Best Practices in Latent Class Analysis using Free Open Source Software

Caspar J. van Lissa[1,2], Mauricio Garnier-Villarreal[3], & Daniel Anadria[1]

[1] Utrecht University, Methodology & Statistics

[2] Open Science Community Utrecht

[3] Vrije Universiteit Amsterdam, Sociology

Author Note

Correspondence concerning this article should be addressed to Caspar J. van Lissa, Padualaan 14, 3584CH Utrecht, The Netherlands. E-mail: c.j.vanlissa@gmail.com

Abstract

Latent class analysis (LCA) refers to a family of techniques for identifying groups in data based on a parametric model. Examples include mixture models, LCA with ordinal indicators, and latent class growth analysis. Despite the popularity of this technique, there is limited guidance with respect to best practices in estimating and reporting mixture models. Moreover, although user-friendly interfaces for advanced mixture modeling have long been available in commercial software packages, open source alternatives have remained somewhat inaccessible. This tutorial describes best practices for the estimation and reporting of LCA, using free and open source software in R. To this end, this tutorial introduces new functionality for estimating and reporting mixture models in the `tidySEM` R-package whose backend relies on `OpenMx`.

Keywords: latent class analysis, mixture models, best practices, free open source software, tidySEM

Word count: 9001

Best Practices in Latent Class Analysis using Free Open Source Software

Latent class analysis (LCA) is an umbrella term that refers to a number of techniques for estimating unobserved group membership based on a parametric model of one or more observed indicators of group membership. The types of LCA have become quite popular across scientific fields, most notably finite Gaussian mixture modeling and latent profile analysis. Jeroen K. Vermunt et al. (2004) defined LCA more generally as virtually any statistical model where "some of the parameters [. . .] differ across unobserved subgroups".

Despite the popularity of LCA, two challenges impede broader adoption and correct application of the method. The first challenge is that most existing software that implements these models is commercial and closed-source. The second challenge is that there is a lack of standards for estimating and reporting LCA, and some myths and misunderstandings have been perpetuated in prior literature. This introduces a risk of misapplications of the technique, and complicates manuscript review and assessment of the quality of published studies. The present paper seeks to address these challenges by introducing updated guidelines for estimating and reporting LCA, based on current best practice. Furthermore, meeting the increasing demand for open source research software, this paper introduces new functionality in the R-package `tidySEM` (**vanlissaTidySEMTidyStructural2022?**) to perform LCA using free open source software (FOSS).

The Argument for Free Open Source Software

Despite the popularity of LCA, most existing software is commercial and closed-source, for example, Mplus (**muthenMplusUserGuide1998?**), for which the R-package `MplusAutomation` offers augmented LCA functionality (**hallquistMplusAutomationPackageFacilitating2018?**), and LatentGOLD [REF]. A downside of commercial software is that its use is restricted to individuals and institutions who can afford a license. Another downside is that the proprietary source code cannot be audited, debugged, or enhanced by third parties. This goes against open science principles (**lamprechtFAIRPrinciplesResearch2019?**), incurs a risk that mistakes in the source code go unnoticed, and curbs progress as independent developers cannot fix bugs or add functionality.

There are several compelling reasons to prefer free open source software (FOSS) instead. FOSS is freely available for anyone to use, modify, and distribute, which means that scientists can access and use a wide range of powerful tools without financial impediments. This is invaluable for reducing gatekeeping of researchers with limited budgets, such as those from developing countries. Furthermore, FOSS promotes collaboration within the scientific community, as its users can contribute to the development, maintenance, and support of the software. Finally, FOSS promotes transparency and reproducibility: Because the source code for FOSS is accessible, researchers can review and verify the methods and algorithms used, which helps to ensure the integrity and reliability of research findings. Overall, FOSS has the potential to significantly improve the efficiency and effectiveness of scientific research, while also promoting transparency, collaboration, and inclusion within the scientific community.

Notable FOSS for LCA includes mclust (Scrucca, Fop, Murphy, & Raftery, 2016), depmixS4 (Visser & Speekenbrink, 2010), and OpenMx (Neale et al., 2016). A crucial limitation of these existing packages is that some have limited functionality (e.g., Mclust), or implement specific innovations (e.g., depmixS4). OpenMx is unique in that, like Mplus, it implement LCA in the context of a fully featured structural equation modeling framework. Remaining obstacles to the use of `OpenMx` is that it is not very user-friendly and its functionality for LCA is poorly documented.

These limitations are addressed by the `tidySEM` package, which provide a user-friendly interface for LCA in `OpenMx`. The package also includes convenience functions to obtain relevant statistics, tests and plots in accordance with best practices in estimation and reporting LCA, as described in this paper. The primary purpose of the `tidySEM` package is to provide a tidy workflow for generating, estimating, reporting, and plotting structural equation models in a software-agnostic way. The user interface is simple, as models can be specified using the widely used `lavaan` syntax (Rosseel, 2012). The package contains user-friendly wrapper functions to conduct LCA with sensible defaults, but the resulting models can still be fully customized (low barrier, high ceiling). Specifically, LCA models constructed in `tidySEM` inherit from `OpenMx`' format `MxModel`, and remain compatible with downstream functions that process such models. Being FOSS, `tidySEM` facilitates fully-reproducible analyses and contributes to open science. It was developed with open science in mind, and adheres to the FAIR software principles (Findable, Accessible, Interoperable, and Reusable, **lamprechtFAIRPrinciplesResearch2019?**), and the OpenSSF Best Practices for FOSS software.

Defining Latent Class Analysis

LCA refers to methods for estimating unobserved groups based on a parametric model of observed indicators of group membership (**vermuntLatentClassAnalysis2004?**). It has become popular across scientific fields, under a number of different names; most notably (finite Gaussian) mixture modeling and latent profile analysis. The concept of LCA can be understood in different ways. Generally speaking, LCA assumes that the study population is composed of K subpopulations. It further assumes that the observed data are a mixture of data originating from those subpopulations; hence the name mixture model. Consider the simplest possible univariate "model", which is a normal distribution. This model has two parameters: the mean and the variance. LCA aims to estimate the values of those parameters across a known number of classes K, as well as the probability of every individual i belonging to classes $1 \ldots k$. Commonly, the same model is estimated across all classes, but with different parameters for each class (i.e., class-specific means and variances).

As an illustrative example, imagine that a detective wants to know if it would be possible to use mixture modeling to identify the sex of a suspect, based on footprints found at the crime scene. To test the feasibility of this approach, the detective records the shoe sizes and sex of 100 volunteers. The resulting observed data look like this:

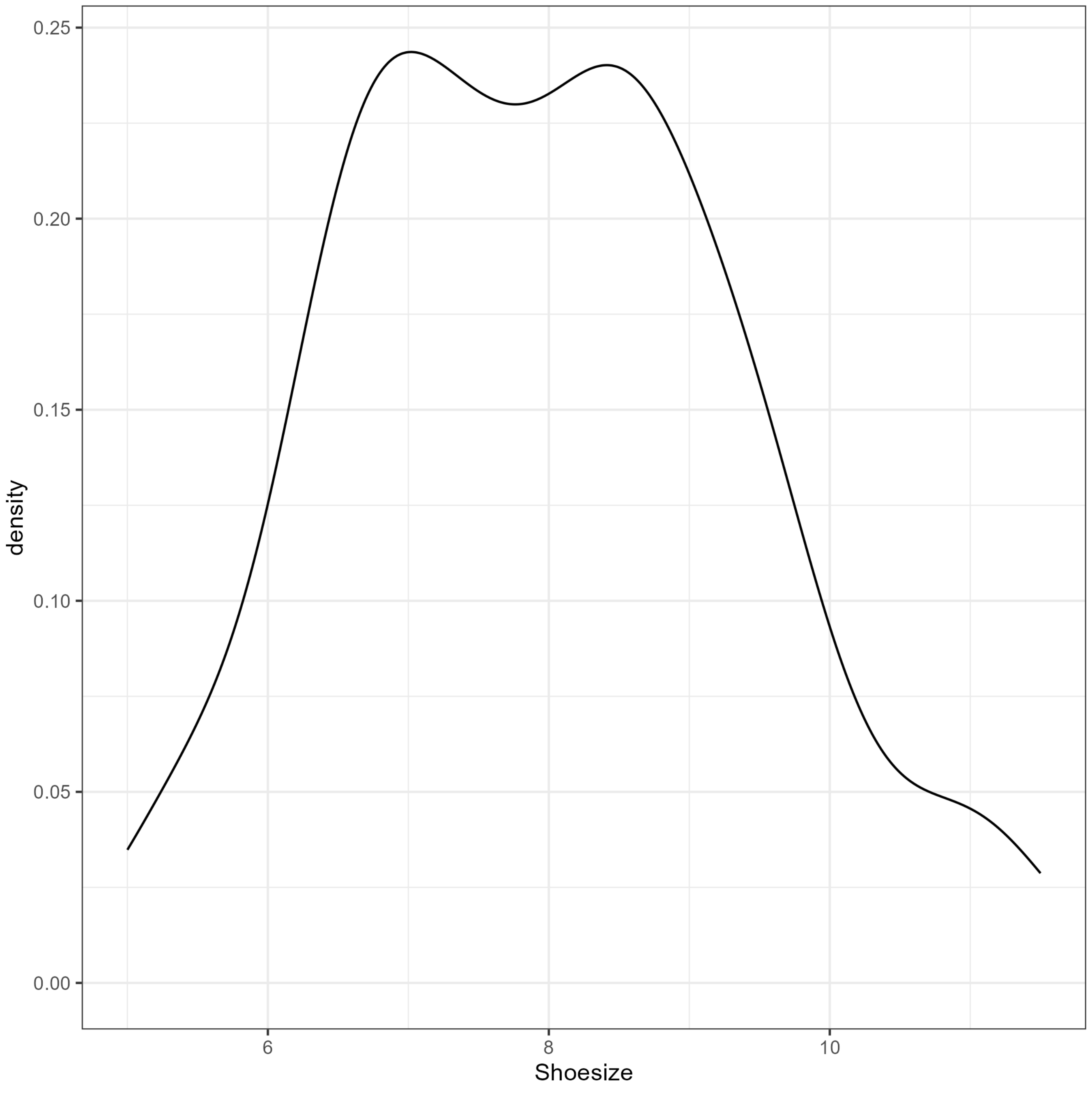

Figure 1. Kernel density plot of shoe sizes.

The distribution is evidently bimodal, which bodes well for the intended mixture model. In this case, the number of classes is known a-priori. When estimating a two-class mixture model, the detective observes that the model estimates the mean shoe size of the two groups are equal to 9.22 and 7.25, which is close to the true means of the two groups, namely 9.04 and 6.93. When tabulating estimated group membership against observed (known) group membership, it can be seen that women are classified with a high degree of accuracy, but men are not:

Table 1

Observed group membership by estimated class membership.

Observed	Class 1	Class 2
Man	28	21
Woman	0	51

Another way to conceptualize LCA is by analogy to a measurement model in structural equation modeling (**molenaarArbitraryNatureLatent1994?**). A mixture model is like confirmatory factor analysis, except with a categorical instead of a continuous latent variable. One difference between the two techniques is that factor analysis can be considered as a way to group observed *variables* into latent constructs, and LCA groups *individuals* into classes. While factor analysis seeks to describe the relations between variables, LCA seeks to describe unobserved groups and classify individual observations to those groups. In line with this distinction, LCA (mixture models) is referred to as a "person-centered" technique, and factor analysis as a "variable-centered" technique (Masyn, 2013; Nylund-Gibson & Choi, 2018).

When the focus is on the model parameters in each group, LCA can be thought of as similar to a multi-group structural equation model. The main distinction is that group membership is not known a-priori, but estimated from the data. Whereas in a multi-group model, the data are split by group and treated as independent samples, in LCA, all cases contribute to the estimation of all parameters in all groups. The relative contribution of each case to the parameters of each group is determined by that case's posterior probability of belonging to that group.

When the focus is on each individual's estimated class membership, LCA can be thought of as a type of clustering algorithm that corrects for (random) measurement error. Specifically, the posterior class probability that an individual belongs to a latent class can be computed from the likelihood of that individual's observed data under given the class-specific model. These posterior class probabilities can be used to weight follow-up analyses, or individuals can be assigned to classes based on their highest class probability. If the classes are clearly distinct, measurement error is low, and the latter approach may be acceptable. This perspective on mixture modeling is sometimes referred to as "model-based clustering" (Hennig, Meila, Murtagh, & Rocci, 2015; Scrucca et al., 2016). Many clustering algorithms apply some recursive splitting algorithm to the data. By contrast "model-based" clustering refers to the fact that LCA estimates cluster membership based on a parametric model. Since a parametric model estimates only a fixed number of parameters, it can be a relatively parsimonious solution compared to non-parametric techniques.

Finally, in the context of machine learning, LCA can be considered as an *unsupervised* learning method for *classification* (Figueiredo & Jain, 2002). The term *unsupervised* refers to the fact that the outcome variable – true class membership – is not known, and the term *classification* refers to the fact that the algorithm is predicting a categorical outcome: class membership.

A Taxonomy of Latent Class Analysis Methods

The term LCA was initially conceived to refer to analyses with categorical, usually binary indicators (Collins & Lanza, 2009). It was later generalized to refer to any technique that estimates a categorical latent variable based on a parametric model of observed indicators, as it only refers to the purpose of the analysis, and does not imply restrictions to the model used, or the level of measurement of the indicators (Jeroen K. Vermunt et al., 2004). To prevent ambiguity, we refer to LCA with binary or ordinal indicators as *LCA with ordinal indicators*. Given the abundance of redundant terms in use for closely related types of LCA models, we here provide a rudimentary taxonomy.

A common type of LCA is the *finite Gaussian mixture model*; a univariate analysis where the observed distribution of a single continuous variable is assumed to result from a mixture of a known number of normal (Gaussian) distributions. The parameters of a finite Gaussian mixture model are the means and variances of these underlying normal distributions. The analysis of shoe sizes presented earlier is a canonical example of this type of analysis. In the multivariate case, with more than one indicator variable, the parameters of a mixture model are the means, variances, and covariances between the indicators (which can be standardized to obtain correlations). These parameters can be estimated freely, or set to be constrained across classes (Collins & Lanza, 2009).

The technique known as *latent profile analysis (LPA)* is a special case of the mixture model, which assumes conditional independence of the indicators. Conditional independence means that, after class membership is accounted for, the residual covariances/correlations between indicators are assumed to be zero. In some cases, such constraints will be inappropriate; for instance when the cohesion between indicators is expected to differ between classes (Collins & Lanza, 2009). Consider, for example, a mixture model of ocean plastic particles, which found two classes of particles based on length and width: A class of larger particles with a low correlation between length and width, and a class of smaller particles with a high correlation between length and width. The reason for this difference in correlations makes theoretical sense: the large particles were heterogenous in shape, whereas the smaller particles had been polished to a more uniform (rounded) shape by the elements (Alkema, Van Lissa, Kooi, & Koelmans, 2022).

A longitudinal extension of LCA is the Hidden Markov Model (HMM, also called Latent Transition Analysis), which estimates the change in class membership over time with the markov assumption over time. The markov assumption states that the class membership at time t is only affected by the class membership at time $t-1$. The structural part of the model is comprise of the initial states (proportion of the sample in each class at the first time point), and transition probabilities (probabilities to switch class membership across time). The transition probabilities are estimated as multinomial logistic regressions between class memberships (Jeroen K. Vermunt, 2010).

Finally, some LCA models use latent variables as indicators. A common application of this approach is in longitudinal research with repeated measures of a construct. Examples of this approach include *growth mixture models* (GMM) and *latent class growth analyses* (LCGA) (Jung & Wickrama, 2008). These techniques estimate a latent growth model to describe grouped trajectories over time. The indicators of class membership in a GMM are the intercepts, and variances, and covariances of the latent growth variables that describe individual trajectories on the observed variables. For linear trajectories, these latent variables are an intercept and slope. The free variances allow trajectories to vary within classes. The LCGA differs from the GMM in that it assumes homogeneity of trajectories within classes. This is achieved by restricting the (co)variances of the growth parameters to zero. Any variance in the indicators not explained by the class-specific latent trajectories is assumed to be error variance.

Use Cases for Latent Class Analysis

There are several use cases for which LCA methods are suitable.

Testing theory. Although LCA is often discussed as an exploratory analysis technique, it can also be used in a confirmatory manner. Given that LCA is similar to confirmatory factor analysis, it can be used to similar ends: To assess whether a theoretical measurement model holds. This use case is relevant when testing a theory that postulates the existence of a categorical latent variable. For example, *identity status theory* posits that, at any given point in time, adolescents reside in one of four identity statuses (Marcia, 1966). LCA can be used to identify these four statuses based on observed indicators (e.g., self-reported identity exploration and commitment) (**luyckxDevelopmentalTypologiesIdentity2008?**). Similarly, *personality type theory* states that dimensional differences in personality can largely be explained by an undercontrolled, overcontrolled, and resilient type, which could also be restored using LCA (**donnellanResilientOvercontrolledUndercontrolled2010?**).

One challenge is that, unlike confirmatory factor analysis, LCA does not provide absolute fit indices (see Model Fit Indices). Thus, to test a theory, one can ascertain that the data are better described by the number of classes dictated by theory than by a different number of classes. Furthermore, it would be possible to test whether the observed pattern of class-specific means corresponds to a hypothesized pattern; either qualitatively or quantitatively, using significance tests for pre-specified values (see Inference).

Appendix A is a tutorial to confirmatory LCA in `tidySEM`, based on a recent study that tested the theory that identities of Belgian adolescents with migration backgrounds could be summarized as those relating to their heritage, national and regional identity (Maene, D'hondt, Van Lissa, Thijs, & Stevens, 2022).

Exploring Heterogeneity. Another use case is to explore whether a population is comprised of latent subpopulations, and if so, how many. The primary focus in this use case is on class enumeration. For example, one study explored heterogeneity in the population of military service members and veterans with prior suicide attempts, and identified two latent profiles characterized by high versus low suicide risk based on self-reported suicidal ideation and information about prior attempted methods and severity (Gromatsky et al., 2022). Knowing that the population of suicidal service members is heterogeneous can inform future research, policy, and care provided. Appendix B offers a tutorial for exploratory LCA, based on the aforementioned study of the distribution of ocean microplastics (Alkema et al., 2022).

Measurement model. LCA can also be used as a measurement model, similar to confirmatory factor analysis. This is useful when class membership is measured imperfectly by several indicators: LCA can partial out measurement error and restore most likely class membership. For example, one study found discrepancies between self-reported employment status and official register data (Pavlopoulos & Vermunt, 2015). While it might be tempting to assume that register data is free of error, this assumption was shown to be false. A mixture model was used to estimate employment status, correcting for measurement error, from both self-report and registry data. The study investigated random and systematic measurement error in both indicators, which reveal strengths and weaknesses of both types of assessment.

This example also illustrates that LCA allows us to evaluate the reliability of different indicators (Geiser & Wurpts, 2014). A continuous indicator that discriminates well between classes will show large differences in class-specific means; an ordinal indicator will show very high or very low conditional response probabilities for some classes but not for others. This information can be used to select the indicators that are most diagnostic of class membership. For an example of LCA with a focus on differential item functioning, consider a study on university admission tests of Saudi Arabian students (Tsaousis, Sideridis, & AlGhamdi, 2020).

Dimension reduction. LCA can also be used to reduce high-dimensional data to just a few prototypes. Factor analysis is also sometimes used for dimension reduction, but only accounts for linear covariance between indicators, whereas LCA can accommodate complex - but discrete - patterns. For example, imagine that there exist two indicators - x and y, and some individuals score high on both, some score low on both, and some score high on x but low on y. Factor analysis would only be able to capture the differences between the first two groups well; LCA captures differences between all three. Note that this is a pragmatic use of LCA; there is no theory about the existence of a categorical latent variable (cf. Testing theory). Consequently, the class solution should be treated as descriptive of the sample, and not be "reified" - treated as evidence of the existence of a categorical latent variable.

The use of LCA for dimension reduction is common in longitudinal research, where one developmental process is summarized using latent class growth analysis (LCGA), and the resulting categories are then used as a moderator of another developmental process in a multi-group model. For example, one study conducted a LCGA of adolescent empathy development on two dimensions of empathy and found three groups: high, average, and low empathy (Van Lissa et al., 2015). These groups were then used as moderator of a latent growth analysis of adolescent- and parent-reported conflict. Results showed that the high-empathy group evidenced greater adolescent-parent agreement about the incidence of conflict than the other groups, and that the low empathy group had more conflict with parents than the other groups. Note that both these key findings are non-linear; modeling them in a single-group model would be difficult. LCGA greatly simplifies model specification and interpretation, albeit at a cost of some loss of information and, potentially, generalizability. Appendix C is a tutorial on LCGA, based on ongoing research on trajectories of depressive symptoms in military service members after deployment.

The need for dimension reduction can be especially pertinent when using ordinal indicators. For example, one study measured fifteen dichotomous health indicators among injured military personnel (MacGregor et al., 2021). Combinatorics informs us that fifteen dichotomous items have 2^{15} or $32,768$ unique symptom combinations. However, LCA was able to reduce response patterns to five clinically meaningful latent classes.

Classification. Another use case is to estimate individuals' unobserved class membership based on observed indicators. The shoe size example in Figure 1 is a rudimentary illustration of this application. In applied research, LCA is often used for classification in clinical contexts. For example, LCA was used to determine clinical cut-off scores for diagnosing whooping cough (Baughman, Bisgard, Lynn, & Meade, 2006). Another study used LCA to not only classify the sample used to estimate the model, but also future cases [REF Zegwaard et al, in prep]. The study identified four types of care providers among those who supported close kin with mental health problems. Importantly, the LCA model was embedded in an interactive app that could be used by healthcare professionals to determine the most likely class membership for new care providers, to provide more targeted support. To obtain predicted class membership based on an existing LCA model, the authors computed the likelihood of data from a new participant under the multivariate normal distributions estimated within all latent classes. These applications of LCA aid the clinical decision making process, and help decide whether additional support is warranted, or what type of intervention is appropriate.

Violations of normality. Finally, LCA can be used to deal with data which violate certain assumptions. For example, zero-inflated and hurdle models are special cases of LCA for dealing data with a very high proportion of zeros (extreme right skew) (**baughmanMixtureModelFramework2007?**). These models estimate one class to explain the "true zeros", and another class with a different distribution to explain all other values, which can include zero. Note that this LCA has different models for both classes, and thus falls beyond the scope of this paper. For instance, the aforementioned study on marine plastic noted that many prior studies had treated the distribution of particles as (multivariate) normal, despite clear deviations from normality (Alkema et al., 2022). A mixture of two normal distributions with distinct variances described the observed distribution better.

Best Practices

The best practices in estimation, as outlined in Table **??**, are rooted in existing recommendations for best practices for estimating specific subtypes of LCA, including latent class growth analysis (Van De Schoot, Sijbrandij, Winter, Depaoli, & Vermunt, 2017) and latent class analysis with ordinal indicators (e.g., Nylund-Gibson & Choi, 2018). These were generalized to be relevant to all types of LCA, and updated to current best practices, as explained below.

Best Practices in Estimation

#	Item
1.	Examining Observed Data
2.	Handling Missing Data
3.	Alternative Model Specifications
4.	Software
5.	Algorithm
6.	Class Enumeration
7.	Model Fit Indices
8.	Classification Diagnostics
9.	Interpreting Class Solutions
10.	Label switching

Examining Observed Data. Examining observed data is essential for any analysis as it may reveal patterns and violations of assumptions that had not been considered prior to data collection. The function `descriptives(data)` provides extensive descriptive statistics, including the number of unique values, variance of continuous and categorical variables, missingness, skew and kurtosis (peakedness of the distribution). However, sample-level descriptive statistics are of limited value when the goal of a study is to identify subsamples using LCA. Plots convey additional information; specifically, density plots for continuous variables, and bar charts for categorical ones. Density plots can also aid in the choice of the number of classes, as explained in the section on visualization.

Particular points of attention include measurement level of the indicators. Recall that `tidySEM` can estimate mixture models with either numeric (type `numeric` and `integer`) or `ordered` indicators. The column `type` should thus only contain these types. Also check that the number of `unique` values match the intended measurement level. Numeric variables with few unique values are typically better modeled as ordinal. Our experience consulting on LCA methods and moderating the `tidyLPA` Google group suggest that the misapplication of mixture models to ordinal (e.g., Likert-type) indicators is the most common source of user error. Whereas it has been argued that some parametric methods are robust when scales with 7+ response categories are treated as continuous (e.g., Norman, 2010), it is very unlikely that an ordinal variable can be modeled as a *mixture* of multiple normal distributions. To illustrate the problem, consider what happens when estimating a 5-class solution on a single 5-point Likert scale indicator. Each class-specific mean would describe a single response category, and a class-specific variance component would be empirically underidentified. In sum, variables with few unique values might not be suitable for Gaussian mixture modeling; LCA with ordinal indicators is more appropriate.

Data preprocessing. LCA can accommodate variables of different measurement levels: continuous, binary, nominal, and ordinal. Numeric variables should have an interval measurement level, such that it is sensible to estimate means and (co)variances. For the remaining three measurement levels, some preprocessing is required. The exact data types accepted by `tidySEM` at the time of writing are `numeric` and `ordered`. To correctly specify ordered variables such as Likert scales, use the function `OpenMx::mxFactor()`. Binary variables, such as presence or absence of some symptom, are also treated as ordinal, with one category coded as zero and the other as one. Nominal variables are converted to binary indicators of group membership via dummy coding, for example, using `mx_dummies()`.

A second consideration in data preprocessing is that models with indicators on different scales may present with convergence issues. The reason for this is that the parameter space is high-dimensional with many potential local optima. If one indicator is on a much larger scale than another, the parameter space may be more extensively explored for that parameter. The problem can be resolved by transforming the indicators to place them on (roughly) similar scales. The most common transformation is standardization, which involves subtracting the mean of each variable from each value and dividing by the standard deviation. In R, this transformation is applied using the `scale()` function. Standardization retains all available information, and model parameters can be transformed back to the original scale of the indicators by reversing the transformation. The transformed variable has a mean of zero and a standard deviation of one. Note that standardization is typically not necessary if indicators are already on the same, or similar, scales.

A third consideration in preprocessing relates to the univariate distributions of indicators. For example, the `plas_depression` data show extreme right skew in depressive symptoms: most respondents report few symptoms, and few respondents report many symptoms. Similarly, the `alkema_microplastics` data are extremely right-skewed: Very small plastic fragments abound in the oceans, and large pieces are rare. These distributional characteristics must be taken into account either in preprocessing, or model specification. In preprocessing, the data can be transformed to reduce skew. After reducing skew, a simpler LCA model may fit the data well. Alternatively, one can specify a model that accommodates the distributional idiosyncrasies. For example, allowing free variances across classes might result in one class with a near-zero mean and small variance to account for the fact that most respondents have few depressive symptoms, and one class with an elevated mean and large variance to account for all respondents with non-zero symptoms.

Handling Missing Data. Previous work has emphasized the importance of examining the pattern of missing data and reporting how missingness was handled (Van De Schoot et al., 2017). While we concur that investigating missingness is due diligence, it is important to emphasize that missingness is adequately handled by default in `OpenMx`, which is the computational backend of `tidySEM`. Its Full Information Maximum Likelihood (FIML) estimator makes use of all available information without imputing missing values, by estimating the model based on all available information for each subject (Enders, 2022; **endersRelativePerformanceFull2001?**). FIML is a best-practice solution for handling missing data; at least on par with multiple imputation (Lee & Shi, 2021). Multiple imputation is less suitable to LCA for two reasons. First, because LCA methods are often computationally intensive, and conducting them on several imputed datasets multiplies the load. Second, because many multiple imputation methods assume normality; an assumption likely to be violated when data come from a heterogeneous population.

Three types of missingness have been distinguished in the literature (Enders, 2022; Rubin, 1976): random missingness (MCAR); missingness contingent on *observed* variables that can be known and controlled for (MAR); and missingness related to unobserved variables, which can be neither known nor controlled for (MNAR). A so-called "MCAR" test is often reported (Jamshidian & Jalal, 2010), but note that this name is somewhat misleading, as a significant test indicates that missingness is MAR, but a non-significant test statistic does not distinguish between MCAR or MNAR. As the classic MCAR test relies on the comparison of variances across groups with different patterns of missing data, it assumes normality (Little, 1988). This assumption is tenuous in the context of LCA. A non-parametric MCAR test may be more suitable (Jamshidian & Jalal, 2010). For this tutorial, the lead author has implemented this test in the `mice` package as `mice::mcar()`. Our recommendation is to inspect and report the proportion of missingness per variable (e.g., using `tidySEM::descriptives()`), and report the `mice::mcar()` test. As FIML estimation assumes that missingness is either MCAR or MAR, one would proceed with FIML regardless of the outcome.

One minor concern is that `tidySEM` uses a different algorithm to determine starting values for complete data (K-means) and for missing data (hierarchical clustering). This should not be a problem, however, as the global optimizer used to estimate the model should be independent of starting values (**coranaMinimizingMultimodalFunctions1987?**). A second concern is that FIML is not suitable for cases with complete missing data. If such cases must be included, their values must be imputed first.

Model Specification. In principle, LCA can be based on any parametric model. At present, `tidySEM` only facilitates estimation of models that can be specified in `lavaan` syntax. The available model parameters depend on the measurement level of the indicators.

Continuous indicators can be parametrized in terms of means, variances, and covariances. LCGA and GMM have these same parameters, but with respect to the latent growth variables. Ordinal indicators can be parametrized differently; the default approach in `tidySEM` assumes that each ordinal indicator reflects an underlying standard normal distribution, $N(\mu = 0, \sigma = 1)$. The parameters are "thresholds" that correspond to quantiles of this distribution, estimated based on the proportion of individuals in each of the response categories of the indicator variable. For example, a binary indicator has a single threshold that distinguishes the two response categories. If responses are distributed 50/50, then the corresponding threshold would be $t_1 = 0.00$. If the responses are distributed 60/40, then the resulting threshold would be $t_1 = 0.25$. All of the aforementioned parameters can be freely estimated, or constrained across classes, or fixed (e.g., to zero).

Prior literature emphasized the importance of considering alternative model specifications (**vandeschootGRoLTSChecklistGuidelinesReporting2017?**). What has remained underemphasized is that alternative model specifications should be approached differently for exploratory versus confirmatory analyses (see Use Cases). Most prior literature has emphasized exploratory LCA (e.g., Nylund, Asparouhov, & Muthén, 2007). In this context, model specification typically consists of an exhaustive grid search along several model specifications and varying numbers of classes. A "final" model is then selected based on a combination of fit indices, significance tests, and interpretability (see Class Enumeration).

Confirmatory LCA does not require such an extensive search over the model space. If hypothesized models have been specified a priori, and possibly even preregistered, it makes sense to focus on these models, rather than proceed in a purely data-driven manner. Nonetheless, the theoretical model specification should still be compared to a few alternatives because LCA lacks objective fit indices. The fit of a theoretical model can thus only be assessed by comparison to other candidates. A one-class model is a sensible benchmark to test whether the theoretical model fits better than a model that assumes a homogeneous population. Other candidates for comparison include competing theoretical models and those with a few more or less classes than the hypothesized number. If the theoretical model has much better fit than other models, for example, based on IC weights, this provides evidence for the theory. If another model fits substantially better, this provides evidence against the theory. If fit differences are small, the theoretical model may still be preferred, as theory is a relevant consideration in model selection (**jungIntroductionLatentClass2008?**). The fact that the theoretical model received little differential support should of course be discussed.

A second consideration in model specification is the risk of overfitting (**hastieElementsStatisticalLearning2009?**). Overfitting occurs when a model has many parameters relative to the number of observations, causing it to capture idiosyncratic noise in the sample and limiting its generalizability to new samples. As explained before, the number of parameters in LCA scales with the number of estimated classes. Consequently, LCA with many classes have a potentially very high number of parameters. In Bayesian mixture modeling, it was observed that if the estimated number of classes exceeded the true number, the number of observations assigned to these classes tended towards zero as sample size increased (**rousseauAsymptoticBehaviourPosterior2011?**). Although it is not known whether frequentist LCA exhibits the same behavior, low class counts are still an indicator of potential overfitting. The section on Class Enumeration addresses ways to guard against overfitting.

Models can be specified in `tidySEM` in different ways. The wrapper functions `mx_profiles()` and `mx_lca()` construct LCA models for continuous and ordinal indicators respectively, without needing to specify model syntax. The function `mx_profiles()` has two arguments to free elements of the covariance matrix: `variances` and `covariances`. By default, `mx_profiles()` fixes covariances to zero, and thus performs latent profile analysis. Finally, function `mx_mixture()` accepts a more flexible `model` argument, which can be a character string specifying a model in `lavaan` syntax, which is translated to `OpenMx` models via the function `as_ram()`, or a list of `OpenMx` models.

Estimation and Convergence. LCA parameters and model fit statistics can be estimated in different ways. The choice of estimator depends on the presence of missing values, sample size, number of indicators, and available computational resources (Weller, Bowen, & Faubert, 2020). LCA in `tidySEM` uses maximum likelihood estimation (ML), which can be conceptually understood as follows: Imagine we are estimating two parameters, e.g. the class-specific means μ_c on a continuous indicator for two classes (ignoring the variance for now). ML attempts to find a combination of values for these two parameters that minimizes minus two times the log likelihood ($-2LL$) of the data. To understand the ML optimization procedure, imagine a three-dimensional landscape: The X and Y dimensions represent potential values of the class-specific means, $X = \mu_1$ and $Y = \mu_2$, and the Z-dimension is determined by $Z = -2LL$. The optimizer must find the deepest “valley” in this landscape, which reflects the combination of μ_1 and μ_2 that minimizes the $-2LL$. The ML optimizer behaves somewhat like a marble dropped in this landscape. It is dropped at some random point in space, and rolls into the nearest valley (a combination of values for μ_1 and μ_2 that has a low $-2LL$). The challenge is that LCA models are often highly parametrized. This means that the landscape is not three-dimensional, but (very) high-dimensional, with many valleys and sparse data to provide information about which valley is deepest. If the marble rolls into one of these valleys, it will settle there and not climb out again. The estimator has “converged”.

A unique challenge in LCA is that not only model parameters, but also class membership must be estimated. Parameters in all classes are estimated from the same dataset, but for each class, observations are weighted by their probability of being in that class - which is in turn computed based on the class-specific parameters. The resulting chicken-and-egg problem is solved by the expectation-maximization (EM) algorithm: It alternates between estimating model parameters, calculating posterior class probabilities from those parameters, re-estimating model parameters using those new class probabilities as weights, and so on, until both have converged.

A risk with this approach is that the optimizer might get stuck in a shallower valley (a "local optimum"), and never discovers the deepest valley (the "global optimum", or best solution). One solution to this problem is to drop many marbles at random places, compare their final $-2LL$ values, choose the solution with the lowest $-2LL$, and make sure that several marbles replicated this solution. This is the "random starts" approach implemented in, e.g., Mplus.

One problem with the random starts approach is that it is computationally expensive to run this many replications. Moreover, because the algorithm begins with random starting values, it is inefficient because many marbles are likely to start far away from any valley. Additionally, if all sets of starting values are close to a specific local optimum, there is a risk that the global optimum is never found. An alternative solution that overcomes these challenges is to use a global optimizer, like simulated annealing (SA, **coranaMinimizingMultimodalFunctions1987?**). SA does not behave quite like the marbles of the random starts approach; instead, it iteratively considers some "destination" in the landscape, and compares its $-2LL$ to the current one. If the destination $-2LL$ is lower, the estimator moves there. However, if the destination $-2LL$ is *higher*, it still moves there occasionally, with a certain probability. Think of this as occasionally flicking the marble, to see if it rolls back into the current valley, or finds another, even deeper one. This property allows SA it to escape local optima and find the global optimum.

Since SA is a global optimizer, it should be independent of starting values. With this in mind, it is efficient to start estimation from a "reasonable solution", instead of a random starting point. For example, if we assume that the different classes are likely to have different mean values on the indicators, then a nonparametric clustering algorithm - like K-means or hierarchical clustering - can be used to determine these cluster centroids (see also **biernackiChoosingStartingValues2003?**). By default, `tidySEM` derives starting values using K-means clustering [REF], computing the mixture model starting values by treating the K-means solution as multi-group model. Next, SA is used to find the global optimum solution. Finally, SA is followed up with a short run of the ML algorithm, as ML inherently produces an asymptotic covariance matrix for the parameters that can be used to compute standard errors. Note that these defaults can be manually overridden.

In practice, models do not always converge. Reasons may include local optima, bad starting values, incorrect model specification (the model may be too complex for the data, or conversely, too simple to capture important patterns), or poor data quality (small data sets, or distributions that are difficult to model). Convergence problems may be indicated by errors and warnings from the estimator, or they may be suspected, for example, when results change if the analyses are reproduced.

Nonconvergence may be due to bad model specification. Consider, for example, an LCA with multiple ordinal indicators. If certain combinations of responses never occur together, in general or within one of the classes, this can result in extreme conditional item probabilities (exactly 0 or 1). Such extreme values impede model convergence. This problem could be addressed by trying alternative model specifications (e.g., fewer classes), or by merging categories of the ordinal indicators with very few responses. Similarly problems occur with continuous indicators. For instance, consider a zero-inflated indicator and a two class LPA. To account for the excess zeroes, one class will have a mean of zero. It is difficult to estimate the standard deviation for this class, however, because all of its members have the same value. If standard deviations are constrained to be equal across classes, this model will likely not converge. This problem can be addressed by considering more appropriate model specifications (e.g., free class variances).

Nonconvergence due to bad starting values can be overcome by manually specifying starting values: Specify the argument `run = FALSE` in the call to `mx_mixture()` and related functions. This returns the model with default starting values, which can be updated. Nonconvergence and local optima may also be overcome by re-estimating the model while permuting the starting values. The `OpenMx` package contains a family of functions for this purpose: `mxTryHard()` and `mxTryHardWideSearch()` for continuous indicators, and `mxTryHardOrdinal()` for ordinal indicators. When poor data quality is to blame, the only solution is to collect more, or higher quality, data.

Class Enumeration. Class enumeration refers to the process of determining the appropriate number of classes. It is approached differently for exploratory versus confirmatory LCA. In exploratory LCA, the set of models typically consists of a sequence from 1 to the maximum number of classes K. The number K may be chosen a-priori or dictated by the data. It is sensible to limit K to the maximum number of classes that result in valid and interpretable solutions (also see Classification Diagnostics). Above a certain number of classes, convergence problems typically arise, or the proportion of cases assigned to the smallest classes may be low relative to the number of class-specific parameters. In Bayesian LCA, it has been observed that, when the estimated number of classes exceeds the true number, the proportion of cases assigned to the excessive classes converges to zero as the sample size increases (**rousseauAsymptoticBehaviourPosterior2011?**). It is not known whether this also happens in frequentist LCA; nonetheless, near-zero class memberships should be taken into account when determining the maximum K.

In confirmatory LCA, the set of models with different numbers of classes to be compared is typically smaller. Its purpose is to provide context for the relative fit of the theoretical model, or to compare multiple theoretical models.

In class enumeration, the set of models is compared on several statistical and substantive criteria. In exploratory LCA, these criteria may be decisive in selecting a final model; in the confirmatory case, the burden of proof is reversed: there must be convincing evidence against the theoretical model in order to reject it. Statistical criteria to consider in class enumeration include model fit indices and likelihood ratio tests (Masyn, 2013; Weller et al., 2020). Substantive criteria include convergence and model identification, class separability, and interpretability. We address each of these criteria below.

Model Fit Indices. As there are no absolute fit indices for LCA, information criteria are typically used to assess relative fit. Information criteria are computed from the $-2LL$, which is the most basic fit measure, testing the overall model fit if the model properly represents the data. Information criteria add a penalty for model complexity to the $-2LL$, and thus balance fit and complexity. The lower the value of an information criterion, the better the overall fit of the model. The original information criterion is the Akaike Information Criterion (AIC), computed as $-2LL + 2p$, where p is the number of parameters. The Bayesian Information Criterion (BIC) multiplies the penalty for complexity with a function of the sample size, $p\ln(n)$, such that complex models are penalized more in smaller samples. Finally, the so-called *sample size adjusted* BIC (saBIC) implements a different ad-hoc penalty for sample size, $\ln(\frac{n+2}{24})$. Although a simulation study showed that the saBIC performed well for class enumeration, its penalty is not based on theory [REF]. BIC also performs well and is theoretically substantiated; as such, it may be the most appropriate information criterion to use for model comparison (Masyn, 2013; Nylund-Gibson & Choi, 2018). All three ICs are available in the output of `table_fit()`.

Information criteria may occasionally contradict each other, so it is important to identify a suitable strategy to reconcile them. One option is to select a specific fit index, such as the BIC, before seeing the results. This choice can be preregistered to avoid concerns about cherry picking. Another option is to always prefer the most parsimonious model that has best fit according to any of the available fit indices. Other approach is to select several competing model based on the ICs, and evaluate the theoretical intepretation. Yet another option is to incorporate information from multiple fit indices using the analytic hierarchy process (Akogul & Erisoglu, 2016), which is implemented in `tidyLPA` (**rosenbergTidyLPAPackageEasily2018?**). Finally, one can use a scree plot to determine the inflection point at which additional classes contribute little to further decreases of the ICs (Nylund-Gibson & Choi, 2018). The preferred number of classes is at the inflection point where the slope of the curve is clearly leveling off. Adding this many classes results in substantial decreases of the IC, and adding further classes results in relatively small decreases.

Given a specific IC, it is also possible to calculate so-called "IC-weights", which indicate the relative support the data provide for each model in the set (**wagenmakersAICModelSelection2004?**). This is particularly useful for confirmatory LCA, as it allows one to quantify the support for the theoretical model. IC-weights are calculated by calling `ic_weights()` on the model fit table. Note, however, that IC-weights will (strongly) favor the model with the lowest IC value; the scree plot approach above may be more suitable to select the most parsimonious model with adequate fit.

Nested Model Tests. When two models are nested, the significance of the difference in fit between these models can be tested using a Likelihood Ratio test (LR). The test statistic is computed as the -2 times the log of the ratio of likelihoods, and its degrees of freedom are equal to the difference in the degrees of freedom of the two compared models. Since a $k - 1$-class model is nested in a k-class model, it is possible to test the difference in model fit using a LR test. One challenge is that the LR test statistic is not asymptotically χ^2 distributed for LCA. The so-called Lo-Mendell-Rubin adjusted LR test (LMR) overcomes this problem by applying an ad-hoc correction to the LR [REF lo 2001]. This test is implemented in `tidySEM`, and is part of the default output of `table_fit()`, comparing subsequent rows of the table.

Another solution is to derive an empirical sampling distribution for the LR difference. The so-called bootstrapped Likelihood Ratio Test (BLRT) does so by simulating data from the $k - 1$-class model B times (e.g., 1000), and fitting the k- and $k - 1$-class models to each simulated dataset [REF Nylund et al., 2007]. The p-value of this test is the proportion of B

in which the LR of the models fit to simulated data exceeds the LR observed in the real data. Simulation studies suggest that the BLRT has greater statistical power than the LRT, although both performed comparably well at detecting the true number of classes [REF Nylund 2007, REF Tein 2013]. A crucial limitation of the BLRT is that it is very computationally expensive. With this in mind, `table_fit()` does not provide the BLRT by default. Specific models can be compared using the function `BLRT()`.

Anecdotal experience as a statistical co-author on LCA studies suggests that both these tests tend to be liberal, returning significant p-values for models with more classes even when other indicators suggest that those models are overfitted (e.g., near-zero class counts). Other authors have reported similar experiences (**sinhaPractitionerGuideLatent2021?**), which matches the differences between LRT and ICs from information theory as LRT tests and compare the models based on in sample predictive accuracy, and ICs evaluates and compare the models out-of-sample predictive accuracy which helps to control for overfitting (McElreath, 2020). This suggests that, like existing ICs, LR tests might not adequately balance fit and complexity in LCA. Better solutions for class enumeration are thus an important target for future research.

A promising novel solution is the so-called *lazy bootstrap*, which is much more efficient than the BLRT because it is based on sample statistics of model-implied data generated from the $k-1$ and k-class models, rather than on fit statistics of LCA models fit to those data (Kollenburg, Mulder, & Vermunt, 2018). This simulation-based approach to inference is similar to the posterior predictive p-value in Bayesian statistics, which describes how well each model is able to reproduce the observed correlation matrix (see McElreath, 2020). One version of the lazy bootstrap is already implemented in the commercial program latentGOLD [REF]. At the time of writing, we are developing a more general simulation-based likelihood ratio test (SBLRT) for class enumeration. Future versions of `tidySEM` will likely incorporate this test.

Classification Diagnostics. Classification diagnostics assess whether the classes identified by LCA are distinct and clearly separable. These diagnostics are not model fit indices and should not be used for class enumeration, because a model can fit the data well and still show poor latent class separation (Masyn, 2013). However, classification diagnostics can be used to eliminate models: If one analysis goal is to interpret properties of (members of) classes, it makes sense to only consider models that classify well (Nylund-Gibson & Choi, 2018). Classification diagnostics should thus not be used for model selection, but they can help restrict the search space of acceptable models.

All classification diagnostics are derived from the posterior classification probabilities: an $n \times k$ matrix called P, with the probability that every individual n belongs to latent class k. When classification accuracy is high, and classes are distinct, each individual's posterior class probabilities are high for one class, and low for the remaining classes. The individual classification probabilities P are obtained by running `class_prob(res, type = "individual")`. Note that one column is added to the matrix, with each individual's most likely class membership M. M is obtained by assigning each individual to the class corresponding to their maximum posterior class probability. Predicted class membership M is often of interest to researchers who wish to carry out follow up analyses. Note, however, that follow-up analyses should take classification error into account. The function `BCH()` does this automatically.

The matrix P can be summarized in different way. For example, the estimated frequency table of the latent categorical variable can be obtained by summing the posterior probabilities by class (i.e., the column sums of P). This frequency table F_P is obtained by calling `class_prob(res, type = "sum.posterior")`. These numbers are fractional because this table takes classification error into account. Individuals can contribute partially to multiple classes.

Second, most likely class membership based on the highest posterior class probability,

F_M, is obtained by running `class_prob(res, "sum.mostlikely")`. This is the frequency table of the variable M. Unlike F_P, F_M ignores classification error. If every observation is classified with perfect accuracy, F_M and F_P are identical. Part of the output of `table_fit()` for LCA analyses are the minimum and maximum percentage of the sample assigned to a particular class, `n_min` and `n_max`, derived from F_M. This is reported by default because the smallest class proportion, `n_min`, is of particular interest. When the size of the smallest class is not much larger than the number of parameters per class, proceed with caution: The model might not be locally identified in that class (**depaoliMixtureClassRecovery2013?**). Existing rules of thumb have focused on smallest class size in terms of percentage of the total sample size. This is not particularly meaningful, however, as the primary concern is that small classes may comprise too few observations to accurately estimate class parameters. Thus, a better rule of thumb might be to ascertain that the smallest class has many more observations than the average number of parameters per class. Another good practice is to ensure that the smallest class has a unique and meaningful substantive interpretation. If the smallest class does not comprise at least a few observations per parameter, or is redundant with another larger class, a simpler solution is preferable.

To assess classification accuracy, we turn to the third diagnostic table: The average posterior probabilities by most likely class membership, $P_{p_{(M=m)}}$, `class_prob(res, type = "avg.mostlikely")`: for participants assigned to one particular class (rows) according to M, this table gives the mean probability of being assigned to any class (columns). This table is obtained by calculating the column means of P for the k subsets of observations with most likely class of $M = 1 \ldots k$. The diagonal represents the average predictive probability of a the subjects related to their most likely class, indicating the certainty of class assignment for the subjects. Some guidelines have being suggested for this diagonal, like for them to be higher than 0.7 (**masyn2013?**), but we suggest researchers to be cautions about these guidelines as they cannot be generalized and should not be use as strict cutoffs to defined "good" or "bad" classification.

The table of classification probabilities for the most likely latent class M by latent class, $P_{p_{(M=m|C=c)}}$ is similar to the aforementioned table $P_{p_{(M=m)}}$, except that it accounts for classification uncertainty. If C is the true class of an observation, which is imperfectly measured, and M is the most likely class, then this table shows the probability of an observation being assigned to class m in M, given that its true class is c in C, or $p(M = m|C = c)$. Values on the diagonal can thus be conceptualized as the reliability with which each level of the categorical latent variable is measured by M, and off-diagonal elements can be conceptualized as measurement error. The minimum and maximum of the diagonal of this table are given by default as part of the output of `table_fit()` for LCA analyses (`prob_min` and `prob_max`). These probabilities are of particular interest because they conveniently summarize classification accuracy per class. Both probabilities should be high, as this means that all classes are reliably classified. If the minimum probability is low, there is at least one class with poor classification accuracy.

Finally, class separability can be summarized in a single statistic: the entropy criterion [REF]. In physics, high entropy refers to maximum randomness or uncertainty, and low entropy corresponds to a strict arrangement of the units of study. Applied to LCA, high entropy means that classes are completely indistinguishable; every individual has equal probability of being in any class. Low entropy, by contrast, means that classes are fully separated: every individual has a near-one probability of being in one class only, and near-zero probability of being in any other class. Crucially, in `tidySEM`, the entropy criterion is defined as 1-entropy (a convention originating in Mplus). Its interpretation is reversed; 0 means the model classification is no better than chance, and 1 means perfect classification. There are no rules of thumb for entropy, because entropy is also sensitive to the sample size (entropy tends to be lower in larger samples), number of latent classes (entropy tends to be lower for higher number of classes), and number of indicators. Like other classification diagnostics, entropy is not a model fit measure and should not be used for model selection (Masyn, 2013). However, it can be used to disqualify certain solutions if the goal is to identify clearly distinct classes.

Labelling Classes. Class names should be chosen to accurately reflect theoretically relevant characteristics of group membership. These class names should aid the interpretation of the class solutions and serve as a shorthand throughout the manuscript. For example, one study used a dual-trajectory LCGA of cognitive and affective empathy development with gender as a covariate. Although this model has dozens of parameters, the resulting classes were labeled low, average, and high empathy because level differences were the most distinguishing feature. However, overly simplistic and general class names can be misleading, a phenomenon known as the *naming fallacy* (Weller et al., 2020). For example, it would be misleading to call one group "low empathy" and another group "medium empathy" if the empathy differences were non-significant, or if other differences, like the effect of the covariate gender, were more substantial than level differences. One way to avoid the naming fallacy is comprehensively reporting model parameters, and explicating that class names are just a shorthand.

When labeling classes, it is important to note that LCA models are identified up to the class order. This means that the order of classes is arbitrary and may change if the analysis is replicated. A simple solution to ensure replicable results is setting a random seed before fitting the model, using `set.seed()`. This will ensure that the classes stay in the same order upon replication, but that may not be the preferred order for, e.g., interpreting the results. To address this problem, the function `mx_switch_labels()` can be used. By default, the function orders classes from largest to smallest class count - but classes can also be ordered by values of a specific parameter (e.g., the mean intercept in a latent class growth model). Note that label switching is a much bigger problem when using Bayesian estimation, where it can occur during model estimation. This is beyond the scope of the present paper.

Follow-up Analyses. LCA is rarely performed in isolation; often, researchers wish to relate class membership to auxiliary variables, or even auxiliary models. It is important to keep in mind that model-implied class membership is subject to classification error. Treating class membership as an observed variable in follow-up analyses disregards this error. The results will be biased, except when class separation is very high - as indicated by high (near 1) entropy and minimum posterior classification probability. It is therefore important to account for classification error.

There are several different ways to do follow-up analyses in latent class analysis (LCA) while accounting for classification error. At present, the so-called BCH method is considered one of the best approaches [after its creators, REF]. It has been implemented in `tidySEM`. The advantage of the BCH method is that it not only allows for the comparison of individual variables, but also entire auxiliary models across latent classes.

Best Practices in Reporting

Among studies using LCA, reporting practices vary significantly (Weller et al., 2020). Various authors have tried to improve and standardize ways of reporting LCA (e.g. Masyn, 2013; Weller et al., 2020), but more work is needed.

Open and Reproducible Science. Journals and funding bodies increasingly require research to be open and reproducible. For instance, many journals (and the APA) have adopted the TOP guidelines, which recommend comprehensive citation of literature, software, and data sources; sharing data and analysis code required to exactly reproduce analyses, and pre-registering study plans prior to data collection or analysis (**nosekTransparencyOpennessPromotion2016a?**). A key advantage of using open source software for LCA is that, compared to commercial software, it enables much greater adherence to open science principles. Sharing code has limited utility if it can only be evaluated in commercial software, as this restricts the ability to reproduce analyses to license holders, and prevents auditing research because the underlying source code is proprietary. The `tidySEM` package enables performing LCA in free open source software, thus increasing the utility of shared code.

Adopting FOSS makes it possible to conduct a fully reproducible LCA analysis compliant with best practices in open science. The Workflow for Open Reproducible Code in Science describes how to create a fully reproducible research archive, and the `worcs` R-package automates many of the steps involved (**vanlissaWORCSWorkflowOpen2021a?**). This workflow is based on three principles: First, the paper (or analysis section) is written as a *dynamic document* that contains both prose and analysis code. Results are automatically rendered as text, tables, and plots. Second, the research archive is *version controlled*, which means that the entire historical record of the project is retained. This version controlled archive can be made public. Third, *dependency management* ensures that all software used is documented, including its version number and provenance. Without dependency management, code might not work (for example, because it is not clear where certain functions come from), or its results may change when some dependencies are updated. By combining `tidySEM` and `worcs`, it is possible to conduct a fully reproducible LCA analysis compliant with best practices in open science.

An additional argument for sharing data in the context of LCA is that reproducing LCA analyses requires individual participant data, not just sufficient statistics like many other methods (i.e., a correlation or covariance matrix). If the original data cannot be shared, sharing synthetic data at least enables reproducing and reusing analysis code. The `worcs` package provides a generic, non-parametric method to create a synthetic copy of data when data cannot be shared. It is also possible to synthesize data directly from the LCA model using `mxGenerateData()`, which often provides a better basis for reproducing the LCA.

Preregistration. LCA analyses inherently involve some subjectivity, because of the potential for disagreement between criteria for class enumeration and the lack of objective fit indices. In our experience, reviewers (and authors) often disagree about which criteria should be used. Preregistering the class enumeration criteria as part of the analysis plan reduces disagreement and eliminates concerns about cherry picking fit measures after the results are known. It is particularly important to pre-specify the criteria that will be used to determine the maximum number of classes, because there is a risk that information criteria and likelihood ratio tests will suggest a larger number of classes, even if there are concerns about overfitting, classification accuracy, or model interpretability.

The aforementioned Workflow for Open Reproducible Code in Science facilitates creating a preregistration. More interestingly, it also enables creating a *Preregistration As Code* (PAC): A draft paper with reproducible analysis section based on fake (synthetic or simulated) data (**peikertReproducibleResearchTutorial2021?**). When the real data are accessed or collected, the analysis is simply reproduced with this new data file, and the results in the paper are automatically updated. PACs are quite unambiguous; the selection of the final model can even be automated by hard-coding the criteria into the analysis section. For an example of PAC, see (**vanlissaComplementingPreregisteredConfirmatory2022?**).

Reporting Model Fit. The function `table_fit()` reports an overview of fit indices, customized for LCA. In reporting model fit, it is important, firstly, to report the effective sample size, which should be the same for all models in order to compare relative fit measures. Secondly, it should be clearly reported, for example, in the strategy of analyses, which parameters are estimated across and within classes. In case of doubt, a full overview of the parameters is given by the `coef()` and `table_results()` functions. The number of parameters for each model should be included in the fit table. Thirdly, it is important that readers can assess the relative fit of different solutions. The most fundamental fit measure is the $-2LL$, as this is the quantity optimized during estimation. From these statistics, information criteria can be computed. By default, `table_fit()` includes the AIC, BIC, and saBIC. We suggest to report at least the BIC, as it is ubiquitous, grounded in theory, and performs well in class enumeration.

Classification diagnostics should be reported, as they may determine whether models are excluded from consideration. Of particular interest is the diagonal of either the average posterior probabilities by most likely class membership, or of the most likely class membership by true class membership. The former indicates the average probability that individuals belong to the class they were assigned to, and the latter indicates the probability that individuals are assigned to a class if they actually belong to that class. By default, `table_fit()` reports the latter as `prob_min` and `prob_max`. The entropy is also of interest as an overall measure of class separation. Finally, the range of proportions of cases assigned to each class is important; the minimum proportion of cases per class can be used to diagnose classes that are too small to be locally identified or practically relevant (see **depaoliMixtureClassRecovery2013?**). By default, `table_fit()` reports the minimum and maximum proportion of cases, based on most likely class membership. Fifth, the LMR or BLRT can be reported if it is part of the class enumeration strategy, using `lr_lmr()` and `BLRT()`. By default, `table_fit()` reports the VLM for subsequent models in the table.

Reporting results. Reporting LCA results involves reporting the class proportions, obtained via `class_prob(res, "sum.posterior")`, and model parameters, obtained via `table_results(res)`. The class proportions are obtained by standardizing the class weights: A k-class model has $k-1$ free class weights, and one weight constrained to 1 for model identification. The free weights are included in the model results, but are redundant with the aforementioned class proportions. Other model parameters may include means, (co)variances, regression coefficients (rarely), and thresholds for ordinal indicators. It is good practice to report the point estimate and confidence interval of these parameters. Although it is conventional to report p-values as well, note that the default p-values tests the hypothesis that the parameter is equal to zero. This hypothesis may be irrelevant in LCA when the focus is on parameter differences between classes. Reporting standard errors allows readers to compute p-values for other more informative hypotheses. Thresholds are more readily interpretable when converted to conditional item probabilities, using the function `table_prob()`. These indicate the probability of an item being endorsed given that the observation belongs to a particular latent (Geiser, 2012).

Inference. As mentioned before, the results table included null-hypothesis tests for all parameters by default. Whether these tests are informative depends on the study goals and scale of the data. For example, if LCA indicators are mean-centered, then the p-values of class-specific means indicate whether the mean in that class differs significantly from the sample average. It is also possible to test more informative hypotheses, using the function `wald_test()`. Equality-constrained informative hypotheses are specified using the `bain` syntax, and evaluated on the model object. This enables testing theoretical hypotheses - for example, if specific values for class-specific means had been hypothesized. The same function allows testing parameters for equality, either within or across classes.

Visualization. In an informal review of published graphics for LCA, we found that most visualizations focused on the estimated model parameters, displaying class means as points connected by lines, with point shape and linetype indicating class membership. Note that the lines introduce spurious information. Furthermore, they omit uncertainty about class means and do not illustrate how well the model describes the data.

The Grammar of Graphics addresses best practices in visualization, which we summarize here with specific emphasis on the application to LCA. Most importantly, good graphics start with the *data* in mind. Visualizing raw data allows one to assess class separability, and which aspects of the data are captured well or poorly by the model. A second consideration are which *aesthetics* to use. In a color plot, class membership can be indicated by color. In a black and white plot, it can be indicated by linetype and point shape. A third consideration is which geometric objects (geoms) to use. As mentioned before, common practice is to plot parameter estimates as points, and connect these points with lines. Points are suitable geoms to visualize point estimates and raw data. For continuous indicators, additional relevant geoms include error bars for standard errors and standard deviations, and density plots to visualize the distribution of raw data (smoothed with a kernel density function). In two dimensions, (co)variances can be visualized as ellipses. Finally, when plotting multiple class solutions, it makes sense to *facet* a plot into multiple panels by class solution. The plots in `tidySEM` are based on best practices as outlined in the grammar of graphics, and are implemented in `ggplot2`. As such, these plots can be further modified just like any other `ggplot2` object. One remaining limitation is that these best practices have only been implemented for LCA with continuous indicators, in `plot_profiles()`, `plot_density()` and `plot_bivariate()`, and for LCGA in `plot_growth()`. Plots for LCA with ordinal indicators are still rudimentary, more work remains to be done in this area.

Potential Pitfalls

Based on our experience in statistical consultations, there are a few common pitfalls and misunderstandings about LCA. First, perhaps the most common problem we see in consultations is treating ordinal data as continuous. As explained previously, LCA for ordinal indicators should be used for Likert-type scales and other data with few unique values. A second common pitfall is omitting the 1-class solution from class enumeration. If a one-class solution has better or similar fit to models with a higher number of classes, LCA may not be appropriate. A one-class solution should thus always be included as a benchmark. Third, although it is widely understood that classification diagnostics (e.g., entropy, posterior probability, class counts) are not class enumeration criteria, they still have a role in class enumeration. Specifically, they can be used to exclude solutions that do not meet predefined criteria from consideration. Fourth, previous reporting standards called for reporting of sample descriptive statistics (see **schreiberLatentClassAnalysis2017?**). While we agree that this is due dilligence, sample descriptives have two limitations in the context of LCA: they are not sufficient to check or reproduce the analysis, and many use cases of LCA assume that the population is heterogeneous, which limits the utility of whole-sample descriptives. These two limitations can be addressed by data sharing and reporting descriptive statistics for each latent class separately. Fifth, a recent publication mistakenly stated that an assumption of mixture models is that observed indicators are normally distributed (Spurk, Hirschi, Wang, Valero, & Kauffeld, 2020). This is incorrect, as evident from the shoe size example: When the number of classes is greater than one, Gaussian mixture models assume that the distribution of observed indicators is a *mixture of multiple normal distributions*. If the indicators were normally distributed, there would be no classes to extract as the whole population would belong to a single (homogeneous) class. Finally, the same paper suggested that robust standard errors should be used when indicators are not normally distributed (Spurk et al., 2020). This is also incorrect; to our knowledge, there is no literature on the merits of robust standard errors for LCA. Finally, we caution against step-wise class enumeration, whereby additional models are estimated until the selected class enumeration criteria no longer show improvement of model fit (e.g., see Spurk et al., 2020). This practice is problematic, because it is possible that an even higher number of classes would lead to a (significant) improvement in fit. Consider this thought experiment: there exist two groups of three relatively similar classes each (six in total). Step-wise class enumeration shows improvement in model fit going from a 1-class to 2-class model: the two classes capture the difference between the two groups of three classes. Then, no noteworthy improvement in model fit is found for adding more classes, until the 6-class solution is reached. In the step-wise approach, the analysis would be terminated after the 2-class model - even though the correct number of classes is 6.

SMART-LCA Checklist

The best practices discussed in this paper are summarized in the SMART-LCA Checklist: Standards for More Accuracy in Reporting of different Types of Latent Class Analysis.

1. Report item descriptives
    - Are all items on similar scales?
    - Is the measurement level correctly coded?
    - Any other reasons why the observed (multivariate) distribution could not be

described by the LCA (e.g., extreme skew)?

2. Report missingness
    - Proportion of missingness per variable
    - MAR test
    - Missing data method for handling missingness (typically FIML)
3. Report model specification
    - In exploratory LCA, compare all sensible alternative specifications
    - In confirmatory LCA, justify alternative models on theoretical grounds
    - Check that the observed (multivariate) distribution could theoretically be captured by the specified model (see #1).
4. Class enumeration
    - Justify the maximum number of classes K
    - In exploratory LCA, estimate 1:K classes
    - In confirmatory LCA, estimate the theoretical number of classes, a 1-class model, and optionally other numbers of classes for comparison
5. Justify the (preregistered) criteria used for class enumeration, which can include:
    - Information criteria
    - LR tests
    - Theoretical interpretability of the solution
6. Justify the (preregistered) criteria used to eliminate models from consideration, including:
    - Classification diagnostics
    - Local identifiability (acceptable number of observations per parameter in each class)
    - Model convergence
    - Theoretical interpretability of the solution
7. Report the following:

- Fit of all models under consideration
- Classification diagnostics for all models under consideration
- All parameters of the chosen model (means, (co)variances, thresholds)
- Proportion of cases per class

8. Provide an informative plot of the chosen solution
   - Show raw data
   - Show parameters
   - Show confidence bounds if possible
   - Show multivariate distributions for multivariate models, if possible
9. Account for classification inaccuracy in follow-up analyses
   - If entropy $> .95$ and posterior classification probability for each class $> .95$, this is less crucial

Conclusions

Write me

References

Akogul, S., & Erisoglu, M. (2016). A Comparison of Information Criteria in Clustering Based on Mixture of Multivariate Normal Distributions. *Mathematical and Computational Applications*, *21*(3), 34. https://doi.org/10.3390/mca21030034

Alkema, L. M., Van Lissa, C. J., Kooi, M., & Koelmans, A. A. (2022). Maximizing Realism: Mapping Plastic Particles at the Ocean Surface Using Mixtures of Normal Distributions. *Environmental Science & Technology*, *56*(22), 15552–15562. https://doi.org/10.1021/acs.est.2c03559

Baughman, A. L., Bisgard, K. M., Lynn, F., & Meade, B. D. (2006). Mixture model analysis for establishing a diagnostic cut-off point for pertussis antibody levels. *Statistics in Medicine*, *25*(17), 2994–3010. https://doi.org/10.1002/sim.2442

Collins, L. M., & Lanza, S. T. (2009). *Latent Class and Latent Transition Analysis: With Applications in the Social, Behavioral, and Health Sciences*. John Wiley & Sons.

Enders, C. K. (2022). *Applied Missing Data Analysis*. Guilford Publications.

Figueiredo, M. A. T., & Jain, A. K. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3), 381–396. https://doi.org/10.1109/34.990138

Geiser, C. (2012). *Data Analysis with Mplus*. Guilford Press.

Geiser, C., & Wurpts, I. (2014). Is adding more indicators to a latent class analysis beneficial or detrimental? Results of a Monte Carlo study. *Frontiers in Psychology: Quantitative Psychology and Measurement*, *5*. https://doi.org/https://doi.org/10.3389/fpsyg.2014.00920

Gromatsky, M., Edwards, E. R., Sullivan, S. R., Van Lissa, C. J., Lane, R., Spears, A. P., … Goodman, M. (2022). Characteristics of suicide attempts associated with lethality and method: A latent class analysis of the Military Suicide Research Consortium. *Journal of Psychiatric Research*, *149*, 54–61.

Hennig, C., Meila, M., Murtagh, F., & Rocci, R. (2015). *Handbook of Cluster Analysis*. 28.

Jamshidian, M., & Jalal, S. (2010). Tests of Homoscedasticity, Normality, and Missing Completely at Random for Incomplete Multivariate Data. *Psychometrika*, *75*(4), 649–674. https://doi.org/10.1007/s11336-010-9175-3

Jung, T., & Wickrama, K. a. S. (2008). An Introduction to Latent Class Growth Analysis and Growth Mixture Modeling. *Social and Personality Psychology Compass*, *2*(1), 302–317. https://doi.org/10.1111/j.1751-9004.2007.00054.x

Kollenburg, G. H. van, Mulder, J., & Vermunt, J. K. (2018). *The Lazy Bootstrap. A Fast Resampling Method for Evaluating Latent Class Model Fit*. 23.

Lee, T., & Shi, D. (2021). A comparison of full information maximum likelihood and multiple imputation in structural equation modeling with missing data. *Psychological Methods*, No Pagination Specified–No Pagination Specified. https://doi.org/10.1037/met0000381

Little, R. J. A. (1988). A Test of Missing Completely at Random for Multivariate Data with Missing Values. *Journal of the American Statistical Association*, *83*(404), pp. 1198–1202. https://doi.org/10.2307/2290157

MacGregor, A. J., Dougherty, A. L., D'Souza, E. W., McCabe, C. T., Crouch, D. J., Zouris, J. M., … Fraser, J. J. (2021). Symptom profiles following combat injury and long-term quality of life: A latent class analysis. *Quality of Life Research*, *30*(9), 2531–2540. https://doi.org/10.1007/s11136-021-02836-y

Maene, C., D'hondt, F., Van Lissa, C. J., Thijs, J., & Stevens, P. A. J. (2022). Perceived Teacher Discrimination and Depressive Feelings in Adolescents: The Role of National, Regional, and Heritage Identities in Flemish Schools. *Journal of Youth and Adolescence*, *51*(12), 2281–2293. https://doi.org/10.1007/s10964-022-01665-7

Marcia, J. E. (1966). Development and validation of ego-identity status. *Journal of Personality and Social Psychology*, *3*, 551–558. https://doi.org/10.1037/h0023281

Masyn, K. E. (2013). Latent Class Analysis and Finite Mixture Modeling. In *The Oxford Handbook of Quantitative Methods*: *Vol. 2: Statistical Analysis* (p. 551). Oxford University Press.

McElreath, R. (2020). *Statistical Rethinking | A Bayesian Course with Examples in R and STAN* (2nd ed.). Chapman; Hall/CRC. Retrieved from https://www.taylorfrancis.com/books/mono/10.1201/9780429029608/statistical-rethinking-richard-mcelreath

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*(2), 535–549. https://doi.org/10.1007/s11336-014-9435-8

Norman, G. (2010). Likert scales, levels of measurement and the "laws" of statistics. *Advances in Health Sciences Education*, *15*(5), 625–632. https://doi.org/10.1007/s10459-010-9222-y

Nylund, K. L., Asparouhov, T., & Muthén, B. O. (2007). Deciding on the Number of Classes in Latent Class Analysis and Growth Mixture Modeling: A Monte Carlo Simulation Study. *Structural Equation Modeling: A Multidisciplinary Journal*, *14*(4), 535–569. https://doi.org/10.1080/10705510701575396

Nylund-Gibson, K., & Choi, A. Y. (2018). Ten frequently asked questions about latent class analysis. *Translational Issues in Psychological Science*, *4*(4), 440–461. https://doi.org/10.1037/tps0000176

Pavlopoulos, D., & Vermunt, J. K. (2015). *Measuring temporary employment. Do survey or register data tell the truth?* 37.

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

Rubin, D. B. (1976). Inference and Missing Data. *Biometrika*, *63*(3), 581–592. https://doi.org/10.2307/2335739

Scrucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). Mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *The R Journal*, *8*(1), 289–317.

Spurk, D., Hirschi, A., Wang, M., Valero, D., & Kauffeld, S. (2020). Latent profile analysis:

A review and "how to" guide of its application within vocational behavior research. *Journal of Vocational Behavior*, *120*, 103445. https://doi.org/10.1016/j.jvb.2020.103445

Tsaousis, I., Sideridis, G. D., & AlGhamdi, H. M. (2020). Measurement Invariance and Differential Item Functioning Across Gender Within a Latent Class Analysis Framework: Evidence From a High-Stakes Test for University Admission in Saudi Arabia. *Frontiers in Psychology*, *11*. Retrieved from https://www.frontiersin.org/articles/10.3389/fpsyg.2020.00622

Van De Schoot, R., Sijbrandij, M., Winter, S. D., Depaoli, S., & Vermunt, J. K. (2017). The GRoLTS-Checklist: Guidelines for Reporting on Latent Trajectory Studies. *Structural Equation Modeling: A Multidisciplinary Journal*, *24*(3), 451–467. https://doi.org/10.1080/10705511.2016.1247646

Van Lissa, C. J., Hawk, S. T., Branje, S. J. T., Koot, H. M., Van Lier, P. A. C., & Meeus, W. H. J. (2015). Divergence Between Adolescent and Parental Perceptions of Conflict in Relationship to Adolescent Empathy Development. *Journal of Youth and Adolescence*, *44*(1), 48–61. https://doi.org/10.1007/s10964-014-0152-5

Vermunt, Jeroen K. (2010). Longitudinal Research Using Mixture Models. In K. Montfort, J. H. Oud, & A. Satorra (Eds.), *Longitudinal Research with Latent Variables* (pp. 119–152). Berlin/Heidelberg: Springer. https://doi.org/10.1007/978-3-642-11760-2_4

Vermunt, Jeroen K., Magidson, J., Lewis-Beck, M., Bryman, A., Liao, T. F., & Department of Methodology and Statistics. (2004). Latent class analysis. In *The Sage encyclopedia of social sciences research methods* (pp. 549–553). Sage. Retrieved from https://research.tilburguniversity.edu/en/publications/0caedd00-27c1-42bd-bb4e-d1dcb0864956

Visser, I., & Speekenbrink, M. (2010). depmixS4: An R package for hidden markov models. *Journal of Statistical Software*, *36*(7), 1–21. Retrieved from https://www.jstatsoft.org/v36/i07/

Weller, B. E., Bowen, N. K., & Faubert, S. J. (2020). Latent Class Analysis: A Guide to

Best Practice. *Journal of Black Psychology*, *46*(4), 287–311.
https://doi.org/10.1177/0095798420930932