, ellipses latent variables, and triangles constant variables. Observed variables are those that are directly measured and are typically contained in your data set. Latent variables are unobserved and can only be inferred indirectly from observed variables. The constant variable is a special variable with a value of 1, used to incorporate means and intercepts into the model. Observed and latent variables both have labels, such as, “x1”, and “f1”. In lavaangui, beyond giving variables a name, the labels of observed variables link the model to the data set. For example, if an observed variable in the model is named “x1”, lavaangui will associate this with the variable “x1” in the data set when fitting the model.

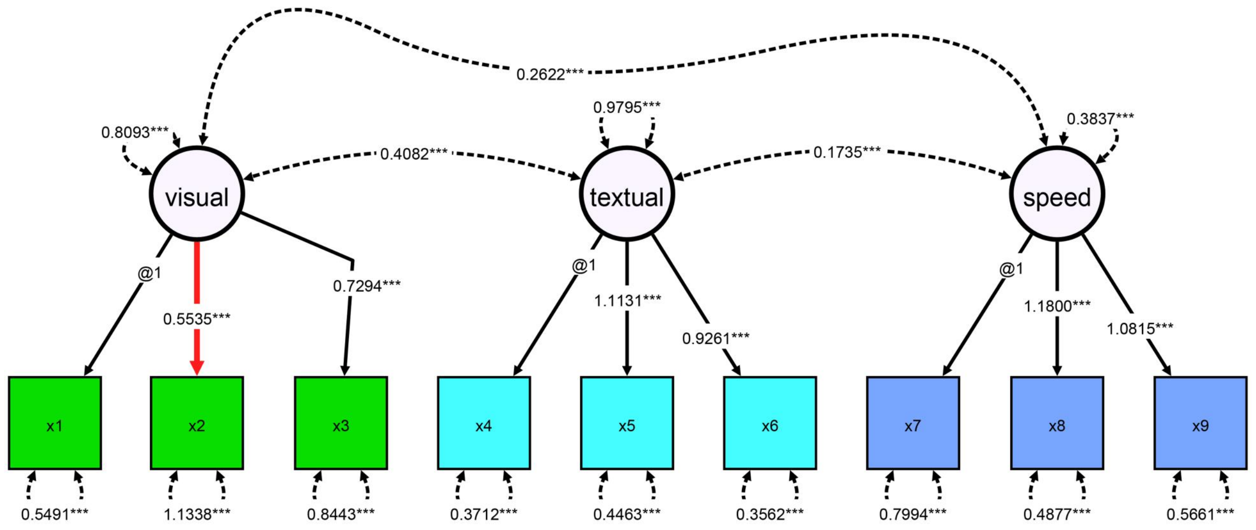
Two types of arrows are used to represent relationships between variables: directed and undirected arrows. Directed arrows are drawn as single-headed, while undirected ones are drawn as double-headed. The exact meaning of directed and undirected arrows can vary depending on the path

[2]The Ωnyx diagrammer allows exporting path diagrams to lavaan syntax. However, crucially, the model cannot be fitted in lavaan directly from the diagrammer.

[3]The first program for plotting an SEM was LISPATH (Marcoulides & Papadopoulos, 1993).



(a) Without customization.



(b) With customizations. The customizations were chosen with the goal of demonstrating most customization options of **lavaangui**; not a visually appealing result.

Figure 1. Path diagrams of a three-factor model as generated by `lavaangui`.

diagram style, particularly with regard to how residual variances are represented (Epskamp, 2015). `lavaangui` primarily follows the so-called "RAM" style (Boker et al., 2002). Specifically, directed arrows, or one-headed arrows, indicate a linear regression relationship where one variable is predicted by another. In contrast, undirected arrows, or two-headed arrows, represent (residual) covariances between variables, indicating a mutual association without implying a predictive direction. If an undirected arrow connects two exogenous variables, that is, those without any directed arrows pointing to them, it represents a covariance. If either of the two variables is endogenous, that is, has directed arrows pointing to it, the undirected arrow represents a residual covariance, which is the covariance between the unexplained portions of the two observed variables after accounting for the effects of the regression equations, as represented by the incoming directed arrows. A special kind of undirected arrow is the one that connects a variable to itself. For exogenous variables, as per the introduced conventions, this represents the covariance of the variable with itself, the variance. For endogenous variables, it represents the residual variance. One point where `lavaangui` deviates from the "RAM" style is that it allows observed exogenous variables without (residual) variance arrows. These represent fixed variables and the means, variances, and covariances of these variables are fixed to their sample values.

Arrows can have labels, such as, "a". Labels serve two purposes: First, to give a relationship a specific name, such that its estimated value can be easily found in result tables, and second, to introduce constraints into the model. I will explain this in more detail later.

Relationships can be specified to be fixed or freely estimated. A fixed relationship is assigned a specific value, while for a freely estimated relationship, its value is estimated based on the data. To represent fixed and freely estimated relationships, `lavaangui` deviates from the "RAM" style and instead uses the following conventions. A fixed relationship is represented by @x overlayed on top of the arrow, where x stands for any number. For example, @1 implies that the relationship, that is, regression coefficient or covariance, is fixed to the value 1. In all other cases, the relationship is considered free.

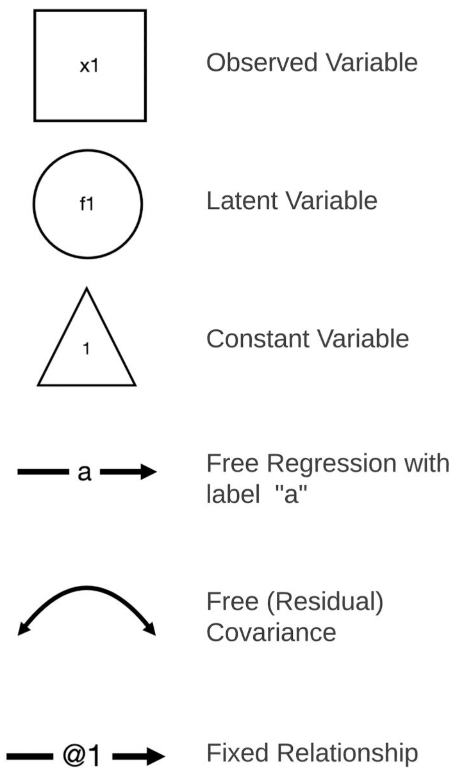


Figure 2. Path diagram components of `lavaangui`.

To ensure that the minimum requirements for a valid `lavaan` model are met, `lavaangui` enforces the following rules:

- Variable names and arrow labels must be valid `lavaan` names; for example, they must start with a letter, not a number.
- There can be at most one arrow between each pair of variables.
- Constant variables can only be the source of directed arrows. Undirected arrows are not applicable to constants.
- Each variable can be connected with at most one constant variable.

Despite these rules, it is still possible to specify invalid models. Therefore, care must be taken to ensure that a valid model is specified, as always.

3. Main Features

To expose the main features of `lavaangui`, I will explain an example analysis using the Holzinger & Swineford data set. The reader is encouraged to follow along with all the described steps in the `lavaangui`. The Holzinger & Swineford data set is a 'classic' data set that has been used in many SEM resources, including the paper introducing lavaan (Rosseel, 2012). The data consists of mental ability test scores of seventh- and eighth-grade children from two different schools. In the version of data provided, which is equivalent to the version included in lavaan, only 9 out of the original 26 tests are included. A CFA model that is often proposed for these 9 variables is displayed in Figure 1.

3.1. Installation & Starting

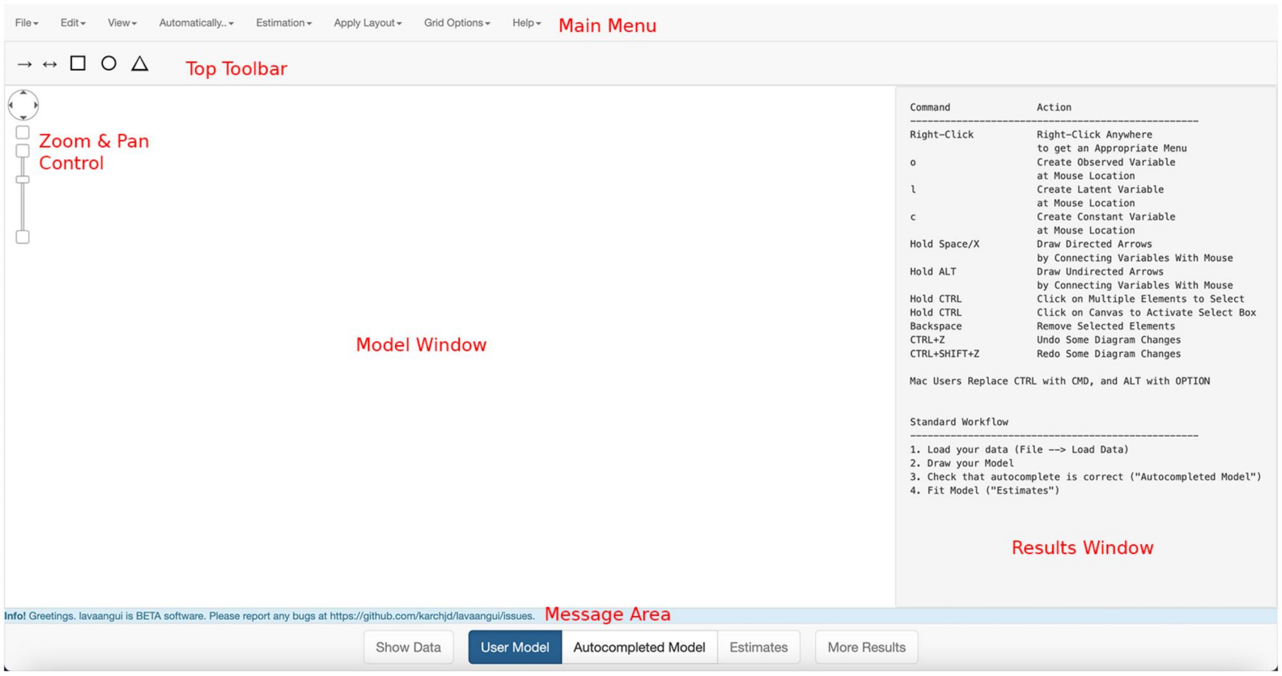
To open lavaangui, visit http://lavaangui.org. You can alternatively run a local version. This is recommended for R users, as it will likely be faster and also enables importing and plotting of `lavaan` models. To do so, first, install the package via the command `install.packages("lavaangui")`. Then, call the function `lavaangui()`. This will open the web application in your browser. You can also import `lavaan` models. In particular, if `fit` is a `lavaan` model, as obtained from the `lavaan`, `sem`, or `cfa` functions, then `lavaangui(fit)` opens the web application with the model and data contained in `fit` imported.

3.2. Overview of `lavaangui` Interface

The `lavaangui` application looks as displayed in Figure 3 (without the red text). The red text in the figure describes the main components. The *Main Menu* is at the very top. Here, most actions not related to drawing the path diagram can be performed. This includes loading and saving data or models and changing settings. Through the *Top Toolbar*, variables (nodes) and relationships (arrows) can be inserted into the path diagram. The *Model Window* displays the path diagram and allows modifications through various mouse gestures, most importantly drag-and-drop and right-clicking. After fitting a model, the *Results Window* shows the main results that cannot be shown directly in the path diagram. The Results Window is also used to display the `lavaan` script corresponding to the path diagram, as well as a command list, which is displayed at startup. In the *Message Area*, messages from `lavaangui`, such as warnings, errors, and success notifications, are displayed. The *Bottom Toolbar* allows switching between the three modes of `lavaangui`. More details on this will follow later.

3.3. Loading the Data

The first step is to load the data. First, download the Holzinger & Swineford ".csv" file using the following link https://zenodo.org/records/14187923/files/cfa.csv?download=1. After, load the data by navigating to the *Main Menu*, clicking on *File* → *Load Data*, and selecting the file you just downloaded. Note that while this is a ".csv" file, the most commonly used file formats can be loaded in the same manner. After loading the data, the *Data Viewer* is automatically shown. The main purpose of the *Data Viewer* is to confirm that the data set was loaded correctly. The second purpose is to rename variables, which can be achieved by double-clicking on a variable name. As a reminder, for model fitting, the variable names in the data set have to correspond to the variable labels of the observed variables in the path diagram, and the variable names in the path diagram need to be valid `lavaan` names. Thus, if variables with invalid names need to be included in the path diagram, they must be renamed first in the *Data Viewer*. For the example analysis, you do not need to rename any variables.



```
Command                Action
-----------------------------------------------------
Right-Click            Right-Click Anywhere
                       to get an Appropriate Menu
o                      Create Observed Variable
                       at Mouse Location
l                      Create Latent Variable
                       at Mouse Location
c                      Create Constant Variable
                       at Mouse Location
Hold Space/X           Draw Directed Arrows
                       by Connecting Variables With Mouse
Hold ALT               Draw Undirected Arrows
                       by Connecting Variables With Mouse
Hold CTRL              Click on Multiple Elements to Select
Hold CTRL              Click on Canvas to Activate Select Box
Backspace              Remove Selected Elements
CTRL+Z                 Undo Some Diagram Changes
CTRL+SHIFT+Z           Redo Some Diagram Changes

Mac Users Replace CTRL with CMD, and ALT with OPTION


Standard Workflow
-----------------------------------------------------
1. Load your data (File --> Load Data)
2. Draw your Model
3. Check that autocomplete is correct ("Autocompleted Model")
4. Fit Model ("Estimates")
```

Info! Greetings. lavaangui is BETA software. Please report any bugs at https://github.com/karchjd/lavaangui/issues.

Figure 3. Overview of the `lavaangui` interface.

3.4. Drawing the Diagram

After closing the *Data Viewer* by pressing the *Esc* key or clicking on the closing symbol in the top right corner, you will notice that more symbols and buttons have been added to the *Top Toolbar*. Among others, there is now a rectangle for each observed variable in the data set. I will now describe how to draw the three-factor model with the help of the *Top Toolbar*. For each factor, I will explain a different, increasingly less general but faster approach. Before proceeding, double-check that in the *Bottom Toolbar*, the *User Model* mode is selected.

3.4.1. Creating Variables

One way to create a latent variable is to drag and drop the circle symbol from the *Main Toolbar* into the *Model Window*. Perform this to create a latent variable. Alternatively, you can right-click on the *Model Window* and select *Add Latent Variable*, or press the *l* key while your mouse hovers in the *Model Window*. The created variable is called “f1”. To rename the latent variable, right-click on it and select *Rename Variable*. Type “visual” in the text field that appears and press the *Rename* button or press the *return* key. To bring the observed variables related to the “visual” latent variable into the model, drag the rectangles “x1”, “x2”, and “x3” from the *Top Toolbar* into the *Model Window*.

3.4.2. Creating Arrows

To add the directed arrows, click on the directed arrow symbol in the *Top Toolbar*. Then, you can draw arrows by connecting variables with your mouse. Undirected arrows can be added in the same manner by clicking on the undirected arrow symbol instead. As an example, create the arrow from “visual” to “x1”, by first clicking on the directed arrow symbol, then clicking on the “visual” latent variable, holding down the mouse button, dragging the arrow that appears to the observed variable “x1”, and then releasing the mouse button. Note that after creating this first arrow, the *Results Window* will display the `lavaan` code corresponding to your path diagram. This display updates automatically while you change your diagram.

A more efficient method to create arrows is using keyboard shortcuts. To draw a directed arrow, hold the *space* or *X* keys while performing the mouse actions described above. For an undirected arrow, hold the *Alt*[4] key. Draw directed arrows from “visual” to “x2” and “x3” using keyboard shortcuts. Your path diagram should resemble Figure 4a, with variables placed differently.

3.4.3. Selecting & Moving Variables

To arrange variables such that they are nicely aligned with each other you can simply drag them around with your mouse. Grey and red helper lines will automatically appear to assist you in aligning the variables. To move the whole model, click and hold somewhere empty in the *Model Window* and move your mouse. To move only certain variables in the diagram, first select those variables and then move them. You can select multiple variables using standard methods. For example, while holding the *Control* or *Shift* keys, each variable you click on will be selected. Alternatively, you can create a selection rectangle by holding down the *Control*[5] key, clicking and holding the mouse button, and then dragging the mouse within the *Model Window*. Clicking on a space in the *Model Window* will deselect all elements.

[4]Mac users, use the *Option* key instead.
[5]For Mac users, the *Command* key also works.

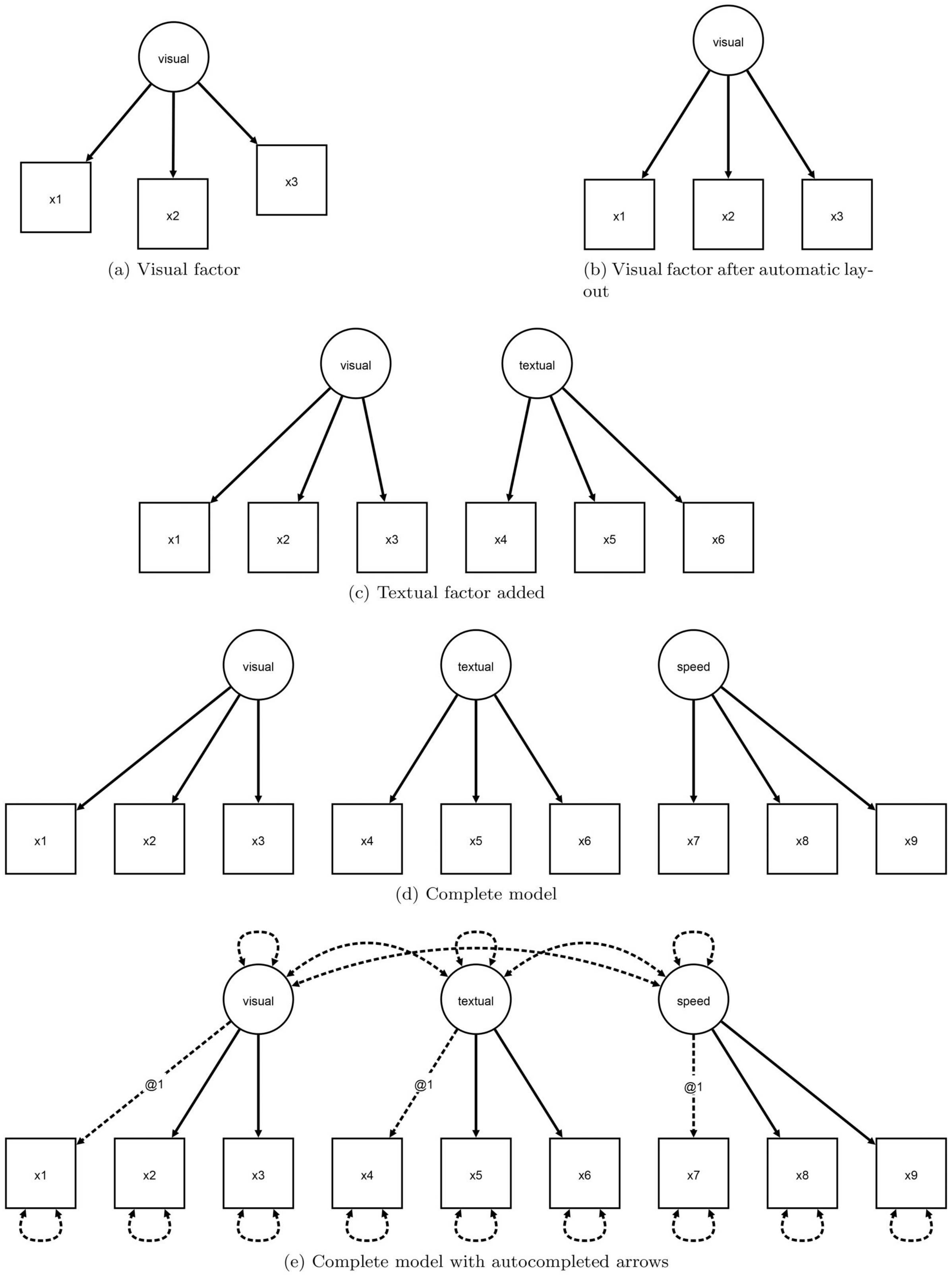


Figure 4. Intermediate steps in the creation of the three-factor model.

3.4.4. Deleting

As in most other programs, the deletion process involves first selecting the elements you want to delete and then deleting the selected elements. To select an element, click on it. To select multiple elements, follow the procedure just described for selecting multiple variables. For deleting selected elements, press the *Backspace* key, which is the key you typically use for deleting.

3.4.5. Automatic Layout

`lavaangui` includes an automatic layout functionality that attempts to arrange the variables in a visually pleasing way automatically. This is available from the *Main Menu* under the *Apply Layout* menu item. Apply the "Recommended: Tree" layout now. Your path diagram should now resemble Figure 4b. lavaangui supports three recommended and five experimental layouts. The recommended layouts are provided by the semPlot R package (Epskamp, 2015) and are based on previous work by Jöreskog and Sörbom (1996), Boker et al. (2002), and Reingold and Tilford (1981). The experimental layouts are provided by the cytoscape graph theory library (Franz et al., 2016). The recommended layout algorithms have specifically been optimized for path diagrams, while the experimental layout algorithms are designed to work well for all kinds of diagrams.

3.4.6. Undo & Redo

After applying a layout, you might want to revert its changes, which would be cumbersome to do manually. To revert all changes made by the layout algorithm, you can instead use the *Undo, Redo* functionality. This also works for most other appearance changes, like moving variables. Other actions, like marking an arrow as free, cannot be undone at the moment because they can be easily reverted otherwise. You can access the *Undo, Redo* functionality via the *Main Menu* item *Edit* or the usual keyboard shortcuts (Control + Z and Control + Shift + Z).[6]

3.4.7. Zooming

You might have noticed that by applying a layout, lavaangui also changed the zoom level. To zoom in and out of the path diagram manually, you can use your mouse in the same manner as you would for scrolling a webpage. For most users, this will be via the mouse wheel or the trackpad of their laptops. Alternatively, you can use the *Zoom & Pan Control* widget at the top left corner of the *Model Window*.

3.4.8. Creating Multiple Variables

To create multiple observed variables at once, you can use the *Multiple Variables* rectangle within the *Top Toolbar*. Drag and drop it at the place where you would like the variable to be created. We will use this functionality to draw the second factor ("textual") faster. First, make sure that there is space for the "textual" factor to the right of the "visual" factor by employing the move or zoom functionalities. Drag and drop the *Multiple Variables* rectangle into the *Model Window*. After you have dropped it, select "x4", "x5", and "x6" in the menu by clicking on those items while holding the *Shift* key. To create the "textual" latent variable and connect it to the just created observed variables, follow the procedure described for the "visual" latent variable. Apply the *Recommended: Tree* layout again. Your path diagram should now resemble Figure 4c.

[6]Mac users can again replace *Control* with *Command*.

3.4.9. Model Templates

lavaangui contains two model templates, the factor template and the latent grown curve template, which allow the creation of those models rapidly. As an example, to create the "speed" factor in one step, drag the *Factor* rectangle into the *Model Window* and select the observed variables "x7", "x8", and "x9". All that is left is to rename the factor to "speed". Apply the *Recommended: Tree* layout again. Your path diagram should now resemble Figure 4d. This concludes the drawing of the three-factor model.

3.4.10. Autocompleted Model

It is important to clarify the distinction between the *User Model* and the *Autocompleted Model* modes that can be selected in the *Bottom Toolbar*. Click on *Autocompleted Model* now. Your path diagram should now resemble Figure 4e. As you can see, many arrows have been added to the diagram and thus model. lavaangui follows the philosophy of lavaan that model specification should be as easy as possible. Thus, it suffices to draw the core arrows, in this case, the directed arrows from the factors to the observed variables. lavaangui does its best to modify the model such that the resulting model meets the minimum requirements for a valid model and to add the arrows that most users would typically expect. This involves adding arrows. In the example, the arrows representing the residual variances and the covariances between the factors are added. It also involves fixing certain relationships. In the example, the arrows from the factors to their first observed variables are fixed at the value 1. To visually distinguish user-created and autocompleted arrows, the added and modified arrows are shown as dashed. The model autocompletion internally relies on lavaan and is thus identical to the model autocompletion in lavaan. The difference is that in lavaan model autocompletion and model fitting are usually done in one step. In contrast, in lavaangui, model autocompletion and fitting are two separate steps. The idea behind this is that users should first visually confirm whether the autocompleted model is correct before fitting it. This is important because fitting a model can sometimes take very long. Thus, it is imperative first to verify that the model is correct.

3.4.11. Reactivity

The *Autocompleted Model* mode is what is called reactive. When changing the path diagram, lavaangui automatically reacts to these changes and updates the autocompletion on the fly accordingly. For example, if you remove the observed variable "x7", while in *Autocompleted Model*, lavaangui immediately fixes the arrow from the "speed" latent variable to "x8" to 1 since this is now the first observed variable for the "speed" factor. The other two modes, *User Model* and *Estimates*, are also reactive. In the *User Model* mode, this implies that the *lavaan* script updates automatically while the diagram is drawn. In the *Estimate mode*, this implies that the model is refitted any time a change is made to the model. Should the automatic updating not work, you can force an update by clicking on the button of the respective mode. Before proceeding, make sure to undo the deletion of "x7".

3.4.12. Fitting Model

After you have confirmed that the autocompleted model is correct, you can fit it. To do this, click on *Estimates* in the *Bottom Toolbar*. After a brief delay, the estimates are shown. In particular, the estimated values are shown on top of the respective arrows in the path diagram, and a summary of the model fit, as generated by `lavaan`, is displayed in the *Results Window*.

3.4.13. Editing Arrows

After applying the *Recommended: Tree* layout again, the path diagram should look as displayed in Figure 1a. This path diagram requires some manual tweaking. One problem is that the arrow representing the residual variance of "textual" intersects with the undirected arrow between "visual" and "speed". To solve this, you can bend the arrow. To bend an undirected arrow, first select it. Then, bend it by clicking and holding the mouse and moving your mouse. After you release your mouse, a black square will appear. This is a so-called "Bend Point". If you want to bend the arrow again, select the arrow. The control point will appear, and you can bend the arrow by moving the control point with your mouse. Alternatively, you can create new bend points by bending the arrow as in the first step or by right-clicking on the arrow and selecting "Add Bend Point". Similarly, you can remove bend points by right-clicking on a bend point and choosing "Remove Bend Point". For directed arrows, editing works slightly differently. You have to explicitly create a "Bend Point" through right-clicking; selecting and dragging the arrow does not work. Additionally, you can create a "Kink Point," which allows you to introduce kinks in the arrow. In Figure 1, the arrow between "visual" and "x3" contains a kink. Note that you cannot mix kink and bend points for the same arrow.

Another modification of arrows that will often be needed is to change the location of (residual) variance arrows. Most of these arrows will be autocompleted by `lavaangui`, and `lavaangui` has an algorithm that tries to determine the best location. However, this will not produce good results for all models. To manually change the location of a variance arrow, drag it to the desired location. Should you want `lavaangui` to optimize the location again at a later point, right-click on the arrow and select *Free Loop Orientation*.

Another modification of arrows that is frequently needed is to change the location of labels. This is particularly the case for directed arrows. To change the location of a label for a directed arrow, drag it into the desired location. Note that `lavaangui` forces the label to be on top of the arrow.

3.4.14. Customizing Results

By default, parameter (point) estimates are shown in the diagram. The *View* menu offers multiple other options. Exemplarily select "Standardized Estimate" and "Estimate + p-values (stars)" to show the standardized estimate as well as their corresponding p values in star notation. Note that you have to be in the *Estimates* mode to see the (standardized) estimates. For this tutorial, this path diagram is considered finished now, but there are many more options to change its appearance, as will be explained later.

3.4.15. Exporting Path Diagram

You can export the path diagram to multiple file formats via *File* → *Export Diagram to X*. Exemplarily, to export the path diagram to PDF, which is the recommended file format, select *Export Diagram to PDF*. This will download a PDF file of the diagram to your computer. This file is displayed in Figure 5.

3.4.16. Saving Model

In the *File* menu, you can find options for saving your model and data. As the standard workflow, I recommend choosing *Download Model and Data*. This will download a ".lvd" file to your computer. This file contains your (fitted) model and the data set. When this file is loaded through *Load Model and Data*, your model and data sets are imported into `lavaangui`.

3.4.17. Additional Results

Besides the main results shown in the path diagram and the *Results Window*, `lavaangui` also offers additional results. These include parameter estimates in tabular form, factor score estimates, modification indices, model residuals, tests of exact fit, and user-defined hypothesis tests. The additional results can be accessed by clicking on the *More Results* button in the *Button Toolbar* and should be self-explanatory for users familiar with the concepts. Note that this button is only enabled when you are in the *Estimates* mode.

4 Secondary Use-Cases

Besides the main functionality just described, `lavaangui` has two important secondary use cases: Creating an interactive path diagram from a model specified in `lavaan` and drawing path diagrams without any model fitting.

4.1. Plotting `lavaan` Models

Note that this section should be skipped by non-R users. To create the interactive path diagrams `lavaangui` offers from `lavaan` models you can use `plot_lavaan` function. As an example, to obtain the interactive path diagram for the famous political democracy SEM, `lavaan` users can write the following commands into the R console.

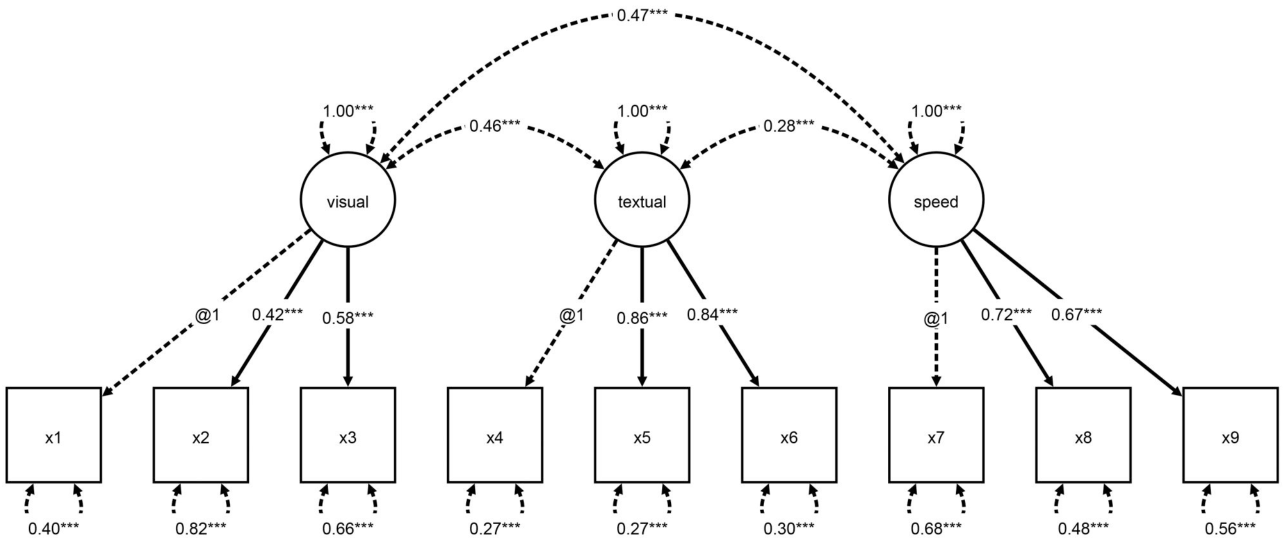


Figure 5. Path diagram for the three-factor model as generated by `lavaangui`.

```
library(lavaan)
library(lavaangui)
model < - '
  # latent variable definitions
     ind60 =~ x1 + x2 + x3
     dem60 =~ y1 + a*y2 + b*y3 + c*y4
     dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # regressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60

  # residual covariances
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
'

fit < - sem(model, data = PoliticalDemocracy)
plot_lavaan(fit)
```

This initially displays the path diagram as shown in Figure 6a. For RStudio users, the diagram is shown directly in RStudio, for others, it is opened in their default browser. Crucially, you can modify the appearance of the diagram interactively in the same manner as was described earlier in this tutorial. For example, you can move variables with your mouse or bend arrows. This allows for much more direct and, in many aspects, greater control over the appearance of your path diagrams compared to traditional solutions that do not allow interactive editing and require all changes to be made through programming commands. To demonstrate this, Figure 6b shows the path diagram after some manual customizations I performed. In particular, I moved the variables “x1”, “x2”, and “x3” closer together, bent the undirected arrows connecting the observed variables, and moved some of the variance arrows as well as the labels of the directed arrows. After your customizations, you can save an editable version of your path diagram through *File* → *Download Model* and export the path diagram as described above.

4.2. Only Drawing Path Diagrams Without Fitting

To only draw path diagrams without fitting a model, first ensure that the *User Model* mode is selected. The only difference compared to drawing a path diagram for model fitting is that you do not load any data, so the *Top Toolbar* will not contain any named rectangles representing your *observed* variables. Instead, to create an observed variable, drag and drop the unnamed rectangle from the *Top Toolbar* into the *Model Window*. Alternatively, you can right-click on the *Model Window* and select *Add Observed Variable*, or press the “o” key while your mouse is in the *Model Window*.

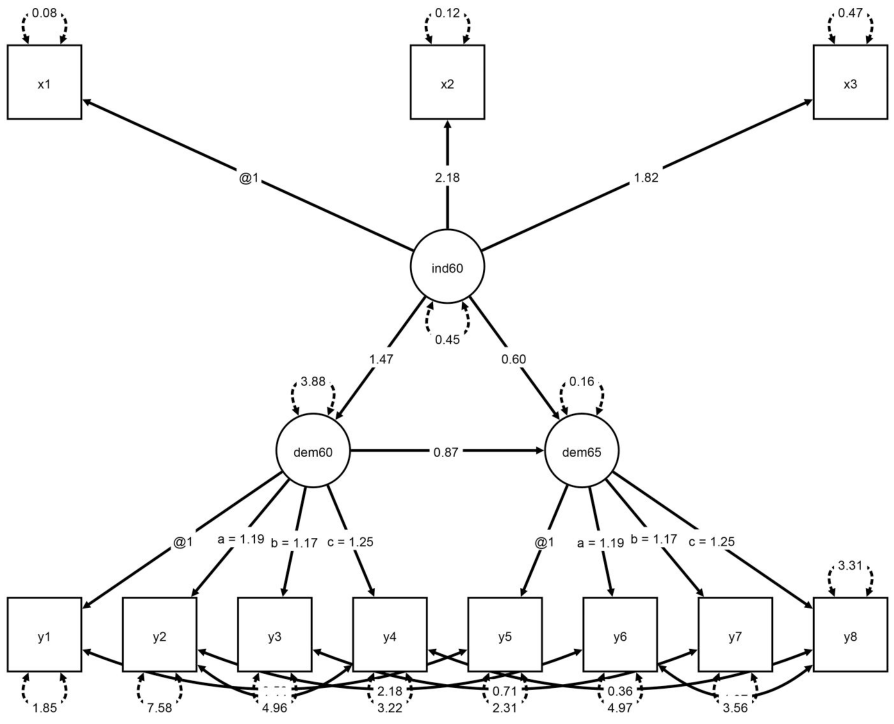
5. Other Features of Lavaangui

5.1. Parameter Labels

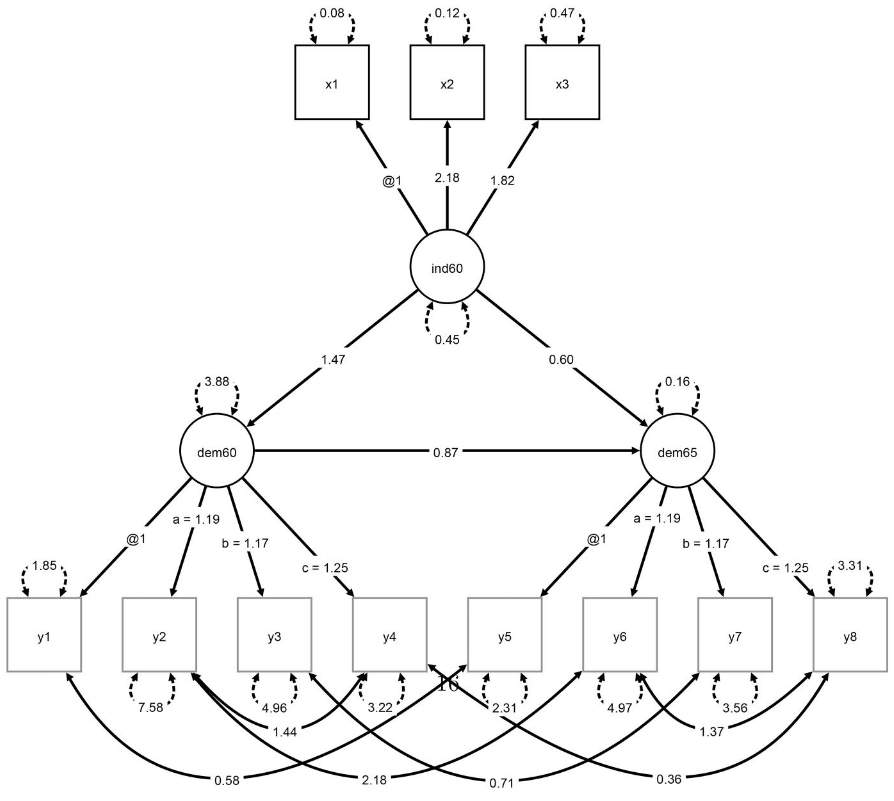
`lavaangui` supports parameter labels, similar to `lavaan`. To label an arrow, right-click on it and select *Add/Change Label*. Labels can be used to define equality constraints. If two arrows are given the same label, their parameters will be constrained to be equal during estimation.

5.2. Changing Autocomplete

In case the autocomplete functionality does not produce the intended model, there are two options. The first is to modify how `lavaangui` autocompletes the path diagram. These settings can be found in the *Automatically…* menu. Alternatively, autocomplete changes can be overwritten, as the user model always has priority. For example, if you prefer the arrow from a factor to the second corresponding observed



(a) Without Customization



(b) With Customization.

Figure 6. Path diagram for the political democracy model.

variable to be fixed to one instead of the first, you can force the first arrow to be freely estimated. To do this, right-click on the arrow and select *Force Parameter Free*. To change an arrow added by `lavaangui`, the easiest approach is to right-click on the respective arrow and choose *Include in User Model*. After that, the arrow can be modified as usual.

5.3. Estimation & Missing Data

By default, `lavaangui` relies on `lavaan` to select the appropriate estimator for the model. Alternatively, the estimator can be manually selected via the *Estimation* → *Estimator* menu. Similarly, standard errors can be adjusted using the *Estimation* → *Standard Error* menu. The *Estimation* menu also allows for changes in how missing data are treated.

5.4. Ordered Variables

By default, lavaangui assumes that the variables are continuous. To declare a variable as ordered, right-click on it and choose *Change to Ordered*. If the estimator is changed, it is important to ensure that it supports ordered data. If unsure, 'default' is a good choice.

5.5. Non-Normal Data

Non-normal continuous data are supported in the same way as in lavaan (Rosseel, 2012). Non-normality robust standard errors (referred to as simply *robust* in lavaangui) or bootstrap standard errors can be selected in the *Estimation* menu. A non-normality corrected test statistic can be requested via *More Results* → *Exact Test* by selecting "Satorra-Bentler" as the test statistic.

5.6. Grid

Besides the automatically appearing helper lines and the layout algorithm, lavaangui also includes a grid to help with the arrangement of variables. This can be activated by clicking on *Grid* → *Show Grid*. The *Grid* menu also allows customizing the grid. The options should be self-explanatory.

5.7. Font Size, Color, Variable Size, & Line Width

The font size and colors of the variables and arrows as well as the line width of the arrows can be adjusted by right-clicking on the respective elements and selecting the appropriate menu item. The appearance of multiple elements can be changed simultaneously by selecting them together. To resize a variable, first select it. A resize handler will appear and you can drag it into the desired shape.

5.8. Cancel Fitting

Sometimes lavaan will take a long time to fit a model. lavaangui offers the option to cancel model fitting at any time. To do so, click on the *Cancel* button that appears while a model is fitted.

5.9. Hiding Arrows

To draw more attention to the important arrows in your diagram, you can hide certain arrows through the View menu.

5.10. Autosave

Autosave is only enabled when using lavaangui via http://lavaangui.org. The server may close the connection due to inactivity to free up resources. To prevent data loss, lavaangui saves the model and data before the connection closes. After a timeout, refresh the page to reload your model and data.

6. Concluding Remarks

This paper provided a tutorial on lavaangui. The main attraction of lavaangui is that it is a free and open source (FOSS) graphical user interface with a diagrammer for the most popular FOSS SEM software lavaan. Additionally, it can be used to plot lavaan models. Crucially, in contrast to existing plotting packages, the produced path diagrams are interactive.

6.1. How to Choose Between Lavaan and Lavaangui

I will now discuss when I would advise to specify a model in lavaan or lavaangui. Note that lavaangui can still be useful for plotting models even when using lavaan for specification.

Beginners in structural equation modeling should start with lavaangui, particularly if they are not active R users. The intuitive graphical interface of lavaangui is particularly suitable for beginners, enabling users to construct path diagrams through simple mouse clicks and gestures without the need to learn a new syntax or even a programming language. By removing the burden of learning specific commands, beginners can concentrate on gaining a deeper understanding of SEM concepts. Compared to almost any other SEM program, lavaangui has the additional advantage that there is no installation necessary; users can begin performing SEM analysis immediately. This should be especially useful in classroom settings.

Eventually, with more experience, users should switch to lavaan for at least some models. lavaangui can aid in the transition to lavaan by showing the equivalent lavaan script for path diagrams. For models that are not currently supported in lavaangui, such as multilevel SEMs, or when a functionality not currently supported by lavaangui is needed, lavaan is the only option. However, also for supported models, lavaan can be a better choice. The main reason is that while drawing a path diagram is initially much easier and less error-prone than specifying the model via syntax, this approach is not without limitations. Drawing the model can become very tedious, especially if many variables are involved. To slightly alleviate this problem, I have included templates for the most popular models so that, for example, drawing a one-factor model is instantaneous, no matter how many observed variables are involved. However, drawing models with many variables for which there is no template available remains tedious. Additionally, models with numerous variables and arrows are difficult to arrange in a way that produces a readable path diagram. While lavaangui excels in displaying large models due to its included automatic layouts and zoom functionality, for some models, it will still be very difficult to achieve a readable path diagram. Another reason to prefer lavaan is that a lavaan analysis can be easily included in an R script, allowing for the reproduction of all analyses performed for publication. While lavaangui analyses can be saved and thus reproduced, inclusion in an overall analysis script is not currently possible.

6.2. Future Plans

To alleviate the reproducibility problem just discussed, future versions of lavaangui will include functionality that allows exporting the full analysis as performed in lavangui into an equivalent R script and data set package. Additionally, future versions will support mobile devices, indirect effects, model comparison via the likelihood ratio test, linear and nonlinear parameter constraints, support for multiple groups, multilevel SEM, and plotting of SEMs specified in the R packages OpenMx and sem.

Acknowledgements

I thank Yves Rosseel, Ladan Rahbari, Terrence Jorgensen, Bunga Pratiwi, Katie Lindefjeld, Juan Claramunt Gonzalez, Andres Perez Alonso, Sjoerd Huisman, Jeffrey Durieux, and Andreas Brandmaier for testing the software and for their suggestions for improvements. I thank Ladan Rahbari for thoroughly testing the tutorial presented in this work and for her editorial assistance.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Arbuckle, J. L. (2019). *IBM SPSS Amos 26 User's Guide* [Computer software manual].

Bentler, P. M. (2006). *EQS 6 Structural Equations Program Manual* [Computer software manual].

Boker, S. M., McArdle, J., & Neale, M. (2002). An algorithm for the hierarchical organization of path diagrams and calculation of components of expected covariance. *Structural Equation Modeling: A Multidisciplinary Journal*, *9*, 174–194. https://doi.org/10.1207/S15328007SEM0902_2

Epskamp, S. (2015). semplot: Unified visualizations of structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *22*, 474–483. https://doi.org/10.1080/10705511.2014.937847

Ernst, M. S., & Peikert, A. (2024). Structural equation models.jl (v0.2.2). Zenodo. https://doi.org/10.5281/zenodo.10475661

Fox, J., Nie, Z., & Byrnes, J. (2022). sem: Structural equation models [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=sem (R package version 3.1-15)

Franz, M., Lopes, C. T., Huck, G., Dong, Y., Sumer, O., & Bader, G. D. (2016). Cytoscape.js: A graph theory library for visualisation and analysis. *Bioinformatics*, *32*, 309–311. https://doi.org/10.1093/bioinformatics/btv557

Hamilton, W. K. (2017). lavaan.shiny: Latent variable analysis with shiny [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=lavaan.shiny (R package version 1.2)

Igolkina, A. A., & Meshcheryakov, G. (2020). Semopy: A python package for structural equation modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, *27*, 952–963. https://doi.org/10.1080/10705511.2019.1704289

Iguchi, T. (2018). ezsem. https://github.com/ToshihiroIguchi/ezsem. GitHub.

JASP Team. (2024). JASP (Version 0.18.3) [Computer software]. Retrieved from https://jasp-stats.org/

Jöreskog, K. G., & Sörbom, D. (1996). *LISREL 8: User's reference guide* [Computer software manual].

Lishinski, A. (2024). lavaanplot: Path diagrams for 'lavaan' models via 'diagrammer' [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=lavaanPlot (R package version 0.8.1)

Mai, Y., Xu, Z., Zhang, Z., & Yuan, K.-H. (2023). An open-source wysiwyg web application for drawing path diagrams of structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *30*, 328–335. https://doi.org/10.1080/10705511.2022.2101460

Marcoulides, G. A., & Papadopoulos, D. (1993). Lispath: A program for generating structural equation path diagrams. *Educational and Psychological Measurement*, *53*, 675–678. https://doi.org/10.1177/0013164493053003008

Mayer, A. (2017). lavaangui: Graphical user interface for lavaan [Computer software manual]. Retrieved from https://github.com/amayer2010/lavaanGUI (R package version 0.1-1.004)

Muthén, L. K., & Muthén, B. O. (1998-2017). Mplus User's Guide (8th ed.) [Computer software manual.]. Muthén & Muthén. (Original edition 1998)

Neale, M. C., Hunter, M. D., Pritikin, J. N., Zahery, M., Brick, T. R., Kirkpatrick, R. M., Estabrook, R., Bates, T. C., Maes, H. H., & Boker, S. M. (2016). Openmx 2.0: Extended structural equation and statistical modeling. *Psychometrika*, *81*, 535–549. https://doi.org/10.1007/s11336-014-9435-8

Recka, K. (2024). shinylavaan. https://github.com/reckak/shinyLavaan. GitHub.

Reingold, E. M., & Tilford, J. S. (1981). Tidier drawings of trees. *IEEE Transactions on Software Engineering*, *SE-7*, 223–228. https://doi.org/10.1109/TSE.1981.234519

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software*, *48*, 1–36. https://doi.org/10.18637/jss.v048.i02

StataCorp LLC. (2023). *Stata statistical software: Release 18* [Computer software manual].

The jamovi project. (2024). jamovi (version 2.5) [computer software]. https://www.jamovi.org

van Lissa, C. J. (2023). tidysem: Tidy structural equation modeling [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=tidySEM (R package version 0.2.6)

von Oertzen, T., Brandmaier, A. M., & Tsang, S. (2015). Structural equation modeling with ωnyx. *Structural Equation Modeling: A Multidisciplinary Journal*, *22*, 148–161. https://doi.org/10.1080/10705511.2014.935842

Yung, Y.-F. (2010, August 4). *Introduction to structural equation modeling using the calis procedure in sas/stat software*. Computer technology workshop presented at the Joint Statistical Meeting, Vancouver, Canada.