UPGRADE MANUAL FOR LATENT GOLD BASIC, ADVANCED/SYNTAX, AND CHOICE VERSION 6.0[1]

JEROEN K. VERMUNT AND JAY MAGIDSON

(January 27, 2021)

[1] This document should be cited as: Vermunt, J.K., and Magidson, J. (2021). Upgrade Manual for Latent GOLD Basic, Advanced, Syntax, and Choice Version 6.0. Arlington, MA: Statistical Innovations Inc.

For more information about Statistical Innovations Inc. please visit our website at http://www.statisticalinnovations.com or contact us at

Statistical Innovations Inc.

7 Stevens Terrace

Arlington, MA 02476

e-mail: support@statisticalinnovations.com

Upgrade manual for Latent GOLD Basic, Advanced/Syntax, and Choice Version 6.0.

CONTENTS

1 INTRODUCTION

This manual describes the new features implemented in Latent GOLD in the transition from version 5.1 to 6.0, our first major upgrade in 8 years! The most important new features include estimation of **latent class *tree* models** in a fully automated manner, various extensions for *stepwise* latent class modeling, such as the ability to deal with **differential item functioning in three-step latent class analysis**, and various extensions to factor analysis with continuous latent variables, such as the newly developed **factor rotation** options. Other modeling extensions include the extension of log-linear scale models to apply to continuous variables and counts, new options for continuous-time Markov models, additional scale types for dependent variables, differential weighting of dependent variables, B-splines for numeric predictors covariates, and anchored best-worst choice models. New output options include user-specified functions of model parameters, expanded options for obtaining scoring equations to classify new cases, marginal effects, and goodness-of-fit measures similar to posterior predictive checks. Other new features include a more compact Syntax options for defining dependent and independent variables, estimating models for a range of classes in all modules including Syntax, and expanded Bayesian multiple imputation options.

The full set of new features are listed below, followed by sections describing each in detail. The Latent GOLD Syntax Examples, GUI Examples, Choice Syntax Examples, and Choice GUI Examples nested in Help contain examples of most of these new features in a separate entry named “New in LG 6.0”.

The new modeling options are:

- Latent class tree models (all modules)
- Extensions of bias-adjusted stepwise latent class models (Step3 and Syntax)
  - Accounting for differential item functioning (DIF) in step-3 models (Step3-Covariate and Syntax)
  - Bakk-Kuha two-step latent class modeling approach (Step3 and Syntax)
  - Step3 analysis using classifications from a step-1 LC model with a caseid (Step3 and Syntax)
  - Causal inference using three-step latent class analysis (Syntax)
  - Alternative classification option for step3 models (Syntax)
- Extensions of factor analysis models with continuous dependent variables (Syntax)
  - Single-group, multiple-group, and mixture exploratory factor analysis (EFA) with simple structure and agreement factor rotation options
  - EM algorithm for factor analysis models
- Implementation of log-linear scale models also for continuous and count variables (Syntax)
- Extensions of continuous-time (latent) Markov models (Syntax)
  - Alternative/better methods to compute matrix exponentials
  - Output of transition intensities and sojourn times
  - Priors for transition intensities
  - Specification of different type of event times
  - Hybrid discrete-continuous time models
- New scale types for dependent variables (Syntax)
  - Multivariate Poisson counts, including truncated and overdispersed variants
  - Multinomial counts, including truncated and overdispersed variants
  - Dirichlet
  - Continuous interval censored
- Differential weighting of indicators (Cluster, DFactor, and Markov) or dependent variables (Syntax)

- B-spline smoothers for modeling effects of independent variables (all modules)
- Anchored best-worst models (Choice with Syntax)

New output options are:

- Expanded options for obtaining scoring equations (Cluster, DFactor, Step3-Scoring, and Syntax)
- Expanded and simplified options for obtaining bootstrap p-values
- Vuong and Lo-Mendell-Rubin likelihood-ratio tests
- Marginal effects, or regression coefficient transformed to effects on probabilities or expected values
- Association statistics with a fast parametric bootstrap for (local) model fit assessment (Syntax)
- Best-worst scores per class and per individual (Choice)
- User-defined Wald tests specified with parameter labels (Syntax)
- Point and standard error estimation for user-defined expressions of model parameters (Syntax)
- Score tests for multiple (sets of) parameters simultaneously (Syntax)
- Standard error computation by simulating parameters (Syntax)
- Distinction between writing parameters and writing internal parameters to a file (Syntax)
- Improved and expanded outfile and ClassPred options (all modules)
- The variant “conditional” for the Prediction Statistics output (Step3-Dependent and Syntax)

The other new options are:

- Flexible specification of priors for model probabilities (Syntax)
- Expanded Bayesian imputation (Syntax)
- Simulation studies with random parameter values (Syntax)
- Special parameter specifications (Syntax)
    - Diagonal and off-diagonal parameters (~dia and ~off)
    - Expanded ~wei option
    - Expanded covariance structures for continuous dependent variables (~ar1 and ~cor)
- Flexible formatting of numbers when running in batch mode (all modules)
- Compact specification of list of dependent and independent variables (Syntax)
- Running models for a range of classes (Markov and Syntax)

2 LATENT CLASS TREE MODELS

A common strategy for estimating latent class (LC) models is to continue increasing the number of classes so long as say the BIC keeps declining. The model with the lowest BIC value is then retained as the final model. Two problems with this approach are 1) it often yields a final model that is difficult to interpret, in part because it contains too many latent classes, and 2) it is difficult to compare various solutions with different numbers of classes to see how they are related, and in particular, solutions with *fewer* classes than that suggested by BIC may be of interest.

To overcome these two problems, Van den Bergh, Vermunt, and colleagues proposed a new LC modeling approach called latent class tree (LCT) modeling. Similar to (divisive) hierarchical cluster analysis, it is based on a sequential algorithm that proceeds as follows:

1) Using the full sample at the root of the tree, a standard LC model is estimated with K “basic”, or “starting” classes, where K will typically be small (say 2, 3, or 4).
2) Subsequently, K *child* nodes are formed by splitting the sample based on the posterior membership probabilities from this K-class model.
3) For each of the newly formed child nodes, 1- and 2-class models are estimated.
4) If the 2-class model fits better than the 1-class model (say based on the BIC), the node concerned becomes a *parent* node with its own two child nodes obtained by splitting the sample at this node based on the posterior membership probabilities, and the algorithm returns to step 3. Otherwise, it goes to step 5.
5) The node concerned serves as an *end node*.

In their first proposal, Van den Bergh, Schmittmann, and Vermunt (2017) worked only with *binary* splits, implying that the LC model at the root node (step 1 above) was also a 2-class model. Later on, Van den Bergh, Van Kollenburg, and Vermunt (2018) showed that it might be better to use more than two classes at the root node, and proposed a new method to determine the value of K. Other relevant recent work on the LCT approach concerns its application in LC growth modeling (Van den Bergh and Vermunt, 2018) and in LC best-worst choice modeling (Magidson and Madura, 2018), and its combination with three-step LC modeling (Van den Bergh and Vermunt, 2019).

Latent GOLD 6.0 implements each of these LCT models, both via the GUI and the Syntax. The papers cited above provide the technical details for the sequential estimation algorithm, based on the work of Van der Palm, Van der Ark, and Vermunt (2016), and contain various illustrative applications. Regarding the internal implementation in Latent GOLD, it is important to note that the K-class model estimated at the root defines the structure of the models that will also be used at the next levels. That is, at each of the next levels of the tree, the same model is estimated, but now with the third and next classes restricted to be empty (to have class proportions equal to 0) and with case weights equal to the individuals’ (cumulative) posterior probabilities of belonging to the parent node concerned. This is how the K-class model used at the root is transformed into a model with 2 classes at the next levels of the tree.

Several final things to note regarding to the LCT implementation are:

- At each split, the latent classes are reordered according to their size (from large to small) to deal with label switching. This prevents obtaining a seemingly different tree when rerunning the same LCT model.
- In the Latent GOLD GUI modules, a LCT model is estimated behind the scenes using the Latent GOLD Syntax system. Thus, the output obtained (e.g., for Parameters) will have the Syntax type format.
- The coding for the tree latent variable is set to dummy first (or coding=first). This is done to assure that one obtains meaningful Parameters when K is 3 or larger. When estimating such models with the Syntax module, this coding can be suppressed using the “coding=…” option for the tree latent variable.
- In LCT models, the specified settings for the starting values procedure (number of start sets and iterations per set) are also applied when estimating the 2-class models at the parent nodes.
- Latent GOLD uses multiple-processing when estimating a LCT model; that is, each of the models to be estimated at a particular level of the tree is dealt with by one of available cores.

2.1 ESTIMATING A LC TREE MODEL WITH THE SYNTAX

The Latent GOLD Syntax contains a new keyword **tree** to indicate which **latent** variable is the one that should be used for splitting. The simplest example of a LCT model is:

```
latent Cluster nominal 3 tree;
```

When running this model, a LC Tree will be formed that splits at the root into 3 classes using the posteriors from a standard 3-class model. At the next levels, the splitting procedure continues using *binary* splits, and stops when one of the stopping criteria is reached (see tree options below).

It is also possible to construct the tree exactly as one wishes, that is, by specifying which nodes should be split. This is achieved with the option split=(list of nodes), for example,

```
latent Cluster nominal 3 tree split=(1 3 31);
```

This specification implies that Cluster 1 and 3 (Node 1 and 3) from the initial LC model should be split, as well as the first child node obtained when splitting Cluster 3 (Node 31).

An example of a model containing two dichotomous latent variables is:

```
latent DFactor1 nominal 2, DFactor2 nominal 2 tree;
```

At the root, a model with 2x2=4 classes is estimated. The splitting method and the model used at the next levels depends on whether the “joint” or the “marginal” method is used, where “joint” is the default approach (see the further description of these options below).

The keyword **tree** can also be used in the Latent GOLD **options** section to change the default settings used for the construction of the tree. These concern settings related to the stopping of the sequential splitting algorithm and to the approach used when the model contains more than one discrete latent variable.

Whether to split a node (yes or no) will typically be decided based on the increase in LL between the 1- and 2-class model. The minimally required LL difference can be defined by the user (option **minLLdiff=<number>**) or be derived from the BIC, AIC, or AIC3 (options **BIC**, **AIC** and **AIC3**). An alternative criterion that one can select for this purpose is the minimum entropy R-squared value threshold for the 2-class model to achieve in order to split into these 2 classes (**minentropyR2=<number>**).

Alternative criteria for *not* splitting can also be specified. These include when the sample at a node is too small (**minN=<number>**), when the largest BVR in the 1-class model is too small (**minBVR=<number>**), when a certain maximum number of levels is reached (**maxlevels=<number>**), or when a certain maximum number of nodes is reached (**maxnodes=<number>**). Note that these latter stop criteria differ from those discussed above in that these are evaluated before new 1- and 2-class models are estimated at the node concerned, except for the stop criterion based on the BVRs, which requires “estimating” the 1-class model.

The last tree option concerns the way the splitting and modeling at the next levels is performed when a model contains two or more discrete latent variables. When using the default option **joint**, at the root node the child nodes are formed by the joint classes obtained by crossing the multiple latent variables. Following the construction of the root level, for the next levels 2-class models are estimated and binary splits are performed (in fact, the number of categories is set to 2 for the tree latent variable and to 1 for the other discrete latent variables). When using the option **marginal**, the splitting is based on the marginal posteriors of the tree latent variable. The model which is estimated at each of the nodes is the original model with multiple latent variables, but with the number of categories of the tree latent variable set to 2.

The following is an example of the specification of the tree options, where we use the default options:

```
options
   <other options>
   tree BIC minentropyR2=0 minN=0 minBVR=0 maxlevels=5 maxnodes=500 joint;
```

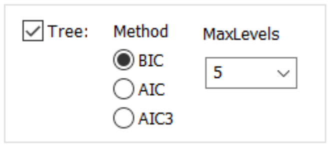
Note: In most applications, these default settings will suffice, meaning that for most applications there is no need to use the tree options subsection at all**.**

2.2 ESTIMATING A LC TREE MODEL WITH THE LATENT GOLD GUI MODULES

It is straightforward to specify a LC Tree model within the Latent GOLD Cluster, DFactor, Regression, and Choice modules. This works as follows:

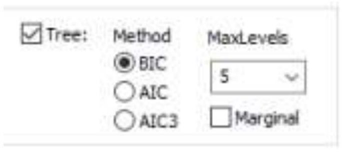
- Define the model to be used at the root of the LC Tree in the usual way using the Latent GOLD GUI, including the output, technical, and other options.
- Check the option **Tree** on the Model tab, and if desired, change the tree options BIC/AIC/AIC3, or maximum number of levels.
- Click on Estimate.

The LC Tree modeling options at the bottom left of the Model tab in Cluster, DFactor and Regression modules look like this:


☑ Tree: Method [● BIC] ○ AIC ○ AIC3 MaxLevels 5

The default settings are the same as those of LCT models run with Syntax, but the options are somewhat more limited.

For Choice models, one may use ***marginal*** instead of the default *joint* approach (see explanation above) when the model includes a second discrete latent variable affecting the scale (sClasses):


☑ Tree: Method [● BIC] ○ AIC ○ AIC3 MaxLevels 5 ☐ Marginal

Using the **marginal** option implies that splitting is done based on the Classes only, and at each next node a model with 2 Classes *and the specified number of sClasses* is estimated (for an example, see Magidson and Madura, 2018).

2.3 OUTPUT OBTAINED WHEN RUNNING A LC TREE MODEL

When running a LC Tree model, one obtains two global entries – the Tree Summary and the Tree Graph – which provide output for the entire tree, followed by detailed entries for each parent node that is split during the LCT model development and entries for the set of end nodes. More specifically, the output entries consist of the following:

- The **Tree Summary** table reports statistics computed for each node of the tree (each node corresponds to a row of this table) and describes the decisions made regarding splitting. Statistics presented can be added or removed from the table via a control pane which is obtained by right clicking on this output table. Each statistic corresponds to a column of this table.
- The **Tree Graph** depicts the tree diagram, with selected statistics listed as separate items in each node of the tree. The Tree Graph is interactive in the sense that items can be added or removed, and the size of various elements of the tree can be adjusted using the relevant plot control, which is activated by a right click on the graph. An additional interactive Tree Graph can also be obtained in a separate Window via the View menu. By left clicking on any parent node of this latter version of the graph, the program jumps to the associated text output which describes the child nodes formed by the split of this parent node.
- **Node - #:** Following the two global entries, separate detailed entries called **Node - #** are provided for each parent node that is split. Each of these entries contains a selection of statistics, as well as subentries which include Parameters and Loadings, Profile, Profile-Posterior, ProbMeans, and EstimatedValues-Model and/or EstimatedValues-Regression output. In Choice models, subentries also includes Attribute Parameters, Marginal Effects, Attribute Profile, and Importance output.
- For the **End Nodes**, one obtains Profile, Profile-Posterior, ProbMeans, and EstimatedValues-Regression output, and in Choice models also the output items listed above.

The Tree Graph can also be used to *prune* the tree. By selecting a node which is split, one can “hide” that split (right click on it and check the “hide” option), thereby reducing the number of end nodes. The output items for the end nodes will then be adjusted automatically, and thus match the Tree Graph. Note that the “hide” option can be selected using either version of the tree graph, and both versions are affected by a selection on either version.

In addition, one may request Classification-Posterior information to an output file for LC Tree models. The classification outfile will contain:

- The (local) posteriors at each of the nodes where a split occurred (variable names start with NodePost).
- The (cumulated) posteriors for the end nodes (variable names start with EndNodePost).
- The cumulated posteriors (or the weights) used for the estimation of the LC models at the nodes (variable names start with NodeWeight).

The (cumulated) posteriors for the end nodes (EndNodePost) sum to 1 and are, in fact, the posterior class membership probabilities of the final LC model consisting of the end-node classes. The (local) posteriors (NodePost) combined with the weights (NodeWeight) can be used for step-3 analyses at the nodes (see Van den Bergh and Vermunt, 2019).

The NodeWeight variables can be used to replicate the analysis performed at a specific node, for example, to request additional output. To facilitate this, Latent GOLD 6.0 implements the option “**Save Tree Models**” in the File menu. Depending on whether the LCT model was estimated with a GUI module or with Syntax, a lgf or lgs file will be created containing all models estimated (after the initial “Node-0” model) when creating the tree. These are models with 2 latent classes, in which the NodeWeight variable for the node concerned is used as **samplingweight** and in which the **samplesizeBIC** option is used to indicate that the original sample size should be used in the BIC computation (instead of the sum of the weights). Note if one wishes to save the estimated tree models with an associated data set containing the NodeWeight variables, one should run the tree model with the "outfile classification" option in Syntax or the "ClassPred Classification-Posterior" option in GUI models.

3 EXTENSIONS OF BIAS-ADJUSTED STEPWISE LATENT CLASS MODELS

3.1 STEP-THREE MODELS ACCOUNTING FOR DIFFERENTIAL ITEM FUNCTIONING OR MEASUREMENT NON-INVARIANCE

An important assumption of the bias-adjusted three-step LC analysis approach proposed by Vermunt and colleagues is that the variables used in the third step are conditionally independent of the indicators given class membership (Vermunt, 2010; Bakk, Tekle, and Vermunt, 2013). In practice, this assumption does not hold when there is item bias – which is also referred to as differential item function (DIF) or measurement non-invariance; that is, when one or more of the external variables used in the step-3 model has a direct effect on one or more of the indicators used in the step-1 model.

Latent GOLD 6.0 implements an extension of the step-3 approach which allows relaxing the no-DIF assumption. While its rationale is described in more detail in an accompanying paper (see Vermunt and Magidson, 2021), here we will explain the approach using a simple example. Suppose G is a covariate or independent variable (say gender or country) one would like to use in the step-3 LC model (in addition to other covariates and/or a distal outcome), but G has direct effects on the indicators used in the step-1 model. The new three-step LC approach that should be used in such a situation proceeds as follows:

1. Include G as a covariate in the step-1 LC model, where the necessary direct effects can be included in the model to account for the differential item function (DIF) across categories of G.
2. Perform the step-2 classification to an output file in the usual manner.
3. Use the new Step3-Covariate option to indicate which of the selected covariates are the DIF variables (with a right click on the covariates concerned). In Syntax, one should use the **<dif=variable name>** option in the step-3 analysis (see below).

As can be seen, the special provisions in Latent GOLD 6.0 concern the step-3 analysis, in which the user should indicate which independent variables (Syntax module) or covariates (Step3 module) are the variables causing the DIF. Using the posteriors in then input data file, the program will then setup a separate classification error matrix for each of the (combined) categories of the DIF covariates concerned. Denoting the latent class variable by X and the class assignments by W, in our small example this implies computing the probabilities P(W|X,G), which will subsequently be used in the step-3 model estimation with an ML or BCH bias adjustment. Note that in the standard no-DIF case, the bias-adjustment is based on P(W|X).

It is important to note that this new approach should only be used if the variable G causing DIF is included in both the step-1 and the step-3 analysis. However, one can think of situations in which G is included in the step-1 model but not in the step-3 model. For example, one might want to account for (slight) differences in the class definitions across say countries, but use only other external variables like age, education, and gender in the step-3 model. In those cases, one should **not** use the step-3 DIF option, but instead proceed in the usual manner.

In step3 Syntax models, the new option involves using the keyword **dif=<independent variable>** with a single DIF variable or **dif=(<list of independent variables>)** with multiple DIF variables. A sample Syntax is:

```
variables
    independent country nominal, age, gender nominal;
    latent Cluster posterior=(clu#1, clu#2, clu#1) dif=country;
equations
    Cluster <- 1 + country + age + gender;
```

Here, country is the variable which was used in the step-1 model to account for country differences in the cluster definitions. If gender was also included in the step-1 model, the syntax would be:

```
variables
    independent country nominal, age, gender nominal;
    latent Cluster posterior=(clu#1, clu#2, clu#1) dif=(country gender);
equations
    Cluster <- 1 + country + age + gender;
```

3.2 BAKK-KUHA TWO-STEP LC ANALYSIS, INCLUDING MODELS ACCOUNTING FOR ITEM BIAS

Bakk and Kuha (2018) proposed a new stepwise LC modeling approach for dealing with external variables. Their two-step method, which can be used as an alternative to three-step LC analysis, does not make use of classifications. Instead, it involves using the estimated step-one model parameters defining $P(\mathbf{y}|x)$ – the conditional response probabilities/densities given the classes – as fixed values in the step-two model in which the classes are related to external variables. Bakk and Kuha (2018) indicated how to estimate such a step-2 model with the Latent GOLD 5.1 Syntax using the option for defining fixed parameter values. In Latent GOLD 6.0 Step3 and Syntax, we implemented the Bakk-Kuha method in a more user-friendly manner.

Using the Bakk-Kuha method with the **GUI modules** works as follows:

1) In the Latent GOLD 6.0 Cluster, DFactor, Regression, Markov, and Choice modules, the ClassPred tab contains a new option for saving the class-specific logdensities from the step-one model, log $P(\mathbf{y}|x)$, to an output file. Thus, instead of saving the posterior classification probabilities, with this new method one should save the logdensities.
2) The Step3 module implements the Bakk-Kuha method. Check the new “Adjustment” method “Bakk-Kuha” and select the class-specific logdensities in the same manner as is done with the posteriors when applying the other adjustment methods. As with other stepwise approaches, the “Analysis” may concern investigating “Covariates” affecting the classes or “Dependent” variables having different means or probabilities across classes.

The implementation in the **Syntax**, is as follows:

1) The new outfile option **logdensity** allows saving the class-specific log $P(\mathbf{y}|x)$ values from the step-one model to an output data file. Note that the log $P(\mathbf{y}|x)$ values depend on the parameter estimates and the responses of the person concerned.
2) The saved log $P(\mathbf{y}|x)$ values can be used in the step-two model by using **logdensity=(<list of logdensities>)** in the definition of the latent variable(s) concerned. The step-two latent variable(s) can be used in the equations in the usual manner.

Actually, a Latent GOLD two-step LC analysis strongly resembles a three-step LC analysis which involves saving the posteriors to an output file and using these in the latent variable definition of the step-3 model.

Setting up a step-two model involves running a step-one Syntax model that includes the outfile statement:

```
outfile `step1logdensities.sav’ logdensity keep=gender age education;
```

where the “keep” option can be used to add the external variables to the output data file.

The step-two model uses `step1logdensities.sav’ as the data file. Here is sample Syntax of a step-two model with three covariates:

```
variables
   independent gender nominal, age, education nominal;
   latent Cluster logdensity=(logdens#1 logdens#2 logdens#3);
equations
    Cluster <- 1 + gender + age + education;
```

The two-step approach can also be used with distal outcomes (when the classes affect a dependent variable), as well as with more advanced step-two models such as models with multiple latent variables or latent Markov models (see Bakk and Kuha, 2018, and Di Mari and Bakk, 2018).

Contrary to the three-step approach, the Bakk-Kuha method can be used without any modification when the step-1 model contains direct effects to account for differential item function (see Janssen et al. 2019). The program will then automatically save the person-specific values of log $P(\mathbf{y}|x,\mathbf{z})$ to the specified output file and these saved log-densities can be used in the step-two model exactly as shown in the above example.

3.3 OTHER EXTENSIONS OF STEP3 MODELS

Another extension of the Step3 GUI module is the ability to specify a **Case ID** variable. This makes it possible to estimate a Step3 model when the step-one model is a Regression or Choice model for repeated measures; that is, when a Case ID is used in the step-one model. Note that this was already possible with Syntax, which allows defining a caseid variable.

Lanza, Coffman, and Xu, S. (2013) proposed using **inverse probability weights** (IPWs) for causal inference in LC analysis. Such an analysis can be performed in Latent GOLD by using the IPWs as sampling weights. In a recent paper, Clouth, Pauws, and Vermunt (2020) proposed a slightly different approach using three-step LC analysis. More specifically, they argue that it is more logical to estimate the step-one LC model that defines the latent classes without weighting, while IPWs should be used in the step-three model where the treatment effect is estimated. However, in the step-3 model the weights should not be used for the computation of the D matrix containing the P(W|X) probabilities, but only for the estimation of the treatment effect(s). The latter is achieved in Latent GOLD Syntax by indicating that the “sampling weight” is, in fact, an IPW. This can be achieved with the new option **ipw**. An example of its use is:

```
samplingweight IPWvalue ipw rescale;
```

Here, it is indicated that the weight “IPWvalue” is an IPW, and which is rescaled to sum to the total sample size.

The next extension concerns an option affecting the classification output for step3 Syntax models. By default, the assigned class membership W is not used to obtain the classifications in a step3 model, which in most situations is not so relevant anyway. That is, the W variable is ignored when computing posterior classification probabilities and classification statistics. This setting can be modified using the new step3 option **noignoreclassification**. An example of its use is:

```
step3 modal ml simultaneous noignoreclassification;
```

This setting indicates modal class assignments and an ML-type adjustment will be used in the step-3 model, and all equations will be estimated simultaneously (instead of one-by-one). The classifications obtained with the step3 model will be based on all information, including the original modal class assignments.

The WriteClassificationError and WriteClassificationErrorProportional output options allow writing to an output file the logits of the classification error probabilities P(W|X) and their covariance matrix. New is that when the filename contains the name of one of the independent variables, a separate set of logits is reported for each category of the variable concerned. This output can be used to setup a step-3 analysis with DIF by specifying the error logits as "fixed" parameters with known covariances, which allows accounting for the uncertainty in these "fixed" parameters (see section 5.15 of Upgrade Manual for Latent GOLD 5.1).

4 EXTENSIONS OF FACTOR ANALYSIS MODELS WITH CONTINUOUS DEPENDENT VARIABLES

4.1 FACTOR ROTATION AND SCALING OPTIONS FOR EXPLORATORY FACTOR ANALYSIS MODELS

The Latent GOLD 6.0 Syntax module implements factor rotation options which can be used to obtain interpretable and identified exploratory factor analysis (EFA) solutions. The approach is based on Jennrich (1973, 2002), which involves transforming the rotation criterion into a set of constraints on the model parameters. This approach makes it possible to obtain standard errors, z tests, and Wald statistics for the model EFA parameters. The rotation criteria to achieve a simple structure are varimax, oblimin, geomin, and target rotation.

Recent work by De Roover and Vermunt (2019) extended the Jennrich approach to multiple-group and mixture EFA. In such applications, the rotation criterion consists of two terms, one rotating to simple structure within groups and one rotating to agreement between groups. For the latter, one can use either the procrustes or the alignment criterion, but we recommend using the procrustes method because it turns out to be much more stable. Users need only specify the weight (w) to be assigned to the agreement criterion, and the weight of the simple structure criterion is then set to the remainder (1-w). De Roover and Vermunt (2019) provide the relevant technical information on this new approach, including details on the rotation algorithm, which is a combination of a gradient projection algorithm and a Fisher-scoring algorithm.

While in single-group EFA, factor variances should be fixed to 1, with multiple-group EFA, various alternative specifications can be used: factor variances can be fixed to 1 in all groups, be fixed to 1 in one reference group and free in the other groups, or be free in all groups. The last specification is the preferred one in multiple-group EFA in which one rotates towards agreement: it yields the closest agreement and, moreover, a solution which is unaffected by the choice of reference group. However, this requires an additional set of identification restrictions on the factor variances. Latent GOLD will restrict the factor variances to have a mean of 1 across groups.

Running an EFA model with factor rotation using the Latent GOLD Syntax involves:

- Specifying the rotation settings in the **rotation** subsection in **options**

- Specifying an unrestricted FA model in **equations**

The keywords for the **rotation** subsection in options are:

- Simple structure: **varimax**, **oblimin**, **geomin**, or **target=<filename>**
- Agreement: **procrustes=<number>** or **alignment=<number>**
- Other options: **epsilon=<number>**, **iterations=<number>**

With target rotation, the target structure should be specified in a text file. Typically, only the 0 loadings are relevant for the target, but also other values of target loadings may be used. Use of either the missing value code "." or the number -99 indicates that the loading concerned does not contribute to the target criterion. A separate target can be specified for each group (they appear stacked in the file), but when the file contains the target only for one group, it is assumed that the target is the same for all groups.

With the procrustes and alignment agreement criteria, a number between 0 and 1 should be specified, which denotes the weight (w) assigned to the agreement term. The weight of the simple-structure term equals 1-w. When using one of the agreement options in combination with free factor variances in all groups, Latent GOLD will restrict the factor variances to have a mean of 1 across groups.

The option epsilon refers to the small number appearing in the formula of the alignment criterion, which by default equals 1.0e-12. The option iterations can be used to increase the maximum number of Fisher-scoring iterations used for the rotation. The default is 25.

Here is sample Syntax for a single-group EFA model with 2 correlated factors:

```
options
   <other options>
   rotation oblimin;
variables
   <specification of variables in the model>
equations
   (1) F1 ;
   (1) F2 ;
   F1 <-> F2;
   Y1 – Y10 <- 1 + F1 + F2;
```

and here is an example of a multiple-group correlated 2-factor EFA with country being the grouping variable:

```
options
   <other options>
   rotation oblimin procrustes=0.5;
variables
   independent country nominal;
   <specification of other variables in the model>
equations
   F1  | Country;
   F2  | Country;
```

```
F1 <-> F2 | Country;
Y1 – Y10 <- 1 | Country + F1 | Country + F2 | Country;
```

4.2 AN EM ALGORITHM FOR FACTOR ANALYSIS FOR CONTINUOUS VARIABLES

Various authors have proposed estimating factor analysis models using the EM algorithm, including updating the FA parameters in the M-step of mixture factor models (see, for example, McLachlan and Krishnan, 2007). Latent GOLD has always used a Fisher-scoring algorithm for this purpose, which also remains the default method in Latent GOLD 6.0. However, as shown in a recent paper by De Roover, Vermunt, and Ceulemans (2021), the Fisher-scoring approach may become extremely slow when the FA model contains a large number of parameters.

In the appendix of their paper on mixture multiple-group factor analysis, De Roover, Vermunt, and Ceulemans (2021) describe a rather general EM algorithm for mixture and/or multiple-group FA, which can be further extended to include factor means (De Roover, 2021). This algorithm in now also available in Latent GOLD Syntax, where it is activated using the algorithm option **emfa**; an example is:

```
algorithm tolerance=1e-008 emtolerance=0.01 emiterations=250 nriterations=50 emfa;
```

When using this EM algorithm, one should set the Bayes constant for the residual variances to 0; that is, use

```
bayes categorical=1 variances=0 latent=1 poisson=1;
```

5 LOG-LINEAR SCALE MODEL FOR COUNT AND CONTINUOUS DEPENDENT VARIABLES

The Latent GOLD 5.0 and 5.1 Syntax allows defining log-scale models for the different types of categorical dependent variables (dichotomous, ordinal, nominal, binomial, choice, ranking, and best-worst). In Latent GOLD 6.0, this option is also implemented for continuous and count dependent variables.

As shown in the technical guide, using a log-scale model implies that the linear term η of the regression model for the response variable concerned is multiplied by the exponent of another linear term, which may itself contain effects of independent and latent variables. For example, with two predictors z_1 and z_2, the scale factor model could have the following form $\exp(d_1z_1+d_2z_2)$. For a continuous dependent variable, the expected values will equal $\eta\cdot\exp(d_1z_1+d_2z_2)$, while in a log-linear Poisson model they will equal $\exp[\eta\cdot\exp(d_1z_1+d_2z_2)]$.

In the Syntax equations, the log-scale model is defined using “<<-” between the dependent variable at the left-hand side and the linear term with the latent and independent variables at the right-hand side. All options available for regression equations can also be used with scale factor equations, such as conditional effects, interactions, and parameter constraints, except for most of the “~” parameter options. An example of a scale factor equation is:

```
response <<- Class + gender;
```

Since the parameters reported are log-linear, one will typically need to exponentiate the resulting estimates prior to interpreting them.

6 EXTENSIONS FOR CONTINUOUS-TIME (LATENT) MARKOV MODELS

Various extensions have been implemented for continuous-time latent Markov models, which are available via the Latent GOLD 6.0 Syntax. These include a better method for dealing with matrix exponentials, additional output, and various new modeling options.

6.1 MATRIX EXPONENTIALS

The most important extension/improvement is the change in the manner in which the matrix exponentials are computed. These are needed to obtain the transition probabilities and their derivatives from the transition intensities. Thus far, we have always used a method based on an eigenvalue decomposition, which is the fastest approach, but which may fail when the transition intensity matrix concerned has eigenvalues which are equal to one another or which are complex (see, Vogelsmeier et al., 2019). Various alternative approaches have been proposed, and Latent GOLD 6.0 implements two of these; that is, the Padé approximation and the Taylor approximation, in both cases with scaling and squaring (Brančík, 2006, 2008; Moler and Van Loan, 2013). The type of method can be specified using the new **algorithm** option **expm=[eigen|pade|taylor]**. An example of its use is:

```
algorithm tolerance=1e-008 emtolerance=0.01 emiterations=250 nriterations=50 expm=pade;
```

However, in practice, Latent GOLD users need not worry about this setting. By default, Latent GOLD 6.0 will use the faster “eigen” method, but automatically switch to “pade” for intensity matrices for which “eigen” fails. This hybrid “eigen-pade” approach turns out to work very well in practice.

6.2 INTENSITIES AND SOJOURN TIMES

Nested within the EstimatedValues output, in continuous-time latent Markov models, a new output section **Intensities/Sojourns** is provided. It displays the transition intensities and the sojourn times for all values of the independent and discrete latent variables. Note that the sojourn time for a particular state -- the expected length of stay in the state -- equals the negative of the inverse of the corresponding diagonal element of the intensity matrix.

6.3 PRIOR FOR TRANSITION INTENSITIES

Latent GOLD 6.0 contains an experimental implementation of a gamma prior for the transition intensities, which is aimed at preventing that these become exactly 0 (their boundary value). This option is activated using the **ct=<number>** statement as part of the **bayes** options.

6.4 ALTERNATIVE OBSERVATION TIMES

The keyword distinguishing continuous-time models from discrete-time models is **timeinterval**, which is used to specify the variable containing information on the length of the interval between consecutive observations. As part of “timeinterval”, one can now use the additional option **obstype=<variable name>**. While by default the data are assumed to represent snapshots of the process at arbitrary times, observations can also represent exact times of transition (as in survival analysis), exact death times (the time entering the absorbing state concerned, but from which type of origin state being unknown), or a mixture of these. With the observation-type option, one can indicate what kind of observation it concerns, where a 1 indicates a time interval of the snapshot, 2 an exact transition time, and 3 an exact death time. The likelihood contribution differs depending on the type of time interval variable (Jackson, 2011). An example of the use of this option is:

```
timeinterval exposure obstype=type;
```

where “type” is the variable in data set indicating the observation type.

6.5 HYBRID DISCRETE-CONTINUOUS TIME MODELS

The last new feature implemented in continuous-time latent Markov models is to indicate that one or more of the nominal or ordinal dynamic latent variables should be modeled as in a discrete-time model; that is, with transition probabilities modeled using standard logit equations. This can be indicated using the option **dt** (meaning discrete time) for the dynamic latent variables concerned. An example is:

```
latent BState nominal dynamic 2 dt, WState nominal dynamic 3;
```

7 B-SPLINE SMOOTHERS

B-spline (basis spline) smoothers allow for flexible modeling of the relationship between quantitative predictors and outcome variables (de Boor, 1978). In Latent GOLD 6.0, this is implemented as an option for independent variables in Syntax models and for predictors and covariates in the point-and-click GUI modules. In choice models defined using the Syntax, the option is also available for Attributes (alternative-specific predictors). The B-spline option is especially useful for modeling the time dependence in (latent class) growth models (see Francis, Elliot, and Weldon, 2016).

In Latent GOLD 6.0 Syntax models, the keyword **bspline** indicates the effects of the independent variable concerned should be modeled using B-spline functions. In addition, one can specify the degree and either the number of knots or the location of the knots using the commands **degree=<number>** and **knots=<number of inner knots>** or **knots=(<knot locations>)**. By default, the degree equals 3 and the number of inner knots equals 0, yielding a cubic polynomial. When specifying the number of inner knots, these will be equally spaced between the lowest and the highest observed value of the independent variable concerned, which serve as the outer knots. It is also possible to specify the locations of all knots, in which case one should also provide the outer knots, in which case the total number of knot locations specified equals the number of inner knots plus two.

The number of parameters associated with a B-spline smoother equals its degree plus its number of inner knots. These B-spline parameters are reported by Latent GOLD in the Parameters output. In the output part describing the variables used in the analysis, one can find the B-spline functions themselves, which are in fact quantitative predictors derived from the observed values of the independent variable concerned.

Sample Syntax illustrating the use of the bspline option is:

```
independent
  time bspline degree=2 knots=4;
```

As can be seen, the “bspline” option is used instead of the “nominal” or “numeric” attribute for independent variables. An example with user-defined knot locations is:

```
independent
  time bspline degree=3 knots=(0,5,10,20);
```

Also in GUI models, the B-spline option is available for independent variables, which depending on the module or their role in the model are referred to as Covariates or Predictors. This option is activated by setting their scale type to B-spline (instead of Nominal or Numeric), and specifying the degree and the number of inner knots if one wishes to override the default values of 3 and 0, respectively.

A final remark to be made about the B-spline option is that it can be combined with the "+" and "-" parameters restrictions for specifying monotonicity of effects. For example, one may model a time effect in a latent growth model using B-splines and at the same time indicate that this time effect is monotone (either increasing or decreasing) for some or all latent classes.

8 NEW SCALE TYPES FOR DEPENDENT VARIABLES

Latent GOLD 6.0 Syntax implements several new distribution types for dependent variables, which share the requirement that multiple variables are specified from the data file. These are:

- multivariate Poisson counts, including truncated and overdispersed versions (**mpoisson**),
- multinomial counts, including truncated and overdispersed versions (**multinominal**),
- Dirichlet (**dirichlet**), and
- interval censored continuous normal (**censored upper=<variable name>**).

The multivariate Poisson distribution is used for multiple (say M) count variables, and is equivalent to multiple independent Poisson variables. This equivalence disappears when using the truncated and/or overdispersed option. A zero-truncated model assumes that the sum of the M counts is at least 1, rather than each count being at least 1. The overdispersed model yields what is typically called a negative-multinomial model, which is an extension of the negative-binomial model for a single count. As in the univariate Poisson model, log-linear regressions are used for the M counts. An example of a truncated multivariate Poisson model specification is:

```
dependent (count1 count2) mpoisson truncated;
```

As can be seen, the M variables containing the counts should be inserted between braces, and the keyword specifying the scale type is **mpoisson**.

The multinomial distribution for counts is a well-known generalization of the binomial distribution, in which the number of categories M is larger than 2. In its truncated version, it is assumed that the sum of the counts for the first M-1 categories is at least 1. The overdispersed version yields the Dirichlet-multinomial model, which is an extension of the beta-binomial model for binary counts. The probabilities of the multinomial distribution are modelled using multinomial logistic regression models. An example is:

```
dependent (count1 count2) multinomial exposure=total;
```

Note that one specifies the number of occurrences of the first M-1 categories, as well as the exposure variable containing the total number of trials. As can be seen, the M-1 count variables are inserted between braces.

The Dirichlet distribution is a generalization of the beta distribution, where the number of categories M is larger than 2. It can be used to model compositional data; that is, a set of variables with positive values which sum to 1. The regression model used is a multinomial logistic model. An example is:

```
dependent (comp1 comp2) dirichlet;
```

Note that as in the multinomial model, one specifies the values for the first M-1 categories. The last value is obtained from the requirement that the M values sum to 1. The keyword is simply **dirichlet**.

The interval censored normal distribution is a generalization of the normal distribution with left censoring at the value 0. Interval censoring allows for left censoring, right censoring, and a combination of these, as well as censoring at any value. It requires specifying two variables from the data file, which indicate the lower and upper bounds between which the actual value lies. A missing value implies there no censoring at that side, while using equal values implies the observation is not censored. The specification in Latent GOLD 6.0 is:

```
dependent ylow continuous censored upper=yhigh;
```

Thus, the lower bound is specified as the dependent variable, the **censored** option is used, and the variable containing the upper bound of the interval is specified using the keyword **upper**.

9 WEIGHTING OF INDICATORS OR DEPENDENT VARIABLES

A new option implemented for indicators in Cluster, DFactor, and Markov models and for dependent variables in Syntax models is differential weighting. This may be useful if one wishes to downweight the influence of a particular (set of) variable(s) relative to the other variables. For example, in a Cluster model with both continuous and categorical indicators, one may wish to reduce the influence of the continuous indicators on the clustering, which may be achieved by assigning a lower weight to them, say .5 instead of 1. The likelihood function used to implement this type of weighting is similar to the one used with replication weights; that is, the class-specific densities are raised to the power defined by the weight.

In Cluster, DFactor, and Markov models, the variable weight can be specified using the **Var Weight** option which appears with the right-click on the dependent variable(s) of interest. In Syntax models, one uses the option **varweight=<number>**. An example of a Syntax model with 4 dependent variables is:

```
dependent y1 continuous varweight=.5, y2 continuous varweight=.5, y3 nominal, y4 nominal;
```

As can be seen, the two continuous dependent variables get a lower weight than the two nominal indicators.

10 ANCHORED BEST-WORST MODEL

Best-worst (or MaxDiff) experiments are sometimes supplemented with questions regarding the relevance of the alternatives for the respondent (Orme, 2009; Lattery, 2010). This dual response format involves following up each MaxDiff question (task) with another question asking something like: Considering the features shown above, would you say that...

1) **All** are important
2) **Some** are important, some are not
3) **None** of these are important

We will refer to this question as the *anchor* item, with response categories 1, 2, and 3, corresponding to all, some, and none are important, respectively. This information can be used to anchor the worths; that is, to obtain worths indicating *absolute* preferences instead of just *relative* preferences. This option requires users to create an additional alternative, here referred to as the "none" alternative, which is added to the list of alternatives. When setting up the MaxDiff model, Latent GOLD will automatically account for the anchor item as follows:

1) When *all* are important, the "none" alternative is treated as an additional worst choice out of a set containing this additional alternative.
2) When *some* are important, the best and worst choices are made from a set which contains "none" as an additional alternative.
3) When *none* are important, the "none" alternative is treated as an additional best choice out of a set containing this additional alternative.

To make this work with the **1-file data format**, users should expand their choice sets with the *none* alternative as the *last* option. With the **3-file data format**, the alternatives file should contain the *none* alternative, and each set should have this alternative as the *last* option. The anchored best-worst model can be run using **anchoritem=<varname>** in the definition of the bestworst type dependent variable.

Here is a sample Syntax:

dependent choice bestworst anchoritem=*important*;

where "important" is the variable in the data file indicating the response on the anchor item.

Examples with both the 1-file and 3-file format are provided in Tutorials ANCHOR1file and ANCHOR3file.

11 NEW OUTPUT OPTIONS

11.1 EXPANDED OPTIONS FOR OBTAINING SCORING EQUATIONS

Latent GOLD 6.0 expands the options to obtain scoring equations for the classification of new observations in three important ways:

- The scoring equations output option which was already available for Cluster models in version 5.1, can now also be obtained for DFactor models and for Cluster- and DFactor-like Syntax models. In DFactor models and other models with multiple discrete latent variables, one obtains the scoring equations for the joint posterior probabilities of all latent variables simultaneously. In Cluster and DFactor models, this option involves checking "Scoring Equations" on the Output tab. In Syntax models, one has to use the new output option **ScoringEquations**.
- The reported scoring equations for Cluster, DFactor, and some Syntax models are expanded to contain coefficients for missing values. This implies that the scoring equations can also be applied when the new observations one wishes to classify contain missing values on one or more indicators. Note that the missing-value coefficients could already be obtained with the Step3-Scoring option, but only for indicators having missing values in the original analysis. The newly implemented approach does not have this limitation.

- Not only in Step3-Scoring, but also in Cluster, DFactor, and certain Syntax models, it is possible now to save the scoring syntax in SPSS or generic format. In addition, the scoring syntax can now also be stored as R code. In Cluster, DFactor, and Step3 models, this new option is available on the Output tab. In Syntax models, one can use the output options **WriteSPSSSyntax=<filename>, WriteRSyntax=<filename>**, and **WriteGenSyntax=<filename>**.

Technical details on how the scoring equations are computed can be found in Vermunt and Magidson (2016). It should be noted that these options are available only when no ID variables are specified, thus not in Regression, Markov, and multilevel models. Moreover, the scoring equations options are not available in models with continuous latent variables.

11.2 EXPANDED AND SIMPLIFIED PROCEDURES FOR OBTAINING BOOTSTRAP P-VALUES

Latent GOLD allows obtaining bootstrap p values for chi-squared goodness-of-fit and likelihood-ratio (-2LL Diff) tests. In GUI models, this involves selecting an estimated model after which one can select the "Bootstrap Chi^2" or "Bootstrap -2LL Diff" option from the Model menu (or from the menu which appears with a right click). In Syntax, this requires using the montecarlo option L2, allchi2, or LLdiff='H0 model name', which yields bootstrap p-values for the L-squared, for all chi-squared statistics including the BVRs, and for the likelihood-ratio test comparing the current model with the specified H0 model. In Latent GOLD 6.0, the way in which one can obtain bootstrap p values has been extended and simplified.

New in ***GUI modules*** is the Output option "Bootstrap Chi^2 & -2LL Diff". When this option is checked, the program computes bootstrap p-values for all chi-squared statistics, including the bivariate residuals. Moreover, if one requests models for a range of classes, the program will also compute bootstrap p-values for the likelihood-ratio tests comparing consecutive models. The latter test is often referred to as the bootstrap likelihood-ratio test (BLRT; McLachlan and Peel, 2000; Nylund, Asparouhov, and Muthén, 2007). For example, if a model indicates the number of Clusters to be "1-4", bootstrap p-values will be obtained for the comparison between the 1 and 2 Cluster model, the 2 and 3 Cluster model, and 3 and 4 Cluster model.

The Technical tab contains a new option to specify the number of "Random Start Sets" to be used when performing the bootstrap. This option may be useful for the -2LL Diff bootstrap, where the H1 model with K+1 classes is estimated using data generated under the H0 model with K classes. By default, Latent GOLD uses the estimates from the K+1 class model as starting values.

New in the ***Syntax module*** is the possibility to use montecarlo option LLdiff without specifying the "H0 model name" (the model to compare the current model with). This new feature can be used in conjunction with the new Syntax feature for running models for a range of latent classes. More specifically, by running models for a range of classes and listing the keyword LLdiff among montecarlo options, one obtains bootstrap p-values for the -2LL Diff tests comparing consecutive models. Sample Syntax illustrating this feature is:

```
options
   <other options>
   montecarlo sets=16 replicates=500 LLdiff;
variables
   dependent Y1, Y2, Y3, Y4, Y5;
   latent  Cluster nominal 1:4;
```

```
equations
    Cluster  <- 1;
    Y1 – Y5 <- 1 + Cluster;
```

Note that here “1:4” means that models with one to four classes should be estimated (see also subsection “Running models for a range of classes”).

The bootstrap p-values for the L^2 and -2LL Diff will not only appear in the Model output, but also in the Summary table containing a selection of statistics for the models estimated with the data set concerned.

11.3 VUONG AND LO-MENDELL-RUBIN LIKELIHOOD-RATIO TESTS

Vuong (1989) proposed three types of tests for comparing pairs of alternative models when the standard asymptotic likelihood-ratio testing approach is not valid, such as when the two models are not nested or when the H1 model cannot be assumed to be correct. These tests concern: 1) a robust likelihood-ratio (-2LL Diff) test for nested or overlapping models, 2) a test for the distinguishability of two models based on the variance of the individual likelihood ratios, and 3) a z-test for the expected LL difference for non-nested models. Merkle, You, and Preacher (2016) and Schneider, Chalmers, Debelak, and Merkle (2020) give excellent descriptions of these three Vuong tests, and illustrate their application in the context of structural equation and item response models.

The first and most important test in the context of latent class and mixture models is the robust likelihood-ratio test for comparing nested or overlapping models. This test is sometimes referred to as the Vuong-Lo-Mendell-Rubin (VLMR) test, because Lo, Mendell, and Rubin (2002) proposed using this Vuong test for comparing models with different numbers of latent classes. These authors also proposed an adjusted version of this test, which is usually referred to as the aVLMR test (see, e.g., Nylund, Asparouhov, and Muthén, 2007). The sampling distribution of the VLMR and aVMLR statistics takes the form of a weighted sum of chi-squared random variables with 1 degree of freedom, with weights equal to the eigenvalues of a matrix Vuong denotes by **W**. **W** is obtained using first and second derivatives of the maximized log-likelihood functions under the H0 and H1 models.

The second test is the test for the distinguishability of the two models being compared. The relevant statistic is the variance of the individual log-likelihood differences, denoted by ω. More specifically, the sample value of ω times N follows a weighted sum of chi-squared distributions, with weights equal to the *squared* eigenvalues of **W**. If the null hypothesis that ω=0 is rejected (i.e., if the models are distinguishable), we may compare the two models via the non-nested likelihood-ratio test described next. Otherwise, we should use the robust test for nested or overlapping models described above.

The third test is a likelihood-ratio test for non-nested distinguishable models. Assuming H1 has a higher log-likelihood value than H0, it is a test on whether the H1 model performs better than the H0 model. The statistic used is the mean of the individual log-likelihood differences, which equals -2LL Diff divided by 2*N. Under the hypothesis that both models perform equally well, this statistic follows a normal distribution with a mean of 0 and a variance equal to ω/N. Latent GOLD reports the resulting z-value and the corresponding one-sided p-value.

Both in GUI and in Syntax models, the three Vuong tests are reported automatically when running models for a range of classes. So, there is no need to request this output if one wishes to compare models with K and K+1 classes.

In GUI models, Vuong tests may also be requested by selecting an estimated model after which one can select the “Vuong” option from the Model menu (or from the menu which appears with a right click). A list of possible H0 models will be shown. An advantage of this approach compared to running models for a range of classes is that local maxima can be prevented since both the H1 and the H0 model will be re-estimated with the starting values yielding the solutions of the estimated models. Moreover, the approach is more flexible in that any two models estimated with the same set of variables can be compared.

In Latent GOLD Syntax, it is possible to compare any pair of models using the new output option Vuong=‘H0 model name”, which works in a similar manner as the LLdiff=‘H0 model name’. Both models are estimated, and the Vuong Likelihood-Ratio Test output is obtained in the calling model.

The Vuong likelihood-ratio tests are reported in the Model output for the H1 model. Latent GOLD reports the LL value of the H0 model, the difference in number of parameters, the mean and the standard deviation of the distribution of the VLMR statistic, the VLMR and aVLMR values and their p-values, the value of ω and its p-value, and the Z-test for mean difference in LL value for non-nested models.

The information on the VLMR test for nested or overlapping models also appears in the Summary table, which contains a selection of statistics for the estimated models.

11.4 MARGINAL EFFECTS

While regression models for discrete variables yield effects on a logit or probit scale, and regression models for transition intensities or Poisson counts yield effects parameters on a log-linear scale, for interpretability one may instead prefer additive effects on a probability or otherwise linear scale. Such effects are typically referred to as *marginal effects*, which are partial effects on probabilities or expected values (Long, 1997; Long and Freese, 2014). Latent GOLD 6.0 Syntax implements marginal effects for all types of regression parameters, including the logit coefficients for covariate effects on latent classes or transition probabilities.

A marginal effect can be defined as the first-order derivative of the expected value (or probability) of interest with respect to the predictor concerned; that is, as dE(y)/dx. Note that in our case x can be an independent or a latent variable, and y a dependent or a latent variable. Various choices need to be made when computing marginal effects, and these choices affect the interpretation of the resulting marginal effects; that is:

1) In non-linear regression models, the value of a marginal effect depends on the values of E(y), or, equivalently, on the values of the predictors determining E(y). A common way to deal with this issue is to compute the marginal effects for each observation and average these across observations. This is what Latent GOLD does, yielding what is usually called *average marginal effects*. When a latent class variable serves as a predictor, in the computation of the marginal effects we use the posterior membership probabilities as weights for the contribution of an observation to the different latent classes.
2) The definition of a marginal effect as the first derivative of y with respect to x makes less sense for categorical predictors. In that case, it is preferable to compute E(y) for each category of the predictor concerned and subsequently compute the difference with respect to either the reference category or the average across categories, depending on whether dummy or effect coding is used (Bartus, 2005). This is what Latent GOLD does for main effects of categorical predictors. Since this approach becomes tedious for interaction terms containing categorical predictors, we use standard derivatives for interactions.

However, instead an interaction with a categorical predictor one can use a conditional effect, which may yield more meaningful marginal effects (see next item).

3) When using the conditional clause "|", one obtains separate regression parameters for each of the categories of the conditioning variable(s) (say for each latent class in a mixture regression model). Latent GOLD will then also report separate marginal effects for each category of the conditioning variable(s) (say for each latent class). While an alternative is to provide the overall marginal effects, those can easily be obtained by averaging the reported conditional marginal effects.
4) For ordinal regression models, it is possible to obtain marginal effects for each category of the dependent variable concerned. However, Latent GOLD reports the marginal effects with respect to the mean since this is more consistent with the fact that in ordinal regression models each predictor has a single effect parameter rather than a separate effect parameters for each category (as in nominal regression models).
5) In Choice modeling, the specified regression model will typically contain attributes, which are predictors which vary across alternatives (across categories of the dependent variable). Their marginal effects vary not only between observations but also between alternatives. Marginal effects for attributes are therefore obtained by averaging over both observations and alternatives. In ranking and best-worst models, only first choices are used when computing marginal effects.

The **output** option **marginaleffects** is used in Latent GOLD 6.0 to produce marginal effects, their standard errors, their z values, and p values for all regression parameters, except for the intercept terms. This output appears in a separate table called Marginal Effects.

For attributes in Choice models, Latent GOLD 5.0 and 5.1 already produced a table named Marginal Effects. However, these are not *average* marginal effects but instead are obtained assuming that all alternatives are equally likely. Moreover, those effects do not use the specific treatment of categorical attributes discussed above (under point 2).

11.5 ASSOCIATION STATISTICS WITH A FAST PARAMETRIC BOOTSTRAP FOR (LOCAL) MODEL FIT ASSESSMENT

Inspired by the Bayesian posterior predictive check (PPC), Latent GOLD 6.0 implements new types of chi-squared statistics for which computation of bootstrap p-values can be very fast (Van Kollenburg, Mulder, and Vermunt, 2018; Nagelkerke, 2018, Chapter 5). This new approach is most useful when the standard parametric bootstrap of chi-squared goodness-of-fit and bivariate residual (BVR) statistics is computationally too intensive.

As in the bootstrap of chi-squared goodness-of-fit and BVR statistics, replicate data sets are generated from the population defined by the ML model parameter estimates and observed values of the relevant statistics are compared with their values in the replicate data sets. However, instead of using statistics which require re-estimating the model of interest for each replicate sample, we use statistics that can be directly computed from the data. These should be statistics that quantify important features of the data desired to be captured by the estimated model. More specifically, we use overall and bivariate chi-squared statistics under independence, which measure the strength of the overall and bivariate associations between the observed variables in the estimated model (after accounting for the sample size). In other words, we are proposing a comparison of the observed associations in the data with their simulated values under the model to be tested. In a good fitting model, the observed and simulated values of these chi-squared association statistics will be similar.

For concreteness, assume the observed value of the Pearson chi-squared statistic quantifying the bivariate association (BVA) between dichotomous variables y1 and y2 equals 310. Note that the Pearson correlation coefficient between y1 and y2 equals $\sqrt{301/N}$, with *N* being the sample size. Suppose we estimate a 3-class model with this data set and wish to check whether this model reproduces adequately the observed y1-y2 association. We would like the association between y1 and y2, as predicted by the model, to be close to the observed value of 310. Or, conceptualized somewhat differently, if we simulate replicate data sets from the estimated 3-class model and compute the y1-y2 BVA for each replicate, we obtain the sampling distribution of this statistics under the 3-class model. We conclude that the 3-class model approximates the y1-y2 association in the data well if the observed value of 310 is located within say the 95% inner interval of the sampling distribution constructed with the parametric bootstrap procedure under the model; that is, if the simulated values are neither systematically smaller nor larger than 310.

It should be noted that computing overall and bivariate chi-squared statistics under independence is equivalent to computing overall goodness-of-fit and bivariate residual (BVR) statistics under the 1-class model, because assuming conditional independence within the single class of a 1-class model is equivalent to assuming unconditional independence. This implies the formulae for the new chi-squared association statistics are the same as those of the overall goodness-of-fit (L-squared, X-squared, and Cressie-Read) and bivariate residual (BVR) statistics, but using the expected frequencies under the 1-class model. This is what Latent GOLD uses behind the scenes: for the original data and each replicate data set, it estimates a 1-class model and computes those statistics.

This new **output** option is activated with the keyword **associationstatistics**:

```
output <other output options> associationstatistics;
```

The values of the association statistics with their bootstrap p-values are reported as “Chi-squared Association Statistics” in the Model output and in a new output section “Bivariate Associations”. The latter also contains the BVA statistics for models for two-level and longitudinal data sets, which are the kind of models for which a standard bootstrap may be computationally intensive.

11.6 BEST-WORST SCORES PER CLASS AND PER INDIVIDUAL

In the context of MaxDiff or best-worst choice modeling, as an alternative to the class-specific partworth parameter estimates reported by Latent GOLD, one can also obtain class-specific best-worst scores (BWS) for each attribute category. BWS can also be computed at the level of the individual, yielding simple measures of preferences for each respondent, and are also available as standard aggregated BWS, aggregated over all respondents. In addition, separate best and worst scores are also available.

Basic components for the computation of BWS are: the number of times each attribute category appears in the choice sets, N(a), the number of times it is selected as best, B(a), and the number of times it is selected as worst, W(a). The best score equals B(a)/N(a), the worst score equals W(a)/N(a), and the BWS equals [B(a)-W(a)]/N(a). The individual BWS are obtained by repeating these computations for each individual. The class-specific coefficients are obtained by aggregating the N(a), B(a), and W(a) across individuals using the posterior class membership probabilities as weights. Note that the BWS for the 1-class model yield the standard aggregated BWS.

Class-specific BWS are reported in the output (nested in Parameters) when running a best-worst model with the Choice module from the Menu, or with the Syntax module. The individual BWS can be written to an output file using the option **Individual Coefficients** on the **ClassPred** tab in Choice or using the outfile option "**individualcoefficients**" in Syntax.

11.7 USER-DEFINED WALD TESTS

Beginning in version 5.0, Latent GOLD allows specifying user-defined Wald tests, which are tests for multiple linear constraints on the model parameters. New in Latent GOLD 6.0, as an alternative to parameter *numbers*, parameter *names* can be used in the specification of the linear contrasts one wishes to test. As before, the constraints to be tested are specified in a separate text file. The output command inducing the user-defined Wald tests is **WaldTest=<filename>**.

A linear constraint in the text file has the following form:

{(c1) p1 + (c2) p2 + … + (ck) pk = (d)}

where c1, c2, ck, and d are constants and p1, p2, and pk are parameter names (or internal parameter numbers). Note that each constraint is enclosed within braces "{…}".

Examples are "{(1) a[1,1] = (2)}" and "{(1) a[1,1] + (-1) b[1,1] = (0)}", indicating that the parameter a[1,1] equals 2 and that parameters a[1,1] and b[1,1] are equal. These two restrictions can also be formulated in a more compact way using the fact that the constants (c1), (c2), etc. can be omitted (default is (1)), that it is allowed to use a minus instead of a plus sign, and that the "= (d)" can be omitted (default is "= (0)"). This yields "{a[1,1] = (2)}" and "{a[1,1] – b[1,1]}", respectively.

A user-defined Wald test may consist of multiple simultaneous restrictions and *multiple* Wald tests can be specified in a single run. This is achieved by concatenating the multiple restrictions belonging to the same Wald test and separating the different Wald test by semi-colons ";". This is an example with 3 Wald tests, testing 3, 2, and 4 restrictions, respectively:

```
{r1} {r2} {r3};          // Wald test for restrictions 1 to 3
{r4} {r5};               // Wald test for restrictions 4 and 5
{r6} {r7} {r8} {r9};     // Wald test for restrictions 6 to 9
```

Here, r1, r2, etc. represent 9 linear restrictions of the form described above. As illustrated above, it is also possible to include comments in the text file specifying the user-defined Wald statistics.

11.8 USER-DEFINED EXPRESSIONS

Often, we are interest in functions of model parameters. Sometimes, getting their point estimates is enough, but in other situations we may also wish to quantify their uncertainty (their SEs) or use them in univariate z or multivariate Wald tests. Examples of applications in which this is useful include the computation of indirect effects, the computation of parameters with another coding, and the computation of predicted values for certain functions of the latent and/or independent variables.

Latent GOLD 6.0 Syntax implements a new option for specifying user-defined expressions or functions. These expressions may contain parameter labels, names of other expressions, numbers, operators (+, -, /, *, and ^), and mathematical functions like sqrt(), log(), and exp(). These expressions can either be defined at the end of the **equations** section, together with the parameter constraints, or in a separate file defined in the output section using **expressions=<filename>**.

The general form of an expression appearing in the equations section is:

%<name> = <expression>;

The name of an expression should be unique in the sense that it does not duplicate any variable name or parameter label in the model specification, and when used in the equations section it should start with the symbol "%". More specifically, if the program encounters a line starting with a % in the equations section, it recognizes it is an expression. Within an expression, it is possible to use the name of other expressions (without the symbol "%").

Here is sample Syntax of a model with an indirect effect of a 4-class latent variable Cluster on a continuous outcome Y via a continuous mediating variable X (for which we have two copies in the data file, namely Xdep and Xindep):

```
equations
   Cluster <- 1;
   Xdep <- 1 + (a) Cluster;
   Y <- 1 + (b) Xindep;
   Xdep;
   Y;
   %c1 = a[1,1]*b[1,1];          // indirect effect for Cluster 1
   %c2 = a[1,2]* b[1,1];         // indirect effect for Cluster 2
   %c3 = a[1,2]* b[1,1];         // indirect effect for Cluster 3
   %c4 = - c1 - c2 - c3;         // indirect effect for Cluster 4
```

As can be seen, we compute the products of the 3 parameters for the effect of Cluster on X and the effect of X on Y. Because of the effect coding scheme, the indirect effect for Cluster 4 equals the negative of the sum of the indirect effects for Cluster 1, 2, and 3. Note that the above model could be a step-3 model using latent classes obtained from a step-1 analysis.

Nested within the Parameters output, the program will report the values of the user expressions, as well as their SEs and corresponding z tests. Note that by default the tests are based on asymptotic standard errors, but it is also possible to use any of the other standard error estimators implemented in Latent GOLD.

The expressions can also be used in the definition of user-defined Wald tests. For the same example, the file with the user-defined Wald tests could have the following form:

```
// pairwise comparisons of indirect effects across Clusters
{c1-c2};
{c1-c3};
{c1-c4};
{c2-c3};
{c2-c4};
{c3-c4};
// overall test of indirect of Cluster on y
{c1}{c2}{c3};
```

This yields Wald tests for the contrasts of the indirect effects between pairs of classes, and for the overall indirect effect consisting of three (non-redundant) terms.

11.9 SCORE TESTS FOR MULTIPLE SETS OF PARAMETERS SIMULTANEOUSLY

Beginning in version 5.0, Latent GOLD allows performing score tests for parameter sets which are restricted. A score test indicates whether the model fit would significantly improve by relaxing the parameter restrictions concerned. Score tests are part of the family of inferential tools to be used in conjunction with ML estimation, and to which also likelihood-ratio and Wald tests belong. In Latent GOLD, score tests are requested using the output command "ScoreTest".

In Latent GOLD 6.0, this option has been expanded to allow testing the significance of specific sets of (restricted) parameters simultaneously. This is achieved using the output command **ScoreTest=<filename>**, where the file lists the parameter names of each set one would like to test, and where sets are separated by a semi-colon ";". Here is an example:

```
ab ac ad ae bc bd be cd ce de;  // all direct effects
ab ac ad ae;                    // all direct effects involving variable a
ab bc bd be;                    // all direct effects involving variable b
ac bc cd ce;                    // etc.
ad bd cd de;
ae be ce de;
```

In this example, the (restricted) parameters represent direct effects between indicators (local dependencies) in a LC model for five indicators. Here, score tests are used to test whether all direct effects are 0, whether those where variable "a" is involved are 0, whether those where variable "b" is involved are 0, etc.

As in the user-defined Wald tests, instead of parameter *labels*, it is also allowed to use parameter *numbers*, where the numbers correspond to internal parameter numbers obtained when estimating the specified model without restrictions.

11.10 STANDARD ERROR COMPUTATION BY SIMULATING PARAMETERS

Latent GOLD implements various types of standard error estimators for the model parameters. The estimated covariance matrix of the parameters can be used for the computation of SEs of functions of the parameters, such as the probabilities and means reported in the Profile and EstimatedValues output, and the user-defined expressions described above. The standard approach for this purpose is to make use of the delta method. However, the delta method involves an approximation, which may not always work well. This is one of the reasons that the bootstrap is preferred sometimes, where SEs of parameters and functions of parameters are obtained as the SDs of these across replicate samples.

However, since it requires re-estimating the model of interest for each bootstrap replicate, bootstrapping can be time consuming. This is why Latent GOLD implements an alternative resampling method that involves sampling parameter sets directly from the MVN distribution which is defined by their point estimates and their estimated asymptotic covariance matrix. The relevant functions of parameters are computed with each sample and their SDs across samples serving as SE estimates. Those functions can be those reported in the standard Latent GOLD output sections Profile and EstimatedValues, but can alternatively be user-defined expressions as described above.

This option is activated by using the keyword **SimulateParameters** in the output options of Syntax. Most logical is to combine this option with standard Hessian based SEs, robust SEs, or complex sampling SEs using the linearization estimator.

The samples of parameters and the resulting values of the functions of parameters (Profile, EstimatedValues, and Expressions) can be stored to a compact csv output file using the "write=<filename>" output option.

11.11 WRITING PARAMETERS TO A FILE

The implementation of the Output option WriteParameters has changed slightly in the transition from Latent GOLD version 5.1 to 6.0. The specified output file now contains the parameters as reported in the Parameters Output, and thus no longer what we previously referred to as the *internal parameters*. The new Output option WriteInternalParameters can be used if one instead wishes the *internal parameters*. Note that these two output options yield different results *only* in models with monotonicity restrictions on the model parameters (using the + and - options) and in cumulative ordinal regression models (where thresholds are restricted to be monotone).

11.12 OUTFILE AND CLASSPRED OPTIONS

The options to write information from an estimated model to a newly created data file – using the *outfile* command in Syntax or the ClassPred tab in GUI models – have been expanded and improved upon. The primary improvement is that the variable names and labels in the newly created data file are somewhat clearer and more consistent. Another improvement in Syntax models is that multiple classification types can be requested simultaneously, which was already possible in the GUI modules. In GUI models with continuous latent variables, the output file will now also contain the posterior standard deviation, which was already the case in Syntax models.

A new item that can be saved to an output file is the class-specific log-densities, the log $P(\mathbf{y}|x)$ values. These values are required when applying the Bakk-Kuha stepwise LC approach described above. In GUI models, this option is accessible from the ClassPred tab, whereas in Syntax it is requested using the keyword ***logdensity***. A sample Syntax is:

```
outfile 'logdensities.sav' logdensity keep=gender age education;
```

As already indicated above, in best-worst models (which are choice models where the dependent variable reflects both the *best* and the *worst* choice), individual **best-worst scores** are written to an output file with the **individualcoefficients** option.

11.13 PREDICTION STATISTICS: CONDITIONAL OPTION

Latent GOLD implements the new setting "conditional" for the Prediction Statistics. This option is used as the standard in Step3-Dependent and can be requested by the user in Syntax models using the output option "predictionstatistics=conditional.

Similar to the setting "posterior", this new setting uses the posterior class memberships as weights, though in a slightly different way. In posterior prediction statistics, Latent GOLD first obtains the predicted value as a weighted average of the class-specific prediction values, which is subsequently compared with the observed value. In conditional prediction, each of the class-specific predictions is compared with the observed value (say with a squared error loss function), and these comparisons are averaged over classes using the posteriors as weights. The latter approach is equivalent to what Latent GOLD does in the computation of the loadings output, which also contains an R-squared value based on squared errors.

11.14 OTHER NEW OUTPUT

Other minor additions are the reporting of "Loadings" for Markov models, the inclusion of the "loadings" (in Cluster, DFactor, and Markov) and "reorderclasses" keywords among the "output" options when using the "Generate Syntax" option, and the reporting of an "Overall" column in Profile for GUI models (this was already in for Syntax models).

Moreover, various additional items are reported in the Summary table with statistics for all estimated models; that is, it now also contains the total and the largest BVR, the bootstrap p-values for the L-squared and the -2LL Diff statistic, the Vuong-Lo-Mendell-Rubin (VLMR) statistic and its p-value, and the entropy R-squared.

12 OTHER NEW OPTIONS

12.1 FLEXIBLE SPECIFICATION OF PRIORS FOR MODEL PROBABILITIES

Latent GOLD allows the use of priors for model probabilities, rates, and variances. Rather than using the *Bayes* options for specifying the weights of the priors, it is now possible to specify the exact values of the Dirichlet parameters of the prior distributions for the model probabilities. This is achieved with the "parameters" option "**~pri**" in combination with the specification of fixed values. The priors specified in this way override the settings used for the bayes options.

This is sample Syntax illustrating the use of this new option in a two-class model for 4 dichotomous indicators:

```
equations
   Cluster <- 1;
   Y1 <- 1 + Cluster + (p~pri) 1 | Cluster;
   Y2 <- 1 + Cluster + (p~pri) 1 | Cluster;
   Y3 <- 1 + Cluster + (p~pri) 1 | Cluster;
   Y4 <- 1 + Cluster + (p~pri) 1 | Cluster;
   p = {4 1
        1 4};
```

Here, the terms "(p~pri) 1 | Cluster" do not refer to a set of model parameters, but to the priors for P(Y1|Cluster), P(Y2|Cluster), P(Y3|Cluster), and P(Y4|Cluster), respectively. The term is labelled "p" and the parameters of the Dirichlet priors for these probabilities are defined as fixed-value "constraints". Note that in this example we use the same set of priors for each dependent variable, which is why a single label "p" is used. In terms of amount of prior information, this specification is equivalent to using the bayes option "categorical=10". However, instead of deriving the prior probabilities from the observed marginal distributions and using the same values for each latent class, here we specify the (1,1) and (2,2) combinations as being 4 times more likely than the (1,2) and (2,1) combinations.

12.2 EXPANDED BAYESIAN IMPUTATION OPTIONS

In Latent GOLD it is possible to perform Bayesian imputation of categorical variables using Gibbs sampling. In version 5.1, this option was available only for simple LC models, but it has been expanded in version 6.0 to work also with multilevel LC models and latent Markov models. For more details on the use of LC models for the multiple imputation of categorical multilevel and longitudinal data, see Vidotto, Vermunt, and Van Deun (2018, 2020).

Using Latent GOLD for multiple imputation is achieved with the outfile option **imputation=<number>**. Add the keyword "**gibbs"** to use the Bayesian MCMC approach. Here is an example:

```
outfile 'data_imputed.sav' imputation=10 gibbs;
```

12.3 SIMULATION STUDIES USING RANDOM PARAMETER VALUES

Latent GOLD can be used to simulate data sets and to perform Monte-Carlo studies. This requires specifying the population model structure and its parameter values, as well as specifying an example data file defining the data structure.

New in version 6.0 is the option to sample parameter values from a MVN distribution rather than using fixed parameter values. This requires specifying both the means and the covariances of the parameters in a text file of which the name is listed at the end of the equations section. This file first lists the parameter means, then the keyword **VarCov**, and then the covariance matrix of the MVN distribution one wishes to sample from.

12.4 SPECIAL PARAMETERS SPECIFICATIONS

12.4.1 DIAGONAL AND OFF-DIAGONAL PARAMETERS

Latent GOLD implements various types of special effects, which are denoted by a "~" in the equations. Two new options for modeling associations in squared tables are **~off** and **~dia**, which define off-diagonal and diagonal parameters in associations consisting of variables with equal numbers of categories. The off-diagonal option is similar to the "~tra" and "~err" specifications, but these work only with latent-latent and latent-dependent associations, respectively. The "~off" option is more general in that it also works with independent-dependent and dependent-dependent associations.

12.4.2 EXPANDED ~WEI OPTION

The functionality of the "parameter" option **~wei** has been expanded. It works with all scale types now, whereas before it could be used only with categorical dependent variables.

12.4.3 EXPANDED RESIDUAL COVARIANCE STRUCTURES FOR CONTINUOUS DEPENDENT VARIABLES

Two new options are implemented for modeling the residual covariance structure for continuous dependent variables. The **~ar1** option yields an autoregressive covariance structure, which may be useful in longitudinal data applications. The **~cor** option gives covariances in terms of correlations times the standard deviations of the two variables involved, which is useful if one wishes to impose constraints – say between latent classes -- on correlations rather than on covariances.

A sample Syntax of a LC growth model with an ar1 covariance structure is:

```
equations
   read1 – read4 <- 1 | Class;
   (s) read1 -read4;
   (a~ar1) read1 - read3 <-> read2 - read4;
```

Note that the parameter labels “s” and “a” are used to restrict the residual variances and the ar1 terms to be equal across (pairs of) dependent variables. For the ar1 terms, this is a logical restriction, but for the variances one may relax this constraint, yielding a heterogeneous ar1 structure.

A sample Syntax of a LC growth model with a compound symmetry structure for the correlations but with unrestricted variances is:

```
equations
   read1 – read4 <- 1 | Class;
   read1 -read4;
   (a~cor) read1 - read3 <-> read2 - read4;
```

12.5 COMPACT SPECIFICATION OF THE LIST DEPENDENT AND INDEPENDENT VARIABLES

In models with more than a few dependent and/or independent variables, the specification of the list of (in)dependent may contain several lines of Syntax code. In most cases, this is not difficult to produce because an initial Syntax can be generated after specifying a model with one of the Latent GOLD point-and-click GUI modules.

Latent GOLD 6.0 allows specifying portions of the Syntax in a more compact manner. That is, a list of variables can be provided, where the specified features such as the coding and the scale type applies to the entire list. This is an example with 10 continuous dependent variables called y1, y2,…, y10 and appearing in this order in the data file:

```
dependent (y1 - y10) continuous;
```

An example with 10 continuous and 10 nominal dependent variables and with a differential weighting of the two sets of dependent variables is:

```
dependent (y1 - y10) continuous varweight=.5, (y11 - y15 y21 - y25) nominal;
```

12.6 RUNNING MODELS FOR A RANGE OF CLASSES

In Cluster, Regression, and Choice models, it was already possible to estimated models for a range of numbers of clusters or classes. This is achieved by using “from-to”, say “1-5”, in the specification of the number of Clusters or Classes. This option has been expanded in Latent GOLD 6.0, so that it also works for the number of GClasses in multilevel LC models, the number of DFactors and DFactor models, the number of sClasses in Choice models, and the number of States and Classes in Markov models.

In addition, the Latent GOLD Syntax is extended to allow running a range of models with different numbers of categories of the discrete latent variables. In Syntax, one should specify the range using “from:to”, that is, using a colon “:“ between the lower and the upper value. An example is:

```
latent Cluster nominal 1:5;
```

12.7 FORMATTING OF NUMBERS WHEN RUNNING IN BATCH MODE

The last new option is relevant when running models in batch mode. We implemented an option which allows changing the formatting of numbers in the lst or html output file. This is achieved with the command line switch **/fmt [f|g|e]<number>**. Here f is fixed format, g is general format, and e is exponential/scientific format. These are immediately follow by <number>, which is an integer specifying the number of decimals (with f) or significant digits (with g or e). An example is “lg60 model.lgs /fmt f5”.

REFERENCES

Bakk, Z. and Kuha, J. (2018). Two-step estimation of models between latent classes and external variables. *Psychometrika*, 83, 871-892.

Bakk, Z., Tekle, F.B., and Vermunt, J.K. (2013). Estimating the association between latent class membership and external variables using bias adjusted three-step approaches. *Sociological Methodology*, 43, 272-311.

Bartus, T. (2005). Estimation of marginal effects using margeff. *Stata Journal*, 5, 309-329.

Brančík, L. (2006). Techniques of matrix exponential function derivative for electrical engineering simulations, Proceedings of 2006 IEEE International Conference on Industrial Technology, 2608- 2613.

Brančík, L. (2008). MATLAB programs for matrix exponential function derivative evaluation. Institute of Radio Electronics, Faculty of Electrical Engineering and Communication Brno University of Technology.

Clouth, F.J., Pauws, S., and Vermunt. J.K. (2020). On the use of inverse propensity weighting in latent class analysis. Paper presented at MBC^2, Workshop on Models and Learning for Clustering and Classification, Catania, September 2020.

de Boor, C. (1978). *A Practical Guide to Splines*. Springer-Verlag.

De Roover, K. (2021). Finding clusters of groups with measurement invariance: Unraveling intercept non-invariance with mixture multigroup factor analysis. *Structural Equation Modeling*, in press.

De Roover, K., and Vermunt, J.K. (2019). On the exploratory road to unravelling factor loading non-invariance: A new multigroup rotation approach, *Structural Equation Modeling*, 26, 905-923.

De Roover, K., Vermunt, J.K., and Ceulemans, E. (2021). Mixture multigroup factor analysis for unraveling factor loading non-invariance across many groups, *Psychological Methods*, in press.

Di Mari, R., and Bakk, Z. (2018). Mostly harmless direct effects: A comparison of different latent Markov modeling approaches. *Structural Equation Modeling*, 25(3), 467–483.

Francis, B., Elliott, A., and Weldon, M. (2016). Smoothing group-based trajectory models through b-splines. *Journal of Developmental and Life-Course Criminology*, 2 (1), 113–133.

Jackson, C.H. 2011. Multi-state models for panel data: The msm package for R. *Journal of Statistical Software*, 38, 8.

Janssen, J.H.M., van Laar, S., de Rooij, M.J., Kuha, J., and Bakk. Z. (2019). The detection and modeling of direct effects in latent class analysis. *Structural Equation Modeling*, 26, 280-290.

Jennrich, R. I. (1973). Standard errors for obliquely rotated factor loadings. *Psychometrika*, 38, 593-604.

Jennrich, R. I. (2002). A simple general method for oblique rotation. *Psychometrika*, 67, 7-19.

Lanza, S.T., Coffman, D.L., and Xu, S. (2013). Causal inference in latent class analysis. *Structural Equation Modeling*, 20, 361-383.

Lattery, K. (2010). Anchoring maximum difference scaling against a threshold – Dual response and direct binary responses. Sawtooth Software Research Paper Series.

Lo, Y., Mendell, N.R., and Rubin, D. B. (2001). Testing the number of components in a normal mixture. *Biometrika*, 88, 767–778.

Long, J. S. (1997). *Regression models for categorical and limited dependent variables*, vol. 7 of Advanced Quantitative Techniques in the Social Sciences. Thousand Oaks, CA: Sage.

Long, J.S., Freese, J. (2014). *Regression models for categorical dependent variables using Stata*. 3rd ed. College Station, TX: Stata Press.

Magidson, J., and Madura, J.P. (2018). Development of an adaptive typing tool from MaxDiff response data. Presentation at 2018 Sawtooth Software Conference.

McLachlan, G., and Krishnan, T. (2007). *The EM algorithm and extensions*. John Wiley & Sons.

McLachlan, G. and Peel, D. 2000. *Finite mixture models*. New York: Wiley.

Merkle, E.C., You, D., and Preacher, K J. (2016). Testing non-nested structural equation models. *Psychological Methods*, 21, 151–163.

Moler, C., and Van Loan, C. (2013). Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45, 3–49.

Nagelkerke, E. (2018). Local fit in multilevel latent class and hidden Markov models (PhD Dissertation). Vianen: Proefschriftmaken.

Nylund, K.L., Asparouhov, T., and 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*, 14, 535–569.

Orme, B.K. (2009). *Anchored scaling in MaxDiff using dual response*. Sawtooth Software Research Paper Series.

Schneider, L., Chalmers, R.P., Debelak R., and Merkle, E.C. (2020). Model selection of nested and non-nested item response models using Vuong tests, *Multivariate Behavioral Research*, 55, 664-684,

van den Bergh, M., Schmittmann, V.D., and Vermunt, J.K. (2017). Building latent class trees, with an application to a study of social capital, *Methodology*, 13(Supplement), 13–22.

van den Bergh, M., van Kollenburg, G.H., and Vermunt, J.K. (2018). Deciding on the starting number of classes of a latent class tree, *Sociological Methodology*, 48, 303–336..

van den Bergh, M., and Vermunt, J.K. (2018). Building latent class growth trees, *Structural Equation Modeling*, 25, 331-342.

van den Bergh, M., and Vermunt, J.K. (2019). Latent class trees with the three-step approach, *Structural Equation Modeling*, 26, 481-492.

van der Palm, D.W., van der Ark, L.A., and Vermunt, J.K. (2016). Divisive latent class modeling as a density estimation method for categorical data. *Journal of Classification*, 33, 52-72.

van Kollenburg, G.H., Mulder, J., and Vermunt, J.K. (2018). The lazy bootstrap. A fast resampling method for evaluating latent class model fit. Working paper.

Vermunt, J.K. (2010). Latent class modeling with covariates: Two improved three-step approaches. *Political Analysis*, 18, 450-469.

Vermunt, J.K., and Magidson, J. (2016). Computation of scoring equations for latent class models, working paper.

Vermunt, J.K., and Magidson, J. (2021). How to perform three-step latent class analysis in the presence of measurement non-invariance or differential item functioning, *Structural Equation Modeling*, in press.

Vidotto, D., Vermunt, J.K., and Van Deun, K., (2018). Bayesian multilevel latent class models for the multiple imputation of nested categorical data. *Journal of Educational and Behavioral Statistics*, 43, 511-539.

Vidotto, D., Vermunt, J.K., and Van Deun, K., (2020). Multiple Imputation of longitudinal categorical data through Bayesian mixture latent Markov models, *Journal of Applied Statistics*, 47, 1720-1738.

Vogelsmeier, L.V.D.E., Vermunt, J.K., Böing-Messing, F., and De Roover, K. (2019). Continuous-time latent Markov factor analysis for exploring measurement model changes across time, *Methodology*, 15(Supplement), 29-42.

Vuong, Q. H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57(2), 307–333.