



Mplus Trees: Structural Equation Model Trees Using Mplus

Sarfaraz Serang, Ross Jacobucci, Gabriela Stegmann, Andreas M. Brandmaier, Demi Culianos & Kevin J. Grimm

To cite this article: Sarfaraz Serang, Ross Jacobucci, Gabriela Stegmann, Andreas M. Brandmaier, Demi Culianos & Kevin J. Grimm (2021) *Mplus Trees: Structural Equation Model Trees Using Mplus*, *Structural Equation Modeling: A Multidisciplinary Journal*, 28:1, 127-137, DOI: [10.1080/10705511.2020.1726179](https://doi.org/10.1080/10705511.2020.1726179)

To link to this article: <https://doi.org/10.1080/10705511.2020.1726179>



Published online: 19 Feb 2020.



Submit your article to this journal [↗](#)



Article views: 1129



View related articles [↗](#)



View Crossmark data [↗](#)





Citing articles: 6 View citing articles [↗](#)

TEACHER'S CORNER



Mplus Trees: Structural Equation Model Trees Using Mplus

Sarfaraz Serang¹, Ross Jacobucci², Gabriela Stegmann³, Andreas M. Brandmaier ^{4,5}, Demi Culianos¹, and Kevin J. Grimm ³

¹Utah State University; ²University of Notre Dame; ³Arizona State University; ⁴Max Planck Institute for Human Development; ⁵Max Planck UCL Centre for Computational Psychiatry and Ageing Research

ABSTRACT

Structural equation model trees (SEM Trees) allow for the construction of decision trees with structural equation models fit in each of the nodes. Based on covariate information, SEM Trees can be used to create distinct subgroups containing individuals with similar parameter estimates. Currently, the structural equation modeling component of SEM Trees is implemented in the R packages *OpenMx* and *lavaan*. We extend SEM Trees so that the models can be fit in *Mplus*, in the hopes that its efficiency and accessibility allow a broader group of researchers to fit a wider range of models. We discuss the *Mplus* Trees algorithm, its implementation, and its position among the growing number of tree-based methods in psychological research. We also provide several examples using publicly available data to illustrate how *Mplus* Trees can be implemented in practice with the R package *MplusTrees*.

KEYWORDS

Structural equation modeling; decision trees; data mining; machine learning

Understanding heterogeneity within a population is an important endeavor in the social sciences. As more intricate methods for modeling subpopulations develop, so too do our approaches for modeling the heterogeneity within them. With the introduction of multiple-group structural equation modeling (SEM) by Jöreskog (1971), we became better able to formally describe and understand differences between observable groups of people within a population. Multiple-group models, however, require that the groups be explicitly defined in advance of analysis. An alternative approach is that of decision trees, popularized by the seminal work of Breiman, Friedman, Olshen, and Stone (1984), but with roots in the automatic interaction detection framework of Sonquist and Morgan (1964). Unlike multiple-group models where the groups are constructed by a theory-driven choice, decision trees use a data-driven approach to empirically construct groups based on patterns in one or more covariates.

Decision trees use recursive partitioning to repeatedly divide a covariate space into non-overlapping regions orthogonal to the axes of the covariate space, such that individuals within the resulting regions are as similar as possible with regard to the outcome of interest while the difference between individuals from different regions is maximized. Since finding the optimal tree is a combinatorial hard problem, trees are typically grown by applying heuristics. This is usually done by testing all possible binary splits of all predictors in the covariate set and choosing the split that makes the outcome(s) in each of the two nodes (groups) resulting from that split as homogeneous as possible. These two daughter nodes are then subjected to this same splitting procedure, and so on, until a full tree is grown.

An example of a decision tree is provided in the left portion of Figure 1, while the right portion depicts a visualization of the

recursively partitioned covariate space. The first split is made on X_1 such that values of $X_1 \leq 1$ are in the left branch and values of $X_1 > 1$ are in the right branch. The left branch is then split on X_2 , such that $X_2 \leq 3$ leads to a new left branch and $X_2 > 3$ forms a new right branch. For those in the original right branch (those with values $X_1 > 1$), this region is split such that those with $X_1 \leq 2$ form the new left branch of this subtree and those with $X_1 > 2$ form the new right branch. In this way, the region R_1 contains individuals who have values of $X_1 \leq 1$ and $X_2 \leq 3$, R_2 has those with $X_1 \leq 1$ and $X_2 > 3$, R_3 has those with values of $1 < X_1 \leq 2$, and R_4 contains individuals with $X_1 > 2$. Decision trees can be read as a set of rules that group together people who are similar on some outcome measure. Figure 1 illustrates two key points: (1) once a partition is created it can only be further subdivided, and (2) a variable can be reused to create additional partitions further down the tree (as in the case of X_1). For further resources regarding decision trees, we refer readers to Strobl, Malley, and Tutz (2009), King and Resick (2014), and for a more comprehensive introduction, to James, Witten, Hastie, and Tibshirani (2013).

Tree-based methods in psychological research

Tree-based methods have become an increasingly popular approach for addressing longstanding problems in psychological research. One such problem is differential item functioning (DIF). DIF occurs when items have different characteristics when presented to some groups of people relative to others. For example, if an item is more difficult for females than males when accounting for the level of the construct, the item would be considered to exhibit DIF. It is clearly of great importance to be able to identify items that exhibit DIF, as most situations call for items that assess individuals in the same way irrespective of

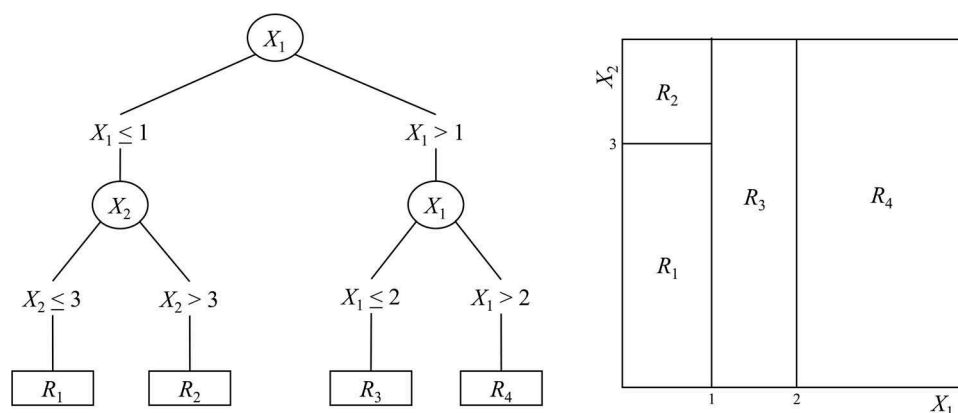


Figure 1. An example of a decision tree (Left) and the covariate space it partitions (Right).

group membership. One popular way to identify DIF is to specify groups beforehand and test whether items perform differently for each group. In practice, DIF is typically evaluated based on demographics such as gender and ethnicity, but combinations of variables and groups based on continuous variables are not usually considered. Furthermore, conventional approaches to DIF require *a priori* knowledge of the optimal way to specify the grouping structure, despite this not being possible to know in practice as there are a near infinite number of ways to define groups such that items may perform differently in one group relative to another.

Decision trees provide a solution to DIF by testing various grouping structures based on combinations of variables. Several methods have recently been developed to apply decision trees in search of DIF. The *Rasch tree* method (Strobl, Kopf, & Zeileis, 2015), which allows decision trees to be applied to Rasch models, proceeds as follows. First, item parameters are estimated for the full sample. Next, the stability of item parameters is assessed with respect to each covariate via structural change tests (see, e.g. Zeileis & Hornik, 2007). If there is significant instability in item parameters (indicating DIF), a split is made in the tree at the value of the covariate that improves model fit most. This procedure is repeated until no further significant splits can be made or the sample size in the nodes becomes too small to split.

One limitation of Rasch trees is that the recursive partitioning algorithm is applied at a global level, treating all items simultaneously. To address this, Tutz and Berger (2016) proposed *item-focussed trees*, which evaluate DIF in Rasch models at the item level. This method identifies the items responsible for the observed DIF and grows a decision tree for each of these items to better understand the grouping structure behind the DIF. Trees are grown in a manner similar to Rasch trees, except that permutation tests are used in lieu of structural change tests to determine whether a split should be made. Extensions have been developed to accommodate models for polytomous items such as the partial credit model for both Rasch trees (Komboz, Strobl, & Zeileis, 2018) and item-focussed trees (Bollman, Berger, & Tutz, 2018).

Tree-based methods have also been used in the context of longitudinal models. For example, *nonlinear longitudinal*

recursive partitioning (Stegmann, Jacobucci, Serang, & Grimm, 2018) applies recursive partitioning to longitudinal mixed-effects models. Person-level covariates are used to grow trees based on growth trajectories, which can take on linear or nonlinear functional forms. In this approach, the user specifies the functional form of a longitudinal model and a set of covariates. The algorithm then uses recursive partitioning to split the data into nodes based on a function of the log-likelihood and the number of nodes. The resulting nodes then represent the expected growth trajectory for individuals within that node. This allows for the comparison of parameter estimates for different subgroups within the data.

SEM Trees

The tree-based approaches discussed thus far have some limitations. Rasch trees, item-focussed trees, and nonlinear longitudinal recursive partitioning are all designed for a specific type of model, whether it be an item response model or longitudinal model. A more general approach can be found in *structural equation model trees* (SEM Trees; Brandmaier, von Oertzen, McArdle, & Lindenberger, 2013). SEM Trees represent a fusion of SEM and decision trees, blending theory-based and data-driven modeling approaches. Essentially, the user specifies a structural equation model and a set of covariates, and a decision tree is generated with that model fit at each node. This allows for the comparison of parameter estimates across terminal nodes to understand how each group differs relative to the others on these estimates. In this way, SEM Trees possess the benefits of exploratory analysis via the decision tree component while also allowing for the accommodation of a theoretical framework, via the SEM piece.

The SEM Tree approach has an advantage over the previously discussed methods in that a structural equation model is fit at each node. This permits greater flexibility in the types of models that can be specified and, as a result, the research questions that can be asked. The decision tree component of SEM Trees provides explicit well-defined groups that can be of great practical interest. For example, the method can be used to show that younger low-income males benefit from a treatment more so than older high-income males, a conclusion that would be

challenging to reach from a regression perspective due to the interaction of several variables (especially if there are a large number of potential covariates). Additionally, such multivariable interactions need not be specified in advance, allowing the potential discovery of a new subgroup of people for whom, for example, certain items do not load as highly onto a given factor as expected.

Because the method we propose in this paper is related to SEM Trees, we discuss the SEM Trees algorithm in detail. To begin, the structural equation models fit using SEM Trees are estimated in the usual way. The case-wise minus two log-likelihood for a structural equation model fit using full information maximum likelihood can be written as

$$-2\log L_i = K_i + \log|\Sigma_i| + (\mathbf{x}_i - \boldsymbol{\mu}_i)' \Sigma_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (1)$$

where for individual i , K_i is a constant depending on the number of complete-data points, \mathbf{x}_i is the vector of complete data, Σ_i is the expected covariance matrix, and $\boldsymbol{\mu}_i$ is the expected mean vector. The minus two log-likelihood (a.k.a. deviance) for the full sample of N individuals can be obtained by taking the sum over the individual log-likelihoods, so that

$$-2\log L = \sum_{i=1}^N (-2\log L_i) \quad (2)$$

Parameters are estimated such that they minimize this quantity (i.e., maximize the likelihood of the parameters under the model).

For each covariate, the data is split along all possible splits of that covariate. For an ordinal or continuous covariate with J possible response options, there are $J-1$ possible splits for this covariate. For a categorical covariate with J observed levels, there are $2^{J-1} - 1$ possible splits that lead to non-empty partitions. For each candidate split, we can obtain two $-2\log L$ values. The first is the $-2\log L$ for the full data at that node, while the second is the $-2\log L$ of a two-group model with the data partitioned according to the split. It can easily be seen that the model fit to the full data is nested within the model fit to the partitioned data because the former is a constrained case of the latter as in a multiple-group model. Thus, these two models can be compared via a likelihood ratio test (LRT). The partition of the covariate that results in the greatest improvement in model fit is retained and the process is repeated in each daughter node. This process continues recursively until a stopping criterion is reached (e.g., a non-significant LRT, a maximal tree depth, or a minimal number of cases in a node).

SEM Trees is currently implemented in R (R Core Team, 2018), via the *semTree* package (Brandmaier & Prindle, 2018). The package uses other R packages to perform all SEM-related operations (such as model specification and parameter estimation), and then implements the decision tree component to grow trees. The primary SEM package for SEM Trees is the *OpenMx* package (Neale et al., 2016), which allows for

very flexible specification of structural equation models. However, the need to fully specify the entirety of the model with limited defaults can present a challenge for casual users.¹ An alternative is to use the *lavaan* package (Rosseel, 2012), where models tend to be much easier to specify. Additionally, the model syntax of *lavaan* as well as its output are very similar to that of *Mplus* (Muthén & Muthén, 1998-2015), making it more accessible to *Mplus* users. However, *lavaan* is currently more limited in the scope of models it can fit, compared to *Mplus*. The goal of the current work is not to replace the *semTree* package, but instead to capitalize on *Mplus*' ability to easily specify a greater variety of models.

The purpose of this article is to introduce an implementation of the SEM Tree approach in *Mplus* which we call *Mplus Trees*. We describe our proposed approach, how it works, and the details of its software implementation. We then conduct a small simulation study and apply the method to three empirical datasets to demonstrate how it can be used in applied contexts. Finally, we discuss the place of *Mplus Trees* in the context of tree-based methods in psychological research, as well as implications for future research.

Mplus Trees

We begin our description of *Mplus Trees* with the algorithm. Consider a structural equation model, M , specified by the user in advance. The algorithm begins by fitting M to the full dataset \mathbf{D} , an $N \times (P + C)$ matrix with N rows (one for each individual) and $(P + C)$ columns, where P of the columns holds observed variables explicitly modeled in M and the remaining C columns contain covariates to be used in constructing the decision tree. Thus, \mathbf{D} can be decomposed into two submatrices, one with variables modeled in M , $\mathbf{D}_{N \times P}$, and another with covariates not modeled in M , $\mathbf{D}_{N \times C}$. Consider a covariate c with J observed response options. Let us, for simplicity, assume that c is an ordinal variable.² To examine c for a possible split at the first response option, $j = 1$, the procedure can divide $\mathbf{D}_{N \times P}$ into two submatrices: those individuals with $j = 1$ (\mathbf{D}_1) and those with $j > 1$ (\mathbf{D}_2). For $j = 2$, $\mathbf{D}_{N \times P}$ would be divided into individuals with $j \leq 2$ (\mathbf{D}_1) and those with $j > 2$ (\mathbf{D}_2). This process can be repeated for all $J-1$ possible splits on c . The *Mplus Trees* algorithm proceeds by fitting M to each of the $J-1$ pairs of \mathbf{D}_1 and \mathbf{D}_2 .

This procedure is repeated for all possible splits on all covariates in $\mathbf{D}_{N \times C}$. The split that produces the lowest possible value of the $-2\log L$ is put forth as a candidate split. Whether the candidate split is actually made depends on user-specified criteria. A number of possible criteria are available, which we describe in greater detail in the section discussing implementation. If the criteria are not met, no split is made, and the current node becomes a terminal node. If the criteria are met, a split is made. This entails creating two daughter nodes, consisting of data subsets \mathbf{D}_1 and \mathbf{D}_2 from the relevant split. Each of these daughter nodes is treated as a new dataset and is

¹We note however that *Onyx* (von Oertzen, Brandmaier, & Tsang, 2015) is a visual modeling tool that allows the automatic generation of *OpenMx* code from path diagrams.

²Categorical covariates can be handled similarly, either by breaking them up into a set of dichotomous indicator variables, or by examining all possible combinations of groups that result in individuals on either side of the split.

recursively subjected to the same splitting procedure. This process continues until no further splits are retained because all nodes are considered terminal nodes.

As a result of the tree-building process, each of the N individuals will be placed in the terminal node corresponding to the values observed on their C covariates. This results in a number of useful properties. For example, the tree itself can be read as a map directing the placement of individuals into terminal nodes according to the rules defined by the covariate splits. Additionally, since each terminal node contains different subsets of individuals, the estimates of the parameters expressed in M resulting from fitting M to each node will be distinct. The expected parameter estimates for a given pattern of covariates are thus the parameter estimates produced by fitting M to the appropriate terminal node. The $-2\log L$ for the full tree can also be obtained by summing the $-2\log L$ values in each terminal node.

Software implementation

The *Mplus* Trees algorithm is implemented in the R package *MplusTrees*, which requires the *Mplus* program to be installed. The package weaves together several packages in order to construct the decision trees. First, the user must specify an *Mplus* model in R using the R package *MplusAutomation* (Hallquist & Wiley, 2018), which serves as an interface that allows R to communicate with *Mplus*. The model is sent to *Mplus*, *Mplus* fits the model, and the relevant results are extracted from the *Mplus* output files and read back into R. The construction of the trees is performed in R by the *rpart* package (Therneau & Atkinson, 2018), a package for recursive partitioning. In short, *Mplus* fits the models, *rpart* builds the trees, and *MplusAutomation* connects R to *Mplus*. The *MplusTrees* package coordinates this activity as well as organizes and presents the results, streamlining the entire process.

The primary function of the *MplusTrees* package is named *MplusTrees()*. As input, this function takes the *Mplus* script written in *MplusAutomation* form, the dataset as an R data frame, and a set of covariates upon which to split. An argument also exists to determine the conditions that control the splitting procedure. These conditions are defined by the *rpart.control()* function in the *rpart* package. The primary argument of this function is the complexity parameter, denoted cp . In general, the complexity parameter determines the relative improvement in model fit in order for the split to be retained. The *rpart* algorithm stems from the algorithmic modeling culture (Breiman, 2001), in which model selection is not based on statistical significance but on prediction accuracy. In this context, cp corresponds to the proportional improvement in the $-2\log L$ needed to perform a split. Thus, if a candidate split improves upon $-2\log L_{root}$, the $-2\log L$ of the root node (the node containing the full sample before any splits are made), by $cp \cdot (-2\log L_{root})$, it is made. Otherwise, the node becomes a terminal node. For example, if the $-2\log L$ of the root node was 1,000 and cp was .01, the candidate split would need to improve the $-2\log L$ by 10 for that split to be retained. Other criteria can also be specified in

addition to cp , including the minimum number of observations within a node needed to attempt a split, the minimum observations within a terminal node, the maximum depth of any node in the tree, and whether/how to use surrogate splits if covariate information is missing.

Using the proportional change in $-2\log L$ is advantageous in that it can be calculated quite easily and is available for all models for which *Mplus* produces the log-likelihood (e.g. when maximum likelihood estimation is used). We also choose it for practical reasons, based on the contexts in which we believe *Mplus* Trees can provide greater utility than other approaches discussed. Specifically, the relative ease with which one can specify and fit a relatively complex model (e.g. a multilevel structural equation model with random intercepts and slopes, a growth mixture model, etc.) in *Mplus* makes *Mplus* Trees a more attractive option when fitting trees to these models. However, complex models typically require greater sample sizes, and approaches based on p values can split based on trivial differences in large samples, leading to very large trees. The proportional change in $-2\log L$ can be conceptualized as a heuristic for relative effect size, in contrast to the type I error rate, α , which is a heuristic for statistical significance.

The output from the *MplusTrees()* function provides the tree structure as well as the model estimates at each terminal node. With regard to the tree structure, the output describes each node, including the splitting criterion (the variable that is split and the cutpoint at which the split is made), the number of individuals within that node, and whether the node is a terminal node. For each terminal node, the output gives the SEM parameter estimates (extracted from *Mplus*) of the model fit to the individuals within that node. It also provides the final terminal node location for each individual used to construct the tree.

Practical guidelines

The challenge that comes in using proportional changes in log-likelihood as a splitting criterion is that no conventional standard exists for this metric (as opposed to a threshold such as $p < .05$). This is further complicated by the fact that the log-likelihood varies with sample size, and so it is unclear whether an objective standard is even achievable. Ideally, the cp parameter would be chosen via a procedure such as cross-validation. Though this feature is available in the *MplusTrees* package, it is likely that its application will be impractical for most users, given limitations further elucidated in the discussion. However, we encourage the user to at least try several values of the cp parameter (e.g. .01, .001, etc.) in order to determine the depth of tree that is most suitable in the context of the problem they are trying to solve. The cp parameter directly governs the depth of the tree. Smaller values of cp lead to more splits, larger values of cp lead to fewer splits. We leave it to the user to determine how many splits are practically useful given their goals, data, and model. We believe this recommendation to be consistent with the exploratory nature of decision trees as a whole, and hope that it encourages users to think more

carefully about the tree-construction process. We also remind users that the chosen value of cp must be defensible in context.

For those concerned about using a cp value that is too small (i.e., too liberal), the p value from the LRT (as in SEM Trees) can also be used as an additional splitting criterion. When this option is used, the split must meet both criteria (a p value less than α and a proportional change in $-2\log L$ greater than cp) for the split to be retained. This can be especially useful in smaller samples, since the $-2\log L$ of the root node would be smaller, potentially causing small changes in the $-2\log L$ to lead to splits. The use of the LRT can help protect against spurious splits due to this phenomenon. It can also help reduce overfitting due to sampling variability, the negative effects of which are magnified in smaller samples.

As mentioned in the previous section, the SEM part of the algorithm can be computationally burdensome. The decision tree algorithm performs an exhaustive search upon all possible subsets of data created by all possible splits of the covariates. Even if it takes only a few seconds to fit M , if M must be fit several thousand times, this can quickly add up. If M is especially complex and takes a nontrivial amount of time to fit even once, this can lead to extended computation times. As such, there are a number of options available in *MplusTrees* to shorten the computation time as much as possible.

One such feature is parallelization. *Mplus* allows for the use of multiple processors using the PROCESSORS option, which can be added into the model script in R, potentially decreasing the time it takes to fit M . Another option deals with how *MplusTrees* treats categorical (nominal) covariates. As mentioned above, a categorical covariate with J observed response options has $2^{J-1} - 1$ possible splits. If J is large, an option exists to convert the categorical variable into J indicator variables, one for each response option. This reduces the computational time from $O(2^J)$ to $O(J)$. If splitting on continuous covariates with a large number of different observed responses, we recommend first rounding these covariates or placing them in bins before using them to build trees. In such cases, we find that the gain achieved in finding the exact cutpoint is rarely worth the computational time spent evaluating each possible partition, not to mention that practical recommendations are typically made using rounded numbers anyway.

Simulation study

To investigate the performance of *Mplus Trees*, we performed a small simulation study. Data from $N = 2,000$ individuals were generated using *lavaan* from a two-factor confirmatory factor analysis (CFA) model, with 4 items per factor. Factor and uniqueness variances were simulated to be 1, and the covariance between factors was simulated to be 0.2. Only the loadings differed across people. Loadings were simulated to be equal for each item *within* a person, but the values these loadings took on differed *between* people, depending on their terminal node. Three dichotomous covariates were also simulated, X_1 , X_2 , and X_3 .

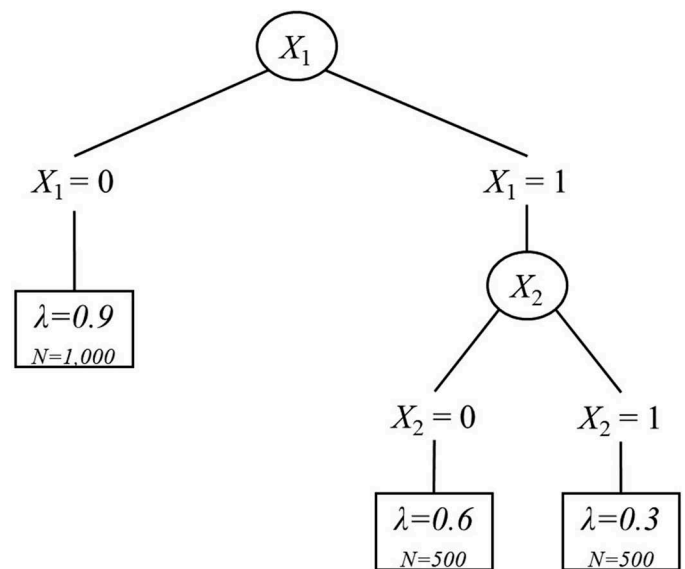


Figure 2. Simulation study population tree structure.

The population tree structure can be found in Figure 2. Individuals in the first terminal node all had large factor loadings (0.9). Those in the second terminal node had medium factor loadings (0.6), and those in the third terminal node had small factor loadings (0.3). Values of the covariates were assigned to achieve the tree in Figure 2. Since the first split was to be made on X_1 , all those with loadings of 0.9 had values of $X_1 = 0$, and those with loadings of 0.6 and 0.3 had values of $X_1 = 1$. The second split was to be made between those with loadings of 0.6 and 0.3, so those with loadings of 0.6 had values of $X_2 = 0$, and those with loadings of 0.3 had values of $X_2 = 1$. Those with loadings of 0.9 were assigned values of 0 and 1 for X_2 with equal probability, since no further splits should be made for these individuals. X_3 was assigned in a similar way, since no splits were to be made on X_3 ; it was simply a “noise” covariate.

The two-factor CFA was fit to the data, with factor variances constrained to 1 and all loadings and uniqueness variances freely estimated. At each node, a minimum of $N = 200$ individuals was required to attempt a split, and each terminal node was required to have a minimum sample size of $N = 200$ as well. The cp parameter was set to .001. This process was repeated 1,000 times, yielding 1,000 replications.

The true population tree structure was correctly recovered in 95.5% of replications. Parameter estimates were fairly close to their population values. For those in the group with large loadings, the average estimated loading was 0.899 (vs. 0.9). The average covariance between factors was 0.200 (vs. 0.2), and the average uniqueness variance was 0.997 (vs. 1). For those with medium loadings, the average estimated loading was 0.600 (vs. 0.6), the average covariance between factors was 0.201 (vs. 0.2), and the average uniqueness variance was 0.994 (vs. 1). For those in the group with small loadings, the average loading was 0.309 (vs. 0.3), the average covariance was 0.200 (vs. 0.2), and the average uniqueness variance was 0.921 (vs. 1). Of all the parameter estimates, only the uniqueness variance for individuals with loadings of 0.3 seemed to deviate slightly from the population value. However, most parameter

estimates were unbiased, with values nearly identical to the population values. This demonstrates that *Mplus* Trees can accurately uncover both the tree structure and parameter estimates within terminal nodes.

Illustrative examples

In this section, we present three examples with empirical data to demonstrate how *Mplus* Trees can be used in practice. We simultaneously use this opportunity to present how the *MplusTrees* package can be used, including sample code. The datasets used are publicly available, so that the user may reproduce these examples. All R code is contained in the Appendix. We note that all splits found using the given values of *cp* were retained when also using the LRT criterion, allowing us to place more faith in the results.

Example 1

In our first example, we use data from Holzinger and Swineford (1939) contained in the *lavaan* package. The data consist of mental ability scores for 301 seventh and eighth grade students from two different schools (see the documentation in the *lavaan* package for further details). The nine items of interest have been well studied, and are known to follow a three-factor model, with three items per factor. The first three items (visual perception, cubes, lozenges) are thought to load onto a visualization factor, the next three items (paragraph comprehension, sentence completion, word meaning) load onto a verbal ability factor, and the final three items (speeded addition, counting of dots, discrimination of capitals) load onto a speed factor.

An *Mplus* script is written in R using the *mplusObject*() function from the *MplusAutomation* package to specify a three-factor CFA model, with R code below.

```
script1 = mplusObject()
TITLE = "Example #1 - Factor Model;",
MODEL = "f1 BY x1-x3; f2 BY x4-x6; f3 BY x7-
x9;",
usevariables = c('x1','x2','x3','x4','x5',
'x6','x7','x8','x9'),
rdata = HolzingerSwineford1939)
```

The script contains the usual elements of a *Mplus* script. The *TITLE* and *MODEL* statements correspond to what they would in *Mplus*. The *MODEL* statement can, of course, be written over multiple lines, though we have placed it on a single line here to conserve space. The *usevariables* statement tells R which variables to use in the dataset, and *rdata* tells R which dataset to use. Next is the *MplusTrees*() function, with R code below.

```
fit.ex1 = MplusTrees(script1, Holzinger
Swineford1939, group=~id, rPartFormula
=~sex+school+grade, control=rpart.con-
trol(minsplit=100, minbucket=100, cp=
.01))
```

This function takes as arguments the script as well as the dataset. The *group* argument, beginning with a tilde, specifies the name of an identification (ID) variable in the dataset for each individual. This allows the user to identify the terminal node in which each individual is placed. If one is not provided, the program creates ID numbers based on each individual's row number. The *rPartFormula* denotes which variables are to be used as covariates upon which splits should be considered. In our case, we use *sex*, *school*, and *grade*, all of which are dichotomous.

The last piece is the *control* argument, which sets some control parameters via *rpart.control()*. These parameters have some initial default values (which are described in its documentation), but can be changed. The first control parameter worth changing is the *minsplit* parameter, the minimum number of observations in a node in order to attempt a split. The second is the *minbucket* parameter, the minimum number of observations in a terminal node. These should be set to the minimum sample size required to fit the structural equation model. By default, *minsplit* is set to 20, and *minbucket* is set to 7. These sample sizes are clearly too small to fit our CFA, so we have set these values to 100. This number depends on the model and the data, but we note that raising it can save computation time by keeping *MplusTrees* from trying to fit models when samples are too small.

The most important control parameter is the *cp* parameter, which we set at .01 for this example. With *minsplit* and *minbucket* set to 100, this produced the same results as a *cp* value of .001 and smaller. When removing the *minsplit* and *minbucket* restrictions, smaller values such as .001 led to terminal nodes containing too few individuals (30) to yield trustworthy estimates when fitting a three-factor CFA. We encourage users to try multiple values of *cp* (e.g. .01, .001, etc.) and select one that produces the tree with the most practical utility. For example, a tree with dozens of terminal nodes is hardly interpretable, defeating the purpose of building it. The *cp* value can be chosen by using *k*-fold cross-validation (simply set the argument *cv* = T), but this approach has some practical limitations, described in the discussion.

The output provides several pieces of information, including the tree structure, terminal nodes for each individual, and parameter estimates. This example's tree is given below.

```
(node), split, n, deviance, yval
* denotes terminal node
1) root 301 7475.490 7475.490
2) school=Grant-White 145 3469.778 3469.
778 *
3) school=Pasteur 156 3894.618 3894.618 *
```

The first two lines provide a key with which to read the tree, denoted by the numbered lines. The first line of the tree corresponds to the root node before any splits, containing the full sample of 301 individuals, with a $-2\log L$ (deviance) of 7475.490. Since the deviance is the outcome in each node, the "deviance" and "yval" will be identical. The second and third lines show that the algorithm chose to make only a single

split. This split was made on school, with the 145 students in the Grant-White school falling in the second node and the 156 students in the Pasteur school in the third node. The $-2\log L$ value for each of these nodes is also given. The asterisks at the end of these lines show that nodes 2 and 3 are terminal nodes.

Though not explicitly presented, the fitted model object also contains information on which nodes are terminal nodes and the terminal node in which each individual is placed. Parameter estimates for the structural equation model are also given for each terminal node. These are the estimates calculated by *Mplus*, and in this case, they would be identical to the estimates from a multiple-group model where *school* was the grouping variable. The factor loadings for each group are presented in Table 1. The largest differences between these schools seem to be in the estimated loadings of the first factor. We emphasize here that standard errors and Wald p values are strictly valid only if the model is specified a priori instead of data-adaptive and if it is correctly specified. Thus, when using *Mplus Trees* we advise against reporting and interpreting these standard errors and p values (see, e.g. Serang, Jacobucci, Brimhall, & Grimm, 2017, as well as Serang & Jacobucci, 2020, for use of this same logic in the context of regularization).

Example 2

Our second example uses data from the National Longitudinal Survey of Youth (Bureau of Labor Statistics, 2015) to demonstrate trees with longitudinal count outcomes. The sample comes from the 1997 cohort, and contains 796 individuals measured on 10 variables: five outcomes and five covariates for splitting.³ To simplify the example, we only analyzed individuals with complete data, but this is not a requirement as *MplusTrees* easily accommodates missing data in both the model variables as well as the covariates. The outcome was the number of cigarettes the participant usually smoked per day within the last 30 days (all participants in the selected sample admitted to smoking since the last interview). The outcome was measured annually from 1998–2002, yielding five repeated observations. The five covariates were time-invariant, all measured in 1997. These included two dichotomous covariates: sex and whether the participant ever had an alcoholic beverage. Three peer-related questionnaire items

were also considered as covariates: the percent of peers who smoked, got drunk at least once a month, and used illegal drugs. The peer variables had five possible response options: less than 10%, about 25%, about 50%, about 75%, and above 90%.

The count data were modeled using a latent growth curve model with a linear basis (Meredith & Tisak, 1990). This problem is unique in that the number of cigarettes smoked per day is a count variable, and as such is best analyzed using a Poisson residual structure model with a log link function. This can be done with a single line of code in *Mplus*, but would be quite challenging to do in *OpenMx* and cannot currently be done at all in *lavaan*. As such, this example highlights the advantages *MplusTrees* possesses in using *Mplus* for estimation.

Using a cp value of .001, only one split was made based on the covariate “peers who used illegal drugs.” Those who had almost no peers (<10%) who used illegal drugs ($N = 161$) were determined to have different change patterns in the number of cigarettes they smoked compared to those who had some peers (25% or more) who did use illegal drugs ($N = 635$). Parameter estimates are provided in Table 2. The corresponding decision tree is given in Figure 3, and the average number of cigarettes smoked per day for each group is provided in Figure 4.

Figure 4 illustrates that those with fewer peers who used illegal drugs smoked on average fewer cigarettes per day at the start of the study, but increased their smoking behavior more than their counterparts. This is also evident in the model

Table 2. Example 2 results – parameter estimates for latent growth curve model with count outcome fit to smoking data.

	<10% peers use illegal drugs	>25% peers use illegal drugs
Intercept mean	1.652	1.987
Slope mean	0.173	0.087
Intercept variance	0.834	0.642
Slope variance	0.067	0.030
Intercept-slope covariance	-0.162	-0.076

Table 1. Example 1 results – factor loadings for CFA model fit to Holzinger-Swineford data.

Factor	Loading	Grant-White school	Pasteur school
Visualization	X1	1.000	1.000
	X2	0.736	0.394
	X3	0.925	0.570
Verbal ability	X4	1.000	1.000
	X5	0.990	1.183
	X6	0.963	0.875
Speed	X7	1.000	1.000
	X8	1.226	1.125
	X9	1.058	0.922

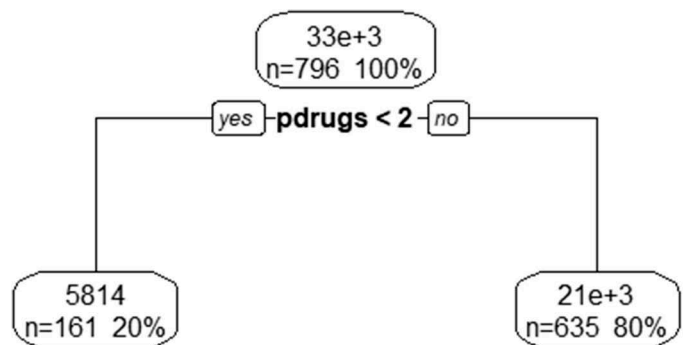


Figure 3. Decision tree for latent growth curve model in Example 2. Covariate-based decisions are in bold face. Decision paths are labeled “yes” or “no” based on the criterion. Each node contains three values. The bottom left value is the sample size of the node. The bottom right value is the sample size relative to the root node. The top value is the deviance ($-2\log L$).

³Downloading the dataset with these 10 variables and the “id” variable from the NLSY website and deleting all individuals with any missing data should yield the exact same dataset we used for this analysis.

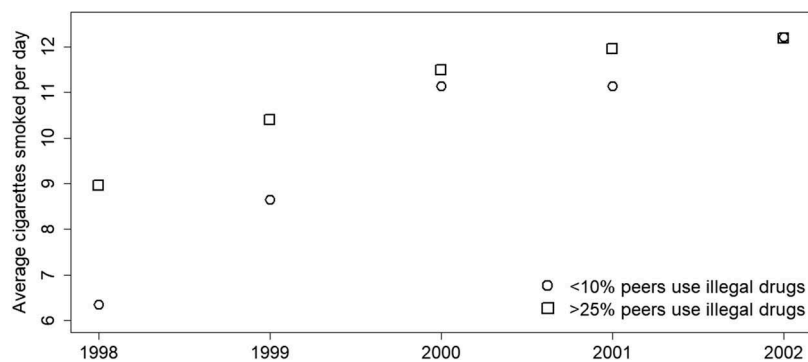


Figure 4. Average number of cigarettes smoked per day for each group in Example 2.

parameters, as those with fewer illegal drug-using peers had a lower intercept (1.652 vs. 1.987) but a higher rate of change (0.173 vs. 0.087) relative to the other group. Since both groups smoked a similar number of cigarettes per day by the fifth wave, multiple explanations could be plausible. One possibility is a pseudo-ceiling effect where exposure to more peers who used illegal drugs accelerated the smoking behavior of these individuals before the first wave of data analyzed, and the other group simply caught up. Evidence for this can be seen in the negative intercept-slope covariances, which indicate that individuals who smoked fewer cigarettes per day at the start of the study changed more. This covariance was larger for those with fewer illegal drug-using peers (-0.162 vs. -0.076), consistent with a ceiling effect given that this group had a lower intercept. Another possibility is that those who had more peers who used illegal drugs were more likely to transition into using illegal drugs at the expense of smoking cigarettes.

Example 3

Our final example applies recursive partitioning to survival analysis, resulting in survival trees. Although the concept of a survival tree is not new (see, e.g. Aichele, Rabbitt, & Ghisletta, 2016; Ghisletta, 2014; Zhou & McArdle, 2015), the implementations are usually specific to survival models alone, much like other tree-based methods we have discussed. The ease of fitting a survival model in *Mplus* makes the *Mplus* Trees framework a convenient approach for these kinds of models, especially since they can, as with other models we have discussed, be very challenging to fit using *OpenMx* and currently impossible to fit using *lavaan*.

The data for this example also come from the 1997 cohort of the National Longitudinal Survey of Youth, with all potential predictor variables measured in 1997. We are interested in whether individuals have ever smoked a cigarette, and the extent to which this can be predicted by the nature of the participant's home environment, as operationalized by the Family/Home Risk Index, a 21-point scale for which higher scores indicate higher risk environments. For those unfamiliar with survival analysis, the feature distinguishing it from logistic regression is our interest in a specific event, in our case, smoking, and the time at which this event first occurs. An important characteristic of this

kind of data is that the event is irreversible; once an individual has smoked a cigarette, they cannot then transition back into the former state of never having smoked. In our sample of 4,548 participants (all under the age of 17), 1,443 (31.7%) smoked a cigarette, and for these individuals, we know the age at which this event occurred. The smoking status for the remaining individuals is *censored* in that we know the last age at which they were observed not yet having smoked a cigarette, but we do not know if or when they do smoke beyond this age. The five covariates considered for splits are the same five as in Example 2: sex, whether the participant had ever had an alcoholic beverage, the percent of peers who smoked, the percent of peers who got drunk at least once a month, and the percent of peers who used illegal drugs. As in the previous example, we use only complete data, though this is not required.

Within each node, a Cox regression model was fit with the score on the Family/Home Risk Index as the predictor. With a *cp* value of .01 one split was made. The regression estimate for those who had never had an alcoholic beverage ($N = 3,084$, $\beta = 0.133$) was larger than those had consumed an alcoholic beverage ($N = 1,464$, $\beta = 0.072$). That is, for those who never had an alcoholic beverage, the hazard of smoking was 14.2% higher per additional point on the Family/Home Risk Index, whereas for those who had consumed an alcoholic beverage, the hazard of smoking was only 7.5% higher per additional point on the Family/Home Risk Index. We can interpret this as a compensatory (negative) interaction between the Risk Index and alcohol usage: for those who have consumed alcohol, the riskiness of the environment is not as influential a factor in smoking behavior compared to those who have not consumed alcohol.

Discussion

The purpose of this paper is to extend the SEM trees developed by Brandmaier et al. (2013) for use with *Mplus*. Our proposed approach, called *Mplus* Trees, accomplishes this in two major ways. First, we use *Mplus* as the SEM engine, allowing users the opportunity to fit a broad range of models using efficient optimization algorithms in a relatively accessible way. Second, we use the proportional change in $-2\log L$ as the primary criterion for splitting. The method is implemented in the freely available R package *MplusTrees* and

several examples have been provided using publicly available empirical datasets to demonstrate how *Mplus* Trees can be applied in practice.

The implementation of this algorithm is relatively straightforward. After loading the *MplusTrees* package, the user specifies an *Mplus* model script using the function `mplusObject()`. The `MplusTrees()` function is then called, with arguments including the model script, dataset, covariates, and control parameters, including the *cp* parameter. Several values of *cp* should be tried, unless the user is using cross-validation in which case this is performed automatically. The output contains the final tree structure, the terminal node for each individual, and the parameter estimates for the model fit in each terminal node.

Mplus Trees occupies a unique position among tree-based methods in psychological research. Compared to approaches such as Rasch trees, item-focussed trees, and nonlinear longitudinal recursive partitioning, *Mplus* Trees is flexible enough to fit a wider range of models. *Mplus* can also model count data, zero-inflated data, complex survey data, survival data, etc., something that can prove challenging (or impossible) to run with current implementations of the SEM Trees algorithm.

Though the conceptual foundations of *Mplus* Trees and SEM Trees are quite similar, the implementation differs. This is especially true with regard to software. SEM Trees fit the structural equation models in *lavaan* and *OpenMx*. The *lavaan* package can fit many commonly used models, but more complex models such as mixture models, models with count outcomes, and general multilevel structural equation models are currently unavailable. Though *OpenMx* can theoretically fit a broader range of models, this can require a level of programming expertise beyond that of the casual user, making these models less accessible to applied researchers in the social sciences. *Mplus* Trees benefits both from the efficiency of *Mplus* as well as its popularity, making it both computationally powerful and accessible to a wide audience.

The two also differ in their splitting criterion. SEM Trees use the LRT to determine whether a split is made, whereas *Mplus* Trees primarily use a threshold based on the proportional improvement in the log-likelihood. A benefit of using proportional improvement in log-likelihood is that it does not require the distributional assumptions made by the LRT. Yuan and colleagues have shown that in models with a large number of variables, the LRT statistic does not follow a chi-square distribution even in large samples with normally distributed data (Yuan, Fan, & Zhao, 2019). This difference in splitting criteria can potentially lead to differences in the structure of the tree, as one approach may lead to a split when the other does not.

Since *Mplus* Trees relies on a user-specified *cp* parameter to determine the depth of the tree, overfitting is a potential concern. By overfitting, we refer to the notion that the tree can become too sample-specific at the expense of generalizability to the population at large. Brandmaier et al. (2013) recommend *k*-fold cross-validation or pruning as methods to ensure generalizability to out-of-bag samples. Although *k*-fold cross-validation is available in the *MplusTrees* package, it is only practical under the following circumstances. First, the model cannot take too long to fit in *Mplus*. Since cross-

validation involves repeatedly fitting (potentially hundreds of) models, if each model takes a nontrivial amount of time to fit, the computational time required to fit models to each fold, for each *cp*, can quickly become prohibitive. Second, cross-validation should currently only be used with a small number of covariates with a small number of response options, also for reasons relating to computation time. Finally, the user must be confident that the model can be fit without issue in samples a fraction of the size of the original. Say, for example, we wished to fit a 1 factor CFA model and believed this required a sample size of $N = 100$ to fit. If we were using 5-fold cross-validation, we would need a sample of at least $N = 500$, since we would need to fit the model in each fold. If we were considering two dichotomous covariates for possible splits, we would need at least $N = 2,000$, since the tree would partition the data into even smaller subsamples. This illustrates that the required sample for even a simple model with only a pair of dichotomous covariates can balloon quite rapidly! As such, though available, we recommend that users employ cross-validation with caution.

Since many users will be unable to perform cross-validation, we recommend they try several values of *cp*. One can start with a smaller (liberal) value, increasing it gradually, in effect pruning the tree. We have also included the LRT as an added criterion for splitting along with *cp*. Based on their properties, the *cp* criterion is more likely to overfit in small samples, whereas the LRT is more likely to overfit in large samples though their exact behavior depends, of course, on the chosen criterion values. As such, we expect that using both criteria together may provide a pragmatic approach for selecting trees with high scientific utility.

We note that our approach inherits the variable selection bias inherent in *rpart*. Given the exhaustive searching for a best split point both within and across variables, those variables with more potential split points (e.g., continuous variables or ordinal variables with many levels) have a higher chance to be selected because they have more split points. As a result, a covariate with a large number of possible splits is more likely to be chosen not (only) because of its importance, but rather because of the large number of partitions available (Brandmaier et al., 2013; Jensen & Cohen, 2000; Strobl, Boulesteix, Zeileis, & Hothorn, 2007). Although a correction for multiple comparisons such as the Bonferroni correction may seem at first a natural solution, Brandmaier et al. (2013) found that this introduces bias into the variable selection, so its use is discouraged. They suggest a “fair selection” approach which splits the dataset in half. One half is used to find the best split value for each covariate, and the second half is used to compare the covariates based on these optimal splits. The authors found this approach to be unbiased in terms of variable selection. However, this approach is not currently available in *MplusTrees* because it relies on *rpart* (which does not implement the approach) to perform the splitting. At this time, we recommend that users encountering this issue reduce the possible response options of their variables by creating substantively meaningful bins (e.g. instead of treating age as continuous, create intervals such as 20–40 years, 40–60 years, etc.).

Finally, we suggest that users make use of the other options offered by `rpart.control()`, especially `minsplit` and `minbucket`. We highly recommend taking this into consideration because the structural equation models themselves have minimum sample size requirements for identification and estimation, and it would seem inappropriate to attempt to further grow the tree if the model cannot be estimated within each daughter node. This can greatly decrease computational time by reducing the number of candidate splits considered.

In addition to the potential future features already discussed, *Mplus* Trees has much room for expansion both as a method and in application. As is well known, decision trees can be unstable due to sampling variability. Random forests grow multiple trees and have gained popularity due to their ability to address this issue. SEM Trees have been extended to SEM Forests (Brandmaier, Prindle, McArdle, & Lindenberger, 2016), and such an extension would seem very natural for *Mplus* Trees as well. On the application side, it can be used to identify groups of people defined by nonlinear interactions of covariates which would otherwise be challenging to find. For example, with *Mplus* Trees one could arrive at the conclusion that there is more baseline variability in the number of occasions of self-disclosure for couples defined by an interaction between four (out of a dozen) covariates. Such a result would be difficult to reach without decision trees, as this would require the specification of at least all possible interaction terms up to the fourth order, which is typically impractical.

Mplus Trees represents a flexible approach to exploratory multiple group modeling to better understand heterogeneity among individuals. It can be used to identify nonlinear interactions between covariates that lead to differences in parameter estimates of the resulting subgroups. Because of its use of *Mplus*, *Mplus* Trees can be used to fit a wide range of models in a relatively accessible way. The approach can be implemented in practice using the *MplusTrees* package, which is freely available in R. We hope that *Mplus* Trees provide researchers with a tool that further assists them in their theory-guided exploration.

Acknowledgments

We thank Bengt Muthén for allowing us to refer to the approach as *Mplus* Trees. The accompanying R package *MplusTrees* is not part of the commercial *Mplus* software and we are solely responsible for maintaining the package.

ORCID

Andreas M. Brandmaier  <http://orcid.org/0000-0001-8765-6982>
Kevin J. Grimm  <http://orcid.org/0000-0002-8576-4469>

References

- Aichele, S., Rabbitt, P., & Ghisletta, P. (2016). Think fast, feel fine, live long: A 29-year study of cognition, health, and survival in middle-aged and older adults. *Psychological Science*, 27, 518–529. doi:10.1177/0956797615626906
- Bollman, S., Berger, M., & Tutz, G. (2018). Item-focused trees for the detection of differential item functioning in partial credit models. *Educational and Psychological Measurement*, 78, 781–804. doi:10.1177/0013164417722179
- Brandmaier, A. M., & Prindle, J. J. (2018). *semtree: Recursive partitioning for structural equation models*. R package version 0.9.13. Retrieved from <https://CRAN.R-project.org/package=semtree>
- Brandmaier, A. M., Prindle, J. J., McArdle, J. J., & Lindenberger, U. (2016). Theory-guided exploration with structural equation model forests. *Psychological Methods*, 21, 566–582. doi:10.1037/met0000090
- Brandmaier, A. M., von Oertzen, T., McArdle, J. J., & Lindenberger, U. (2013). Structural equation model trees. *Psychological Methods*, 18, 71–86. doi:10.1037/a0030001
- Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science*, 16, 199–215. doi:10.1214/ss/1009213726
- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. New York, NY: Chapman & Hall.
- Bureau of Labor Statistics. (2015). *National longitudinal survey of youth 1997 cohort*. [Data file and code book]. Retrieved from <https://www.nlsinfo.org/investigator/pages/search.jsp>
- Ghisletta, P. (2014). Recursive partitioning to study terminal decline in the berlin aging study. In J. J. McArdle & G. Ritschard (Eds.), *Contemporary issues in exploratory data mining in the behavioral sciences* (pp. 405–428). New York, NY: Routledge.
- Hallquist, M., & Wiley, J. (2018). *MplusAutomation: An R package for facilitating large-scale latent variable analyses in Mplus*. *Structural Equation Modeling*, 25, 621–638. doi:10.1080/10705511.2017.1402334
- Holzinger, K. J., & Swineford, F. (1939). A study in factor analysis: The stability of a bi-factor solution. In *Supplementary educational monograph* (Vol. 48). Chicago, IL: University of Chicago Press.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning with applications in R*. New York, NY: Springer. doi:10.1007/978-1-4614-7138-7
- Jensen, D. D., & Cohen, P. R. (2000). Multiple comparisons in induction algorithms. *Machine Learning*, 38, 309–338. doi:10.1023/A:1007631014630
- Jöreskog, K. G. (1971). Simultaneous factor analysis in several populations. *Psychometrika*, 36, 409–426. doi:10.1007/BF02291366
- King, M. W., & Resick, P. A. (2014). Data mining in psychological treatment research: A primer on classification and regression trees. *Journal of Consulting and Clinical Psychology*, 82, 895–905. doi:10.1037/a0035886
- Komboz, B., Strobl, C., & Zeileis, A. (2018). Tree-based global model tests for polytomous Rasch models. *Educational and Psychological Measurement*, 78, 128–166. doi:10.1177/0013164416664394
- Meredith, W., & Tisak, J. (1990). Latent curve analysis. *Psychometrika*, 55, 107–122. doi:10.1007/bf02294746
- Muthén, L. K., & Muthén, B. O. (1998–2015). *Mplus user's guide* (7th ed.). Los Angeles, CA: Muthén & Muthén.
- Neale, M. C., Hunter, M. D., Pritikin, J. N., Zahery, M., Brick, T. R., Kirkpatrick, R. M., ... Boker, S. M. (2016). *OpenMx 2.0: Extended structural equation and statistical modeling*. *Psychometrika*, 81, 535–549. doi:10.1007/s11336-014-9435-8
- R Core Team. (2018). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Rosseel, Y. (2012). *lavaan: An R package for structural equation modeling*. *Journal of Statistical Software*, 48, 1–36. doi:10.18637/jss.v048.i02
- Serang, S., & Jacobucci, R. (2020). Exploratory mediation analysis of dichotomous outcomes via regularization. *Multivariate Behavioral Research*, 55, 69–86. doi:10.1080/00273171.2019.1608145
- Serang, S., Jacobucci, R., Brimhall, K. C., & Grimm, K. J. (2017). Exploratory mediation analysis via regularization. *Structural Equation Modeling*, 24, 733–744. doi:10.1080/10705511.2017.1311775
- Sonquist, J., & Morgan, J. (1964). *The detection of interaction effects: A report on a computer program for the selection of optimal combinations of explanatory variables* (No. 35). Ann Arbor, MI: University of Michigan, Institute for Social Research, Survey Research Center.
- Stegmann, G., Jacobucci, R., Serang, S., & Grimm, K. J. (2018). Recursive partitioning with nonlinear models of change. *Multivariate Behavioral Research*, 53, 559–570. doi:10.1080/00273171.2018.1461602

- Strobl, C., Boulesteix, A.-L., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8, 25. doi:10.1186/1471-2105-8-25
- Strobl, C., Kopf, J., & Zeileis, A. (2015). Rasch trees: A new method for detecting differential item functioning in the Rasch model. *Psychometrika*, 80, 289–316. doi:10.1007/s11336-013-9388-3
- Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological Methods*, 14, 323–348. doi:10.1037/a0016973
- Therneau, T., & Atkinson, B. (2018). *Rpart: Recursive partitioning and regression trees*. R package version 4.1-13. Retrieved from <https://CRAN.R-project.org/package=rpart>
- Tutz, G., & Berger, M. (2016). Item-focussed trees for the identification of items in differential item functioning. *Psychometrika*, 81, 727–750. doi:10.1007/s11336-015-9488-3
- von Oertzen, T., Brandmaier, A. M., & Tsang, S. (2015). Structural equation modeling with Ω nyx. *Structural Equation Modeling*, 22, 148–161. doi:10.1080/10705511.2014.935842
- Yuan, K.-H., Fan, Z., & Zhao, Y. (2019). What causes the mean bias of the likelihood ratio test statistic with many variables? *Multivariate Behavioral Research*, 54, 840–855. doi:10.1080/00273171.2019.1596060
- Zeileis, A., & Hornik, K. (2007). Generalized m-fluctuation tests for parameter instability. *Statistica Neerlandica*, 61, 488–508. doi:10.1111/j.1467-9574.2007.00371.x
- Zhou, Y., & McArdle, J. J. (2015). Rationale and applications of survival tree and survival ensemble methods. *Psychometrika*, 80, 811–833. doi:10.1007/s11336-014-9413-1

Appendix

Example 1 R code

```
> library(MplusTrees)
> library(lavaan)
```

```
> script1 = mplusObject (
  TITLE = "Example #1 - Factor Model;",
  MODEL = "f1 BY x1-x3; f2 BY x4-x6; f3 BY x7-x9;",
  usevariables = c('x1','x2','x3','x4','x5','x6','x7','x8','x9'),
  rdata = HolzingerSwineford1939)
> fit.ex1 = MplusTrees(script1,
  HolzingerSwineford1939, group=~id,
  rPartFormula=~sex+school+grade, control=rpart.
  control(minsplit=100, minbucket=100, cp=.01))
> fit.ex1
```

Example 2 R code

```
> script2 = mplusObject (
  TITLE = "Example #2 - Count LGM;",
  VARIABLE = "COUNT = y1-y5;",
  MODEL = "i s | y1@0 y2@1 y3@2 y4@3 y5@4;",
  usevariables = c('y1','y2','y3','y4','y5'), rdata =
  data2)
> fit.ex2 = MplusTrees(script2, data2, group=~id,
  rPartFormula = ~ sex + alc + psmoke +
  pdrunk + pdrugs, control=rpart.control(cp=.001))
> fit.ex2
```

Example 3 R code

```
> script3 = mplusObject (
  TITLE = "Example #3 - Survival;",
  VARIABLE = "SURVIVAL = age;
  TIMECENSORED = smk (0 = RIGHT 1 = NOT);",
  MODEL = "age ON risk;",
  usevariables = c('age','risk','smk'), rdata =
  data3)
> fit.ex3 = MplusTrees(script3, data3), group=~id,
  rPartFormula = ~ sex + alc + psmoke +
  pdrunk + pdrugs, control=rpart.control(cp=.01))
> fit.ex3
```