Chapter 11

Unified Visualizations of Structural Equation Models

Abstract

Structural Equation Modeling (SEM) has a long history of representing models graphically as path diagrams. This chapter presents the freely available semPlot package for R, which fills the gap between advanced, but time-consuming, graphical software and the limited graphics produced automatically by SEM software. In addition, semPlot offers more functionality than drawing path diagrams: it can act as a common ground for importing SEM results into R. Any result useable as input to semPlot can be also represented in any of the three popular SEM frameworks, as well as translated to input syntax for the R packages sem (Fox, 2006) and lavaan (Rosseel, 2012). Special considerations are made in the package for the automatic placement of variables, using three novel algorithms that extend earlier work of Boker, McArdle, and Neale (2002). The chapter concludes with detailed instructions on these node-placement algorithms.

11.1 Introduction

The semPlot package for the freely available statistical programming language R (R Core Team, 2016) extends various popular structural equation modeling (SEM) software packages with a free, easy to use and flexible way of producing high quality graphical model representations—commonly termed *path diagrams*—as well as providing a bridge between these software packages and the main SEM frameworks.

A path diagram utilizes a network representation, in which variables are represented as nodes—square nodes indicating manifest variables, circular nodes indicating latent variables and triangular indicating constants—and relations between

This chapter has been adapted from: Epskamp, S. (2015). semPlot: Unified visualizations of Structural Equation Models. *Structural Equation Modeling, 22*, 474–483.

variables are represented by a set of unidirectional and bidirectional edges, which typically represent regression equations and (co)variances respectively.

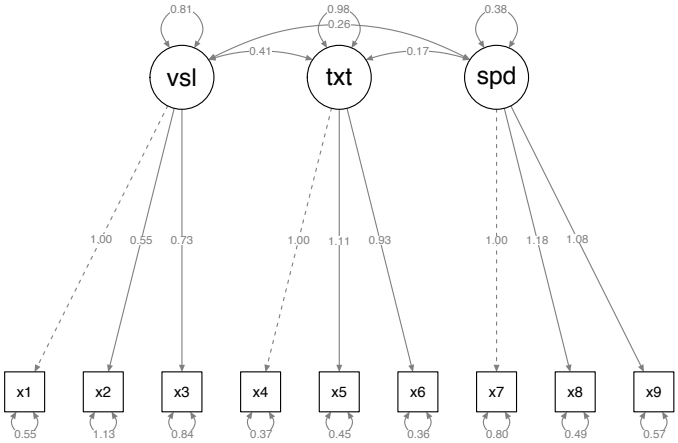
Currently there are two common ways of drawing path diagrams. Many of the available SEM software packages have an option to display the path diagram graphically, either directly in the package (e.g., LISREL; Jöreskog & Sörbom, 1996), by creating syntax for external network drawing software (e.g., sem; Fox, 2006) or through third party extensions (e.g., Lispath; Marcoulides & Papadopoulos, 1993). Some packages in addition allow the model to be specified in a graphical way, by letting the user draw the path diagram directly in an interactive command window (e.g., Amos; Arbuckle, 2010, MPlus; Muthén & Muthén, 1998–2012, PLSgraph; Chin, 2001, and Onyx; von Oertzen, Brandmaier, & Tsang, 2013). Alternatively, instead of generating a path diagram from a given model, the path diagram can also be drawn manually, using many free and commercial software packages (e.g., Cytoscape; Shannon et al., 2003, Microsoft® Powerpoint® and igraph; Csardi & Nepusz, 2006).

Both of these methods, however, have important limitations. The path diagrams created by SEM packages produces path diagrams that are hardly customizable, and produce images unsuited for publication. On the other hand, manually drawing path diagrams in external software can take a very long time and is prone to error. The semPlot package offers a middle way; it is designed to automatically produce high quality path diagrams from the output of various popular SEM software packages, while retaining a high level of customizability. Thus, in semPlot, the user feeds a raw output file to the program, which then returns a high-quality image ready for publication. In addition, as will be described below, semPlot creates an internal model representation that can serve as a translator between SEM programs; for instance, on the basis of, say, LISREL model *output*, semPlot automatically generates the corresponding lavaan (Rosseel, 2012) *input*.

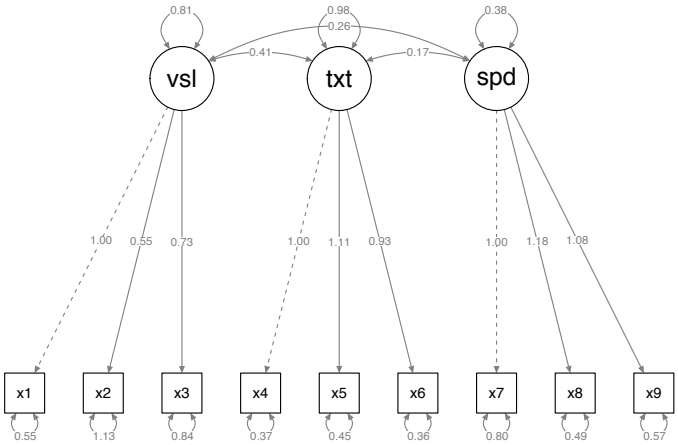
The semPlot package supports the output from R packages sem (Fox, 2006), lavaan (Rosseel, 2012), OpenMx (RAM specification only; Boker et al., 2011) and standalone software MPlus (Muthén & Muthén, 1998–2012, using R package MplusAutomation for the import; Hallquist & Wiley, 2013), LISREL (Jöreskog & Sörbom, 1996, using R package lisrelToR for import; Epskamp, 2013) and Onyx. Several base R functions for related statistical techniques such as exploratory factor analysis and general linear modeling are also supported. In addition, semPlot can also be used without the need of fitting a SEM using the lavaan modeling syntax, or matrix specification according to the RAM (McArdle & McDonald, 1984), LISREL (Hayduk, 1987) and Mplus (Muthén, 1998–2004) modeling frameworks.

The graphs that semPlot produces are drawn using the qgraph package, which itself is designed as a network drawing package aimed at applications in statistical visualizations. Customization of the graphs can be done either via semPlot itself (using many options designed for SEM models, such as omitting exogenous variances) or post-hoc via the qgraph package (using options designed to visualize graphs, such as manually recoloring edges).

This chapter consists of two sections: the first section describes the functionality of the package and in the second section describes the algorithms used for automatically constructing a path diagram.



(a)



(b)

Figure 11.1: Generated path diagram of the Holzinger-Swineford CFA example. Panel (a) shows a visualization of the path diagram with estimates as labels and Panel (b) shows a visualization of the standardized parameter estimates.

11.2 General Use of the semPlot Package

The semPlot package can be downloaded from CRAN or installed directly in R:

```
install.packages("semPlot")
```

After which the package can be loaded:

```
library("semPlot")
```

This will load the functions from the semPlot package into R.

Drawing Path Diagrams

The `semPaths` function can be used to plot path diagrams and visualize (standardized) parameter estimates. It takes as first argument either a SEM object (from R packages) or a string indicating the location of an output file from external SEM software (MPlus or LISREL). The second and third arguments can be assigned strings indicating what the edge color and label respectively indicate. For example, the following code plots a model where the edges are colored according to standardized values and the edge labels indicate the unstandardized estimates:

```
semPaths(input, "standardized", "estimates", ...)
```

Where `...` indicate any number of other arguments controlling the output which are further explained in the package manual:

```
?semPaths
```

To illustrate this, one could use one of the lavaan package documentation examples to compute a confirmatory factor analysis (CFA) on the famous Holzinger and Swineford (1939) example:

```
library("lavaan")
example(cfa)
```

Next, sending the resulting `fit` object to `semPaths` plots a path diagram of the model with parameter estimates on the labels:

```
semPaths(fit, "model", "estimates")
```

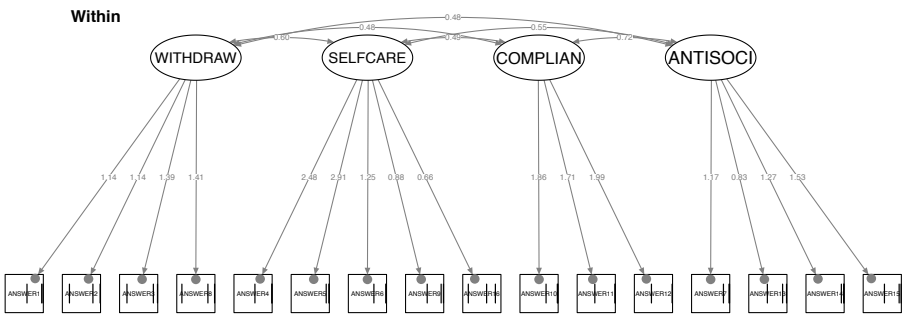
We could also visualize the parameter estimates by coloring positive parameters green or red indicating positive or negative estimates and varying the width and color of an edge to indicate the strength of the estimate (see Chapter 9). This works best with standardized parameters:

```
semPaths(fit, "standardized", "hide")
```

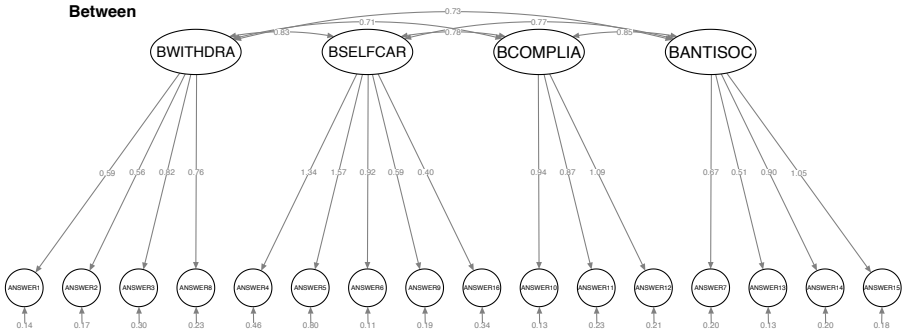
The resulting graphs can be seen in Figure 11.1. This figure also shows that fixed parameters—in this case scaling by fixing factor loadings—are visualized by default by using dashed lines.

The semPlot package can handle larger complicated measurement models. The next example is based on the Mplus output of the multilevel factor analysis model as described by Little (2013), in which the factor structure of the Life Skills Profile-16 (LSP-16) was assessed. The following codes produce the two plots in Figure 11.2:

```
semPaths(file.choose(), "model", "estimates",
         style = "lisrel", curve = 0.8, nCharNodes = 0,
         sizeLat = 12, sizeLat2 = 6, title = TRUE,
         mar = c(5, 1, 5, 1), edge.label.cex = 0.5)
```



(a)



(b)

Figure 11.2: Generated path diagram for multilevel factor analysis model of LSP-16. Panel (a) shows the within-cluster model, with vertical bars representing the estimated thresholds of each of the ordinal variables. Panel (b) shows the between-cluster model.

In which `file.choose()` is a base R function that opens a convenient file browser to select the Mplus output file.

Figure 11.2 shows that two plots are now generated: one indicating the within-cluster model and one indicating the between-cluster model. In the within-cluster model the closed orbs inside manifest indicate random intercepts and the vertical bars inside the manifest variables indicate the estimated thresholds; in the between-cluster model the indicators are represented by a circle for random intercepts.

The argument `style = "lisrel"` specifies that (residual) variances are plotted similar to the way LISREL plots these: as arrows without origin on endogenous variables only. The default, `style = "ram"`, would plot these residuals as described by Boker et al. (2002): as double-headed self-loops on both endogenous and exogenous variables. To illustrate this consider an example of the famous 'Industrialization and Political Democracy' dataset used by Bollen (1989), which

has been implemented as example in the Lavaan package:

```
library("lavaan")
example(sem)
semPaths(fit, "model", "hide", style = "lisrel",
         rotation = 2)
semPaths(fit, "model", "hide", style = "ram", rotation = 2,
         cardinal = "man cov")
```

The resulting graphs can be seen in Figure 11.3.

Color can also indicate equality constrains: by coloring parameters that are constrained to be equal with the same color (unconstrained parameters are still colored gray)—especially useful in identifying the different steps in assessing measurement invariance (Meredith, 1993). For example, the semTools package (Pornprasertmanit, Miller, Schoemann, & Rosseel, 2013) can be used to test for measurement invariance using lavaan on the Holzinger and Swineford (1939) example:

```
library("semTools")
fits <- example(measurementInvariance)
semPaths(fits$value$fit.intercepts, "equality", "estimates",
         sizeLat = 5, title = FALSE, ask = FALSE,
         levels = c(1, 2, 4), edge.label.cex = 0.5)
```

Figure 11.4 shows one of the steps in testing for measurement invariance: strict measurement invariance with free factor means. It can be seen that the factor loadings and intercepts are constrained to be equal over groups, but the factor means and variances are not.

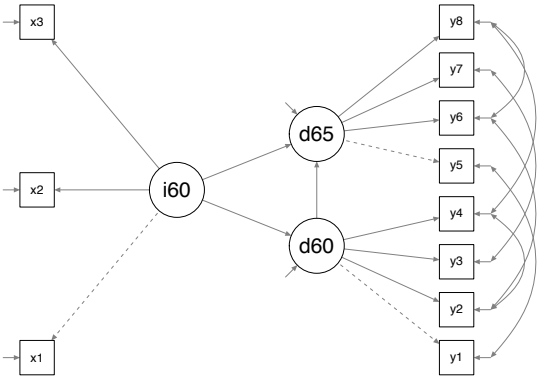
Investigating Correlational Structures

SEM models are usually fit by comparing the observed covariances to the model implied covariances. The qgraph package used as back-end to semPlot supplies a novel framework for visualizing correlational structures as networks (as is described in Chapter 9): a correlation matrix can be visualized as a network in which each variable is represented by a node and each correlation as a weighted edge between two nodes.

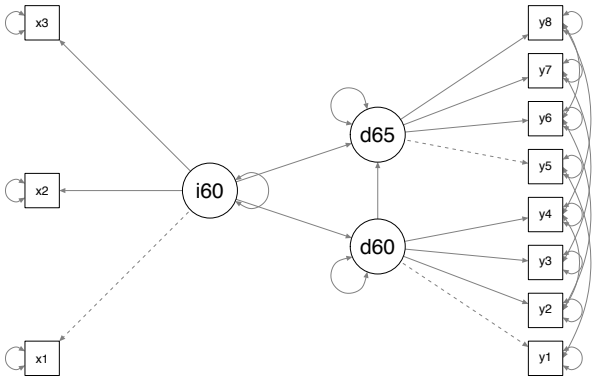
In the semPlot package, the `semCors` function visualizes the model implied correlation matrix (which is either provided as input or computed from data) and the observed correlation matrix (must be provided as input) using qgraph with parameters automatically chosen such that the graphs are comparable. To illustrate this, consider the following simulated dataset (using lavaan):

```
library("lavaan")

Mod <- '
A =~ 1*a1 + 0.6*a2 + 0.8*a3
B =~ 1*b1 + 0.7*b2 + 0.9*b3
a1 ~~ 1*b1
A ~~ -0.3* B
```



(a)



(b)

Figure 11.3: Generated path diagrams for Industrialization and Political Democracy dataset example. Panel (a) shows the path diagram with residuals drawn in 'lisrel' style and Panel (b) shows the path diagram with residuals drawn in 'ram' style.

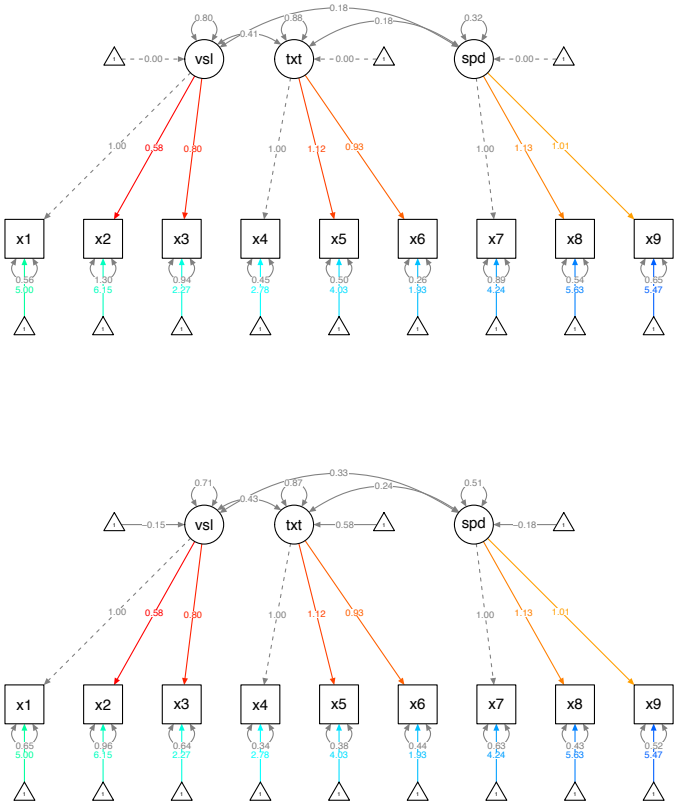


Figure 11.4: Path diagrams for two groups in the Holzinger-Swineford CFA example, testing for strict measurement invariance with free factor means.

```
,

set.seed(5)
Data <- simulateData(Mod)
```

This dataset, called `Data`, is simulated under a two-factor model with two negatively correlated factors. However, the residuals of the first indicator of each factor are strongly positively correlated. After fitting a general CFA model to this data, not including the residual correlation, the implied and observed correlation matrices can be inspected:

```
Mod <- '
A =~ a1 + a2 + a3
B =~ b1 + b2 + b3
```

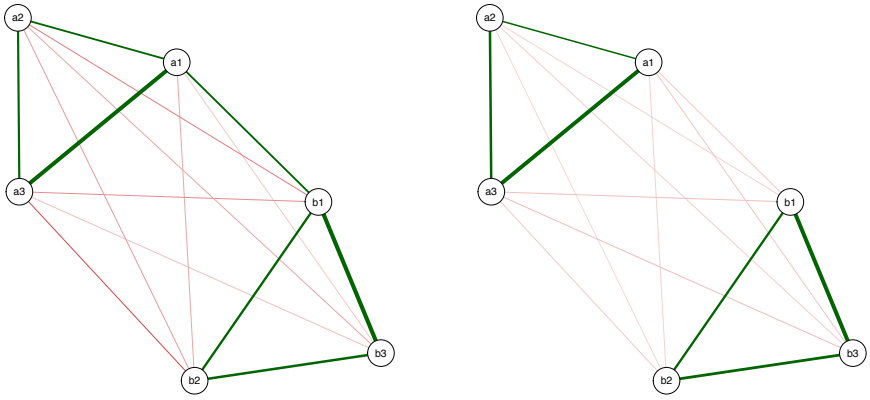


Figure 11.5: Observed (left) and model implied (right) correlation matrices of simulated data example.

```
,
fit <- cfa(Mod, data=Data)
semCors(fit, layout = "spring", cut = 0.3, esize = 20)
```

Figure 11.5 shows that the observed and implied correlation matrices are very similar except for the correlation between `a1` and `b1`, which cause the misfit in this model. This provides a visual way of judging the fit of a SEM model and a way of seeing where misfit is occurring.

Linking SEM Software Packages and Models

An important design philosophy of semPlot is unifying different SEM software packages in a freely available interface. To this end, the package can also be used as bridge between different SEM software packages and SEM models. First, the `semSyntax` function generates model syntax for R packages sem and lavaan given any input supported in semPlot. For example, the output file of example 5.1 of the MPlus user guide (Muthén & Muthén, 1998–2012) can be imported:

```
ex5.1 <- tempfile(fileext = ".out")
url <- "http://www.statmodel.com/usersguide/chap5/ex5.1.out"
download.file(url, ex5.1)
```

Next, the file can be used to generate a model to use in the lavaan package:

```
lavMod <- semSyntax(ex5.1, "lavaan")
## Model <- '
## F1 =~ 1*Y1
## F1 =~ Y2
## (...)
```

```
## Y5 ~~ Y5
## Y6 ~~ Y6
## '
```

The function returns an object, and prints the R script needed to create this object. A useful application of this bridge is to simulate data in R given any SEM output file using lavaan's `simulateData` function. To do this, first specify the model with all estimated parameters set to fixed:

```
lavMod <- semSyntax(ex5.1, "lavaan", allFixed = TRUE)
```

Next the model can be sent to `simulateData`:

```
head(simulateData(lavModFixed))
##          Y1      Y2      Y3      Y4      Y5       Y6
## 1  0.88695  0.2414  0.8060  0.6778  1.2228 -0.34377
## 2  1.30715 -0.4904  0.8651  0.4772  0.4611  0.58303
## 3 -0.62939 -1.5140 -0.3916  1.0225  1.2060 -0.65448
## 4  0.99210 -1.8682 -1.0856  0.3514 -0.3357 -2.01952
## 5  0.02836 -0.4113 -0.3776 -1.1781  0.1050 -1.23260
## 6  1.12654  1.9011  1.0472  0.6976 -0.8670 -0.03874
```

Second, the `semMatrixAlgebra` The `semMatrixAlgebra` function offers a unified interface for extracting model matrices of any of the three major SEM frameworks, RAM (McArdle & McDonald, 1984), LISREL (Hayduk, 1987) and Mplus (Muthén, 1998–2004), using any of the supported input software packages. For example, the RAM framework uses three model matrices: $\boldsymbol{A}$, $\boldsymbol{S}$ and $\boldsymbol{F}$:

$$\begin{aligned} \boldsymbol{v} &= \boldsymbol{A}\boldsymbol{v} + \boldsymbol{u} \\ \boldsymbol{u} &\sim N(\boldsymbol{0}, \boldsymbol{S}) \\ \text{Var}(\boldsymbol{v}) &= \boldsymbol{F}\left(\boldsymbol{I} - \boldsymbol{A}\right)\boldsymbol{S}\left(\boldsymbol{I} - \boldsymbol{A}\right)^{-1^\top}\boldsymbol{F}^\top \end{aligned}$$

In which $\boldsymbol{v}$ is a vector containing both manifest and latent variables, $\boldsymbol{A}$ a matrix of regression parameters (usually termed the asymmetric matrix), $\boldsymbol{S}$ a matrix of (residual) variances (usually termed the symmetric matrix) and $\boldsymbol{F}$ (usually termed the filter matrix) can be used to distinguish between latent and manifest variables. `semMatrixAlgebra` can be used to extract e.g., the $\boldsymbol{A}$ matrix of Mplus user guide example 5.1:

```
semMatrixAlgebra(ex5.1, A)
##       F1    F2 Y1 Y2 Y3 Y4 Y5 Y6
## F1 0.000 0.000  0  0  0  0  0  0
## F2 0.000 0.000  0  0  0  0  0  0
## Y1 1.000 0.000  0  0  0  0  0  0
## Y2 1.126 0.000  0  0  0  0  0  0
## Y3 1.019 0.000  0  0  0  0  0  0
## Y4 0.000 1.000  0  0  0  0  0  0
## Y5 0.000 1.059  0  0  0  0  0  0
## Y6 0.000 0.897  0  0  0  0  0  0
```

Note that the use of `A` automatically let `semMatrixAlgebra` detect that we are interested in the RAM framework specifically. Requesting matrices from other frameworks, such as the $\boldsymbol{\Lambda}$ matrix—containing factor loadings—from the MPlus modeling framework, works in the same way:

```
semMatrixAlgebra(ex5.1, Lambda)
##       F1    F2
## Y1 1.000 0.000
## Y2 1.126 0.000
## Y3 1.019 0.000
## Y4 0.000 1.000
## Y5 0.000 1.059
## Y6 0.000 0.897
```

The `semMatrixAlgebra` function cannot only be used for extracting individual model matrices but also for extracting the result of algebraic computations using these model matrices. For example, one could compute the implied covariances on the same example model as follows—using helper function `Imin(A,TRUE)` to compute $(\boldsymbol{I} - \boldsymbol{A})^{-1}$:

```
semMatrixAlgebra(ex5.1,
  F %*% Imin(A,TRUE) %*% S %*% t(Imin(A, TRUE)) %*% t(F))
##          Y1       Y2       Y3       Y4       Y5       Y6
## Y1  1.97100  1.02128  0.92423 -0.03000 -0.03177 -0.02691
## Y2  1.02128  1.94796  1.04069 -0.03378 -0.03577 -0.03030
## Y3  0.92423  1.04069  1.95179 -0.03057 -0.03237 -0.02742
## Y4 -0.03000 -0.03378 -0.03057  2.05000  0.80484  0.68172
## Y5 -0.03177 -0.03577 -0.03237  0.80484  1.70633  0.72194
## Y6 -0.02691 -0.03030 -0.02742  0.68172  0.72194  1.67750
```

`semMatrixAlgebra` returns the results in a list rather than a single matrix if the model contains multiple groups.

11.3 Algorithms for Drawing Path Diagrams

When drawing a path diagram the variables need to be placed in a structured way, such that the diagram is easily interpretable (Boker et al., 2002). Manually defining such a graph layout can be tedious and time-consuming work; an automated solution to placing variables would work best in most situations. This section introduces three novel algorithms—which are implemented in semPlot—that can be used to automatically place variables such that complex SEM models are easily interpretable.

The three layout algorithms are each designed to place variables in a tree-like structure next to each other on horizontal levels. They are chosen such that first, the structural part of the model—especially the relationship between exogenous and endogenous variables—is clearly visible; and second, Indicators of a latent

variable are placed next to each other and either below or above the latent variable. To achieve this, all three algorithms start with exogenous variables[1] or their indicators placed at the top level of the graph (level 0) and expand downwards to the bottom of the graph (level n).

The first algorithm is based on the way the LISREL program (Jöreskog & Sörbom, 1996) plots path diagrams. In this algorithm variables are placed on one of four horizontal levels. The top level contains manifest variables that are either exogenous themselves or only indicators of exogenous latent variables. The second level contains latent variables that are either exogenous themselves or regressed only on exogenous manifest variables. The third level contains all other (endogenous) latent variables and the fourth level contains all other (endogenous) manifest variables. Intercepts can be added by placing a representation of the unit vector next to or below/above each variable. In defining the horizontal placement the latent variables are placed in the order they appear in the model, and manifest variables are placed such that they are closest to latent variables they are connected to.

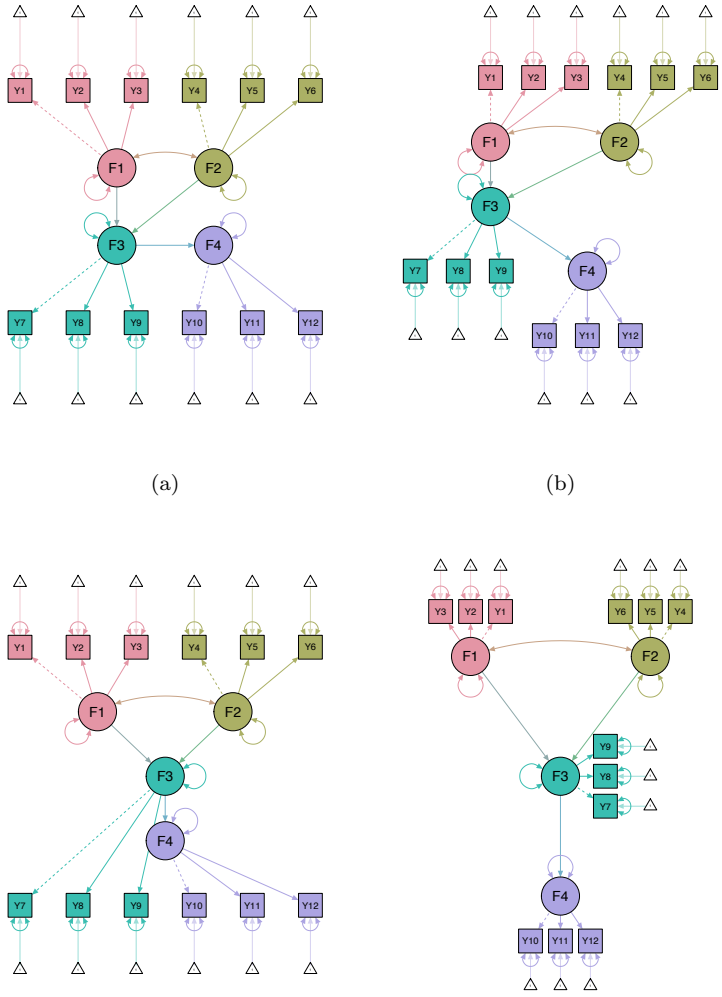
The second algorithm is a variation of the Reingold-Tilford algorithm (Reingold & Tilford, 1981) which places variables in a tree structure originating from a set of user defined root nodes at the top. The igraph package (Csardi & Nepusz, 2006) can be used to compute the Reingold-Tilford algorithm. However, in the presence of intercepts, exogenous latents, or covariances, this algorithm does not produce proper diagram structures out of the box. To solve this, the algorithm is applied to a modified version of the network representation of the model: by removing all arrows (making edges undirected) and removing all covariances. Through a specific choice of root variables, a tree structure is obtained in which exogenous variables are placed on top and endogenous variables at the bottom.

Finally, the third algorithm uses a variation of the placement algorithm described by Boker et al. (2002). This algorithm computes for each node the longest outgoing path, and places nodes accordingly on horizontal levels from highest (top) to lowest (bottom) longest outgoing path-length. For more stable results (e.g., indicators of exogenous latents should be placed above the latent), this algorithm can be enhanced by not using the original network representation of a model but one in which the direction of the edges between exogenous latent variables and their indicators is reversed and all double-headed edges (covariances) are removed.

In all three algorithms, horizontal levels that do not contain any nodes are not included in the graph, and if there are only exogenous latent variables and no regressions between manifest variables (e.g., factor analysis models) the layout is flipped. In cases which feature many indicators per latent variable, it is more useful to place variables in a circle-like fashion; here, the origin of the tree placement is not at the top, expanding to the bottom, but at the center, expanding outward. To do this, we may transform the horizontal levels to nested circles; the higher the level, the smaller the circle.

Often, the structural part of a model—containing only regressions between latent variables—is the only part that requires specifically thoughtful placement of variables; for the measurement parts—the factor loadings of indicators on each

[1] A variable is treated as exogenous if it has no incoming directed edges attached.



(a) (b)

(c) (d)

Figure 11.6: Path diagram including parameter estimates of example 5.25 of the Mplus user guide. Panel (a) shows default placement, panel (b) the adjusted Reingold-Tilford algorithm, panel (c) the adjusted Boker-McArdle-Neale algorithm and panel (d) a layout where only the structural part is based on the adjusted Boker-McArdle-Neale algorithm and the measurement sub-models are drawn around the latent variables.

latent variable—indicators simply need be placed on a straight line under, over, or next to the latent. To this end, it might not be necessary to run a complicated placement algorithm over all variables, but rather only over the structural part of a model, followed by placing indicators near the latent. Such a placement of nodes for the structural part of a model could be used on the basis of any of the above mentioned algorithms, but also through any network drawing algorithm (e.g., by using a force-embedded algorithm; Fruchterman & Reingold, 1991).

In `semPaths`, the `layout` argument can be used to control which algorithm is used to define the placement of the nodes. This argument can be set to `"tree"` to obtain the default layout, `"tree2"` to obtain the adjusted Reingold-Tilford algorithm or `"tree3"` to obtain the adjusted Boker-McArdle-Neale algorithm. To obtain circular versions of these algorithms, `"circle"`, `"circle2"` and `"circle3"` can be used. To split the layout algorithm for structural and measurement models, the `layoutSplit` argument can be used. Finally, the `layout` argument can also be used to manually define the placement of nodes (see package documentation for examples). Figure 11.6 shows the result of these algorithms on example 2.25 from the MPlus user's guide (Muthén & Muthén, 1998–2012).

11.4 Conclusion

The semPlot package extends many popular SEM software packages with advanced visualization functions. These functions can be used to display specified models, parameter estimates, model constraints, and implied correlation structures. Furthermore, semPlot provides a bridge between these software packages and different modeling frameworks. The package uses several novel algorithms for automatic placement of variables in the path diagrams and allows for detailed manual customizations[2].

semPlot is sufficiently user-friendly to be used by researchers with limited experience in R, while it presents more advanced users with a broad scope of functionality and flexibility. Several features are open to further development. First, the use of semPlot key be extended in various ways—such as though web interfaces (RStudio & Inc., 2013). Second, support is to be added for additional SEM software packages such as Amos (Arbuckle, 2010), EQS (Bentler & Wu, 1993, using the REQS R package; Mair & Wu, 2012) and R packages semPLS (Monecke & Leisch, 2012) and lava (Holst & Budtz-Joergensen, 2013). The developmental version of semPlot is available at GitHub, `http://github.com/SachaEpskamp/semPlot`, where new ideas for the package can also be submitted.

[2]See for detailed instruction the package website: http://sachaepskamp.com/semPlot