

# **Latent Class v4.5**

**Software for Latent Class  
Estimation for CBC Data**

**(Updated September 26, 2012)**



**Sawtooth Software, Inc.  
Orem, UT**

<http://www.sawtoothsoftware.com>

In this manual, we refer to product names that are trademarked. Windows, Windows XP, Windows Vista, Excel, PowerPoint, and Word are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

## **About Technical Support**

We've designed this manual to teach you how to use our software and to serve as a reference to answer your questions. If you still have questions after consulting the manual, we offer telephone support.

When you call us, please be at your computer and have at hand any instructions or files associated with your problem, or a description of the sequence of keystrokes or events that led to your problem. This way, we can attempt to duplicate your problem and quickly arrive at a solution.

For customer support, contact our Orem, Utah office at 801/477-4700, email: [support@sawtoothsoftware.com](mailto:support@sawtoothsoftware.com), (fax: 801/434-5493).

Outside of the U.S., contact your Sawtooth Software representative for support.



# Table of Contents

---

## Getting Started

Introduction .....	1
Capacity Limitations and Hardware Recommendations .....	3
What's New in Version 4.5? .....	4

## About Latent Class

Latent Class Estimation .....	5
Estimating Interaction Effects .....	6
Linear Variables .....	7
Latent Class Output .....	9
How Sawtooth Software's Simulators Use Latent Class Utilities .....	11

## Using The Latent Class System

Creating Your Own Datasets in .CSV Format .....	12
Project Management .....	16
Home Tab and Estimating Parameters .....	18
Choice Data File Tab .....	21
Attribute Information Tab .....	22
Choice Task Filter Tab .....	24
Settings Tab .....	25
Monotonicity (Utility) Constraints .....	28

## Numeric Results

The Data Set .....	30
A Sample Computation .....	32
Choosing the Number of Segments .....	36
Practical Problems in Latent Class Analysis .....	38

## Why Latent Class?

Clusterwise Logit vs. Latent Class .....	40
HB vs. Latent Class .....	42

## References

References .....	44
------------------	----

## Appendices

Appendix A: .LCU File Format .....	45
Appendix B: .CHO and .CHS Formats .....	47
Appendix C: Directly Specifying Design Codes in the .CHO or .CHS Files .....	53
Appendix D: Analyzing Alternative-Specific and Partial-Profile Designs .....	55
Appendix E: How Constant Sum Data Are Treated in Latent Class .....	58
Appendix F: Utility Constraints for Attributes Involved in Interactions .....	61
Appendix G: Estimation for Dual-Response "None" .....	64
Appendix H: Estimation for MaxDiff Experiments .....	66



# 1 Getting Started

## 1.1 Introduction

The Latent Class System is software for estimating part worths for Choice-Based Conjoint (CBC) questionnaires. It can use either discrete choices, best-worst CBC, or constant sum (chip) allocations among alternatives in choice sets. Other advanced options include the ability to enforce utility constraints, estimate first-order interactions, and estimate linear terms.

When you purchase the Latent Class v4.5 system (currently bundled with the CBC software), latent class capabilities also become available through additional menus integrated within your SSI Web interface (in the **Analysis + Calculate CBC Utilities using Latent Class** area). However, this standalone Latent Class system offers a few additional advanced capabilities that aren't available through the SSI Web menus, including the ability to analyze MaxDiff data and user-specified design matrices for choice data.

Latent Class uses data files that can be automatically exported from Sawtooth Software's SSI Web System (CBC/Web). It can also use data collected in other ways, so long as the data conform to the conventions of the text-only and .csv format files, as described in the appendices of this manual.

### Quick Start Instructions:

1. Prepare the .CHO, .CHS, or .csv file(s) that contains choice data to be analyzed.
  - a. From CBC/Web (SSI Web System), select **File | Data Management** (v8) or **File | Export Data** (earlier versions), edit an export job, and click **Add** to choose among the .CHO, CHS, or .csv options. (Depending on the response type used in the CBC questionnaire, specific file formats are supported.)
  - b. Or, create your own [.CHO](#), [.CHS](#) or [.csv](#) data files.
2. Start Latent Class, by clicking **Start | Sawtooth Software Latent Class**.
3. From the Latent Class Project Wizard (or using **File | Open**) browse to the folder containing a .CHO file (or .CHS or .CSV file), and click **Continue**. (Wait a few moments for Latent Class to read the file and prepare to perform analysis.)
4. To perform a default Latent Class estimation (2 through 5 groups), click **Estimate Parameters Now....** When complete, a file containing the part worths for each segment (class) called studyname\_solutions.CSV (easily opened with Excel) is saved to the same folder as your original data file. The file studyname\_segment\_membership.csv contains the membership assignment into groups for each respondent. If using the Latent Class utilities in the market simulator (SMRT software), within SMRT click **Analysis | Run Manager | Import**, specify *Files of Type* should be **Latent Class Utilities (\*.pxx)**. The pxx indexing refers to how many groups are in the solution. For example, Studyname.p02 file contains the 2-group solution for Studyname, Studyname.p03 contains the 3-group solution, etc. Browse to the studyname.pxx file you wish to import and click **Open**.

The earliest methods for analyzing choice-based conjoint data did so by combining data for all individuals. Although the earliest researchers in choice analysis came to realize that aggregate analyses for choice data obscure important aspects of the data, methods for detecting and modeling underlying latent segments of respondents (that share similar preferences) only became available during the 1990s.

The Latent Class Segmentation Module is an analytical tool for use with CBC (Choice-Based Conjoint) studies. It detects segments of respondents having similar preferences based on their choices in CBC questionnaires. It uses latent class analysis for this purpose, which simultaneously estimates part worth utilities for each segment and the probability that each respondent belongs to each segment. The results (part worths, plus segment membership as filters/banner points) can be taken forward to Sawtooth Software's market simulator systems (both the separate SMRT platform and the Online Market Simulator platform).

This documentation describes only the capabilities provided by this add-on analytical tool. Readers unfamiliar with CBC should start with the CBC documentation.

Use of latent class analysis as a segmentation method has been examined in many articles in the marketing literature, and it has been found to be effective for that purpose. Thus, it holds promise of solving the problems occurring with aggregate conjoint analysis:

- There is usually sparse information at the individual level for defining segments.
- If there truly are segments with different preferences, an aggregate analysis may give incorrect answers.

Latent class became popular in about the mid-1990s as a tool for analyzing CBC data sets. The model typically provided more insight about the structure of respondent preferences than aggregate logit, and the resulting market simulations were usually more accurate than similarly defined aggregate models. The latent class approach was effective in reducing the negative effects of the IIA assumption in logit analysis. At about the same time, another even more computationally intensive technique called hierarchical Bayes (HB) became available to leading researchers and academics. Latent class provided a discrete model of respondent heterogeneity, whereas HB assumed a continuous model of heterogeneity following a multivariate normal distribution. During the late 1990s and through today, the use of HB for modeling CBC data has eclipsed that of latent class in terms of popularity. However, latent class retains a strong following and offers unique benefits (especially the ability to detect segments and assign respondents to segments). We'll further compare and contrast HB and latent class in the last stage of this documentation.



## 1.2

# Capacity Limitations and Hardware Recommendations

Because we anticipate that the Latent Class System may be used to analyze data from sources other than our CBC software programs, it can handle data sets that are larger than the limits imposed by CBC questionnaires. The professional license for the Latent Class System has these limitations:

- The maximum number of groups (classes) is 30.
- The maximum number of parameters to be estimated for any group is 1000.
- The maximum number of alternatives in any choice task is 1000.
- The maximum number of conjoint attributes is 1000.
- The maximum number of levels in any attribute is 1000.
- The maximum number of tasks for any one respondent is 1000.

The Student Lab version has the following additional limitations:

- The maximum number of respondents is 250

The Demo Version (for evaluation purposes only) is limited to 50 respondents and 3 groups (classes).

## 1.3 What's New in Version 4.5?

The latest edition of Latent Class offers a number of improvements to the interface and also to the functionality of the software:

- **Ability to use .csv and dual .csv data file formats.** Many researchers find the .csv file formats (as earlier introduced with our CBC/HB system) to be easier to manage than the .cho and .chs file formats for CBC data. Latent Class v4.5 supports these newer formats.
- **Support for Best-Worst CBC data.** SSI Web v8.1 and later supports the Best-Worst option for collecting CBC data. For each Best-Worst CBC question, the respondent indicates the best and worst product concepts within each choice task. A best-worst task is coded as two separate tasks for the purpose of utility estimation, following the same pattern as we use with our MaxDiff estimation.
- **Improved User Interface.** The new user interface is based on the popular CBC/HB v5 platform and allows you to do things in fewer clicks than before.
- **Improved Output Format in Grid (tabbed report).** Previously, the output of Latent Class was quite daunting, as it reported the results for all the replications and all the group solutions within a very long text-only document. With v4.5, we organize the output in a grid-type layout (feels more like a spreadsheet) with multiple tabs, where the different group solutions are summarized on each tab. (The complete text-based output is still available within your project file folder in a .log file if you still want to refer to it.)
- **.CHO to .CSV and .CHO to CHS Conversion Tools.** From the *Tools* menu, you can take an existing .CHO file and convert it into the much easier to read/manipulate .csv file. A tool is also available for converting the .CHO file to .CHS format.
- **Improved Compatibility with Online Market Simulator.** Version 4.5 writes out a .xml layout file that the *Online Market Simulator* uses for importing your latent class results.
- **More Efficient Procedure for Constant Sum Data.** A few years back, we implemented a mathematically equivalent but faster approach for handling constant sum data within CBC/HB. With version 4.5, we've brought that improvement over to latent class. (See Appendix E)

## 2 About Latent Class

### 2.1 Latent Class Estimation

Latent class has a role analogous to that of CBC's logit program, but rather than finding average part worth utilities for all respondents together, it detects subgroups with differing preferences and estimates part worths for each segment. The subgroups have the characteristic that the respondents within each group are relatively similar but the preferences are quite different from group to group. You may specify how many groups are to be considered, such as the range of 2 through 6. A report of the analysis is shown on the screen and saved to a log file, and the part worths for subgroups along with each respondent's probabilities of membership in the groups are stored in other files for subsequent analysis or later use by the simulator.

The latent class estimation process works like this:

1. Initially, select random estimates of each group's utility values.
2. Use each group's estimated utilities to fit each respondent's data, and estimate the relative probability of each respondent belonging to each group.
3. Using those probabilities as weights, re-estimate the logit weights for each group. Accumulate the log-likelihood over all groups.
4. Continue repeating steps 2 and 3 until the log-likelihood fails to improve by more than some small amount (the convergence limit). Each iteration consists of a repetition of steps 2 and 3.

Latent class reports the part worth utilities for each subgroup or "segment." Latent class analysis does not assume that each respondent is wholly "in" one group or another. Rather, each respondent is considered to have some non-zero probability of belonging to each group. If the solution fits the data very well, then those probabilities approach zero or one.

## 2.2 Estimating Interaction Effects

Our implementation of Latent Class estimation by default assumes a main effects (additive) model. Most of the information regarding respondent preferences is usually captured with main effects. However, sometimes interaction effects can significantly improve model fit, and CBC analysis under Latent Class permits inclusion of first-order interaction effects (two-way interactions between attributes).

For example, consider two attributes (automobile models and colors):

<u>Car</u>	<u>Color</u>
Convertible	Black
Sedan	Grey
Limousine	Red

Under main effects assumptions, we assume that we can accurately measure preferences for car types independent of the color. However, the color red goes exceptionally well with convertibles and generally not so well with limousines. Therefore, there indeed may be a relatively strong and important interaction effect between car and color. The interaction terms may indicate a *decrease* in the net utility of a red limousine after accounting for the main effects of limousine and the color red, and an *increase* in the net utility of a red convertible after accounting for the main effects of convertible and the color red.

CBC is an excellent tool for measuring all potential two-way interactions among attributes. However, many of the interactions we observe in aggregate analysis (such as with aggregate logit), are largely due to unrecognized heterogeneity. It is entirely plausible that a latent class analysis may uncover different groups of respondents: a group that prefers convertibles, a group that prefers sedans, and a group that prefers limousines. And, it may also detect that the group that prefers convertibles also tends to prefer the color red, and the group that prefers limousines also tends to reject red as a color for the automobiles they choose. If this occurs, then the latent class solution used within market simulations will indicate a "revealed" interaction during sensitivity simulations (after accumulating shares across segments), even though an interaction between model and color was never specified in the model. For example, if a black limousine is changed (through sensitivity analysis) to a red color, those respondents principally contributing share to that product would react quite strongly, reducing its share to a greater degree than if either a convertible or a sedan were modified from black to red. To the degree that interactions observed in conjoint data sets are due principally to unrecognized heterogeneity, methods that model heterogeneity, such as latent class, may obtain excellent model fit using just main effects models.

*The above example involving automobiles and colors was deliberately extreme, to convey the point. It is possible that in the real world an additional first-order interaction would be required between model and color, because the interaction may indeed occur within each individual's utility structure, rather than be an interaction observed due to respondent heterogeneity.*

In the previous example, we argued in favor of more parsimonious models within latent class, such as main effects only. However, some data sets may indicate the presence of significant interaction terms, even after accounting for heterogeneity, and latent class often provides very good estimation of interaction effects, particularly if you have relatively large sample sizes.

To specify interactions between two attributes in Latent Class software, go to the **Attribute Information** tab. Click the **+Add...** button in the bottom *Interactions* panel area, and follow the prompts.

## 2.3 Linear Variables

You can treat quantitative attributes as *linear* rather than as *discrete*. For example, suppose you have a price variable with five price levels. The standard approach is to solve for separate part worth utilities for those five discrete levels. With Latent Class, you have the option of fitting a single linear coefficient to the five price levels, which requires estimating only one parameter rather than four (five levels, minus one for effects-coding). This capability may be useful for several reasons:

*Smoothing noisy data:* Often in conjoint analysis we know that levels of an attribute have an *a priori* order, and we want the resulting utilities to display that order. However, unless the sample is very large, random error may cause some part worth values to display order reversals. For example, we usually want higher prices to have lower utilities. Sometimes we may even be confident of the shape of the curve relating utility to the underlying attribute. If we can assume utility is linearly related to the log of price, then all the information in our data is concentrated on estimating the single parameter which represents the steepness of the utility curve, and we can be assured that the resulting utilities will lie on a smooth curve.

*More power to study interactions:* The usual way to handle interactions is to estimate separate coefficients for each combination of levels. For example, with 5 brands and 5 price levels, there are a total of 25 combinations, each of which needs to be estimated in some way. Often, the size of the largest interaction dictates the sample size for the study. However, if we could treat price as a linear or log-linear variable, we would only need to estimate five values, the slope of the price curve for each brand. Even more dramatically, if we have two attributes that can be regarded as linear, such as price and, say, speed, then we could handle both main effects and their interaction with a total of only three coefficients.

*More parsimonious models:* If linear relationships hold for intermediate levels, you can conserve degrees of freedom relative to part worth models.

Of course, to use this capability properly, you must be confident of the shapes of the underlying utility curves. Latent class can be of some help in that regard as well, since you can use trial and error to see whether you achieve better fit with linear values, their logs, or other transformations. We should note that analysts often find that more parsimonious models often improve individual levels of fit (such as hit rates), but often reduce aggregate measures of fit (such as share prediction accuracy). Therefore, the modeling goals influence the decision regarding whether to fit linear terms rather than part worths.

To specify linear estimation for an attribute, go to the **Attribute Information** tab. Select *Linear* under the *Coding* column for the attribute you wish to change to linear estimation. When you choose linear estimation for an attribute, you should also provide information about the level values to be used in the design matrix for this attribute (unless you want the defaults of 1, 2, 3, etc. be used for levels 1, 2, 3, etc. of your linear attribute). Type the values to be used in the *Value* column at the right of the dialog.

For example, assume you have used the following price levels in a CBC study: \$1200, \$2000, \$3000, and you want to fit a linear term to the price attribute, rather than estimate the part worths separately for each price point. You should let Latent Class know which values to use in the design matrix for price (otherwise, defaults of 1, 2, and 3 are used as level values). You specify the values by typing the new level values in the *Values* column. For this example, it might make sense to assign the level values 1.2, 2.0 and 3.0 to represent \$1200, \$2000, and \$3000.

You are free to use any values you like, but it's a good idea to scale them to be fairly small numbers. For example, if you were studying prices of multi-million dollar products, it would be better to use values of 1, 2, and 3 instead of 1000000, 2000000, and 3000000.

*Important: The values you use in the design matrix for estimation must be the same values you use later when specifying products in the market simulator. You cannot use one set of values for price during estimation, and then expect the coefficient to work properly in simulations if you later change the assigned level values.*

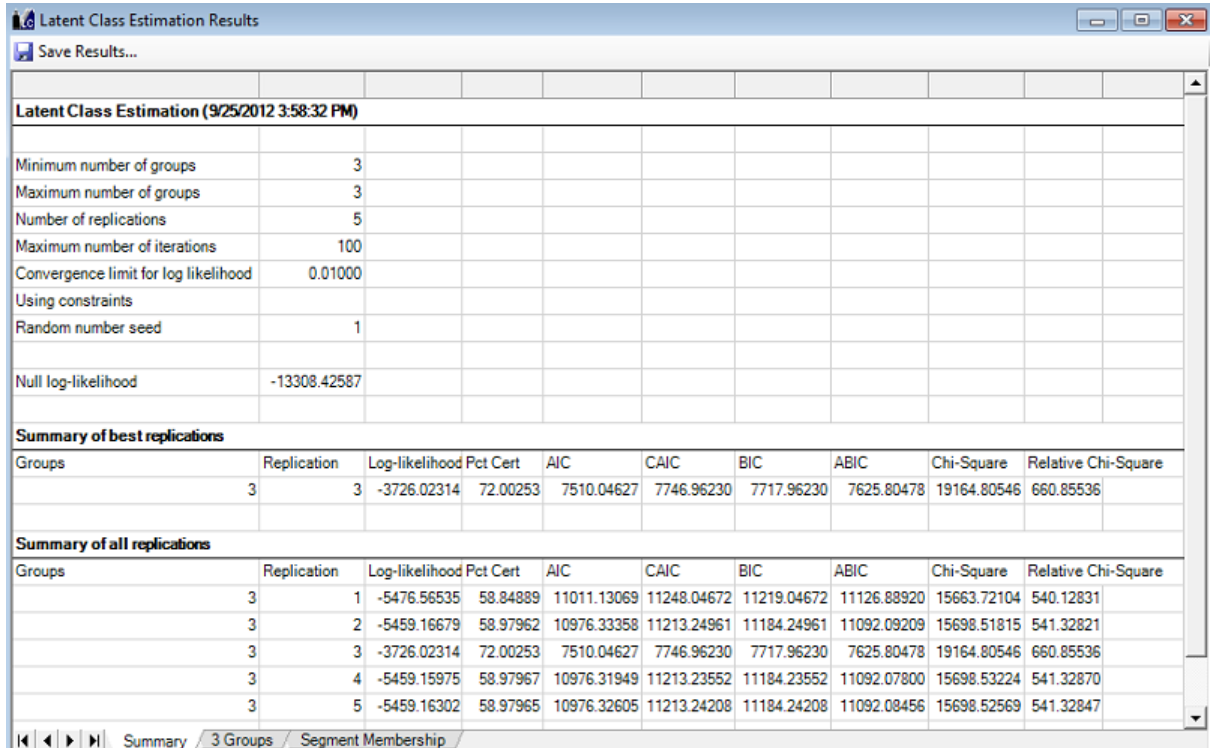
Please note that Latent Class automatically zero-centers the values you specify for a coefficient. So, if you specify three values for price as 1.2 (level 1), 2.0 (level 2), and 3.0 (level 3), the actual values used in the design matrix for the linear term for price are converted to -0.8667, -0.0667, 0.9333 (this result is obtained by subtracting the mean value from each level value). In similar fashion, Sawtooth Software's market simulators automatically zero-center any level values you assign to attributes estimated as linear coefficients. So, the coefficients resulting from zero-centered independent variables are used properly in market simulations, as the simulator always zero-centers level values prior to multiplying by coefficients.

All this is transparent to the user, but you should be aware of this treatment, should you wish to analyze the data outside of our market simulators, or use linear coefficients estimated using outside tools within Sawtooth Software's simulators.

Note that if we use the logs of level values for an attribute such as price, then price would have been treated in the analysis as the log of price. This results in a non-linear function fit to price.

## 2.4 Latent Class Output

At the end of the computation, a multi-tabbed grid-style report is given (with tabs listed along the bottom) where each tab includes information summarizing the results for different group solutions. Each respondent's segment membership is available within the *Segment Membership* tab. You can click **Save Results...** to save this output to an Excel .xlsx file.



The screenshot shows a window titled "Latent Class Estimation Results" with a "Save Results..." button. The main content is a table with the following sections:

**Latent Class Estimation (9/25/2012 3:58:32 PM)**

Minimum number of groups	3
Maximum number of groups	3
Number of replications	5
Maximum number of iterations	100
Convergence limit for log likelihood	0.01000
Using constraints	
Random number seed	1
Null log-likelihood	-13308.42587

**Summary of best replications**

Groups	Replication	Log-likelihood	Pct Cert	AIC	CAIC	BIC	ABIC	Chi-Square	Relative Chi-Square
	3	-3726.02314	72.00253	7510.04627	7746.96230	7717.96230	7625.80478	19164.80546	660.85536

**Summary of all replications**

Groups	Replication	Log-likelihood	Pct Cert	AIC	CAIC	BIC	ABIC	Chi-Square	Relative Chi-Square
	3	-5476.56535	58.84889	11011.13069	11248.04672	11219.04672	11126.88920	15663.72104	540.12831
	3	-5459.16679	58.97962	10976.33358	11213.24961	11184.24961	11092.09209	15698.51815	541.32821
	3	-3726.02314	72.00253	7510.04627	7746.96230	7717.96230	7625.80478	19164.80546	660.85536
	3	-5459.15975	58.97967	10976.31949	11213.23552	11184.23552	11092.07800	15698.53224	541.32870
	3	-5459.16302	58.97965	10976.32605	11213.24208	11184.24208	11092.08456	15698.52569	541.32847

Navigation tabs at the bottom: Summary | 3 Groups | Segment Membership

There are also several files that are automatically saved to your project folder that contain results:

**studyname\_solutions.csv** contains the final part worth estimates for each respondent group (class). It may be opened with Excel and is organized to be easily read by humans, including labels.

**studyname\_segment\_membership.csv** contains the respondent ID numbers, followed by respondent weight and group membership (the group for which the respondent has the highest likelihood of membership). This data file may be used within your cross-tabulation package, to merge respondent group membership as a new segmentation variable.

**studyname\_x\_groups\_individual\_utilities.csv** contains pseudo-individual-level utilities based on the x-group solution (where x is the number of groups requested in the latent class run). These individual-level utilities are not as accurate as HB individual-level utilities, and are simply based on the weighted average of the group utilities, where the weights are each respondent's likelihood of belonging to each group. Sawtooth Software's market simulators use these pseudo individual-level utilities if you read the latent class utilities into the simulators as utility runs. [More info.](#)

**studyname.log** contains the text-based reporting output of your run, including the random seed

used in the estimation.

**studyname.pxx** contains the respondent ID numbers followed by probabilities of membership in each group. This file is read by the SMRT program when importing the results for running market simulations. For the two-group solution, the file extension is .p02; for the three-group solution, the extension is .p03, etc.

**studyname.lcu** contains the final part worth estimates for each respondent group (class). This file is not organized to be read easily by humans, but is read by the SMRT program when importing the results for running market simulations. The layout is provided in Appendix A.

**studyname\_utility\_layout.xml** contains required information if you decide to import your latent class results into the Online Market Simulator.



## 2.5

## How Sawtooth Software's Simulators Use Latent Class Utilities

With Latent Class, rather than computing a set of part worths (utilities) for each respondent, the algorithm finds groups of respondents with similar preferences and estimates average part worths within these segments.

As when using cluster analysis, the Latent Class analyst specifies how many groups to use in the segmentation. In contrast to cluster analysis, respondents are not assigned to different segments in a discrete (all-or-nothing) manner under Latent Class analysis, but have probabilities of membership in each segment that sum to unity. The sum of the probabilities of membership across respondents for each group defines the total weight (class size) of that segment.

One can conduct overall market simulations with Latent Class results by computing shares of preference within each segment and taking the weighted average of these shares across segments.

Another way to use Latent Class data is to convert the segment-based results into individual-level estimates. While these estimates are not as accurate at characterizing respondent preferences as Hierarchical Bayes analysis, they are an appropriate extension of the Latent Class model.

The Market Simulator converts the group-based Latent Class part worths into individual-level part worths in the following way: For each respondent, a weighted combination of the group part worth vectors is computed, where the weights are each respondent's probabilities of membership in each group.

Converting the Latent Class utilities to individual-level part worths provides added flexibility for market simulations. It lets the analyst apply segmentation variables as filters, banner points or weights without requiring that a new Latent Class solution be computed each time.

However, creating individual-level utilities from a segment-based solution slightly alters the results when comparing the output of Sawtooth Software's Latent Class module to the same data used within the Market Simulator. While the overall shares of preference for the market are nearly identical, the within-class results reported in the Market Simulator output are slightly less differentiated between segments (pulled toward the overall market mean). That is because for the purpose of banner points and filters, the Market Simulator assigns respondents fully into the latent class for which they have the greatest probability of membership. For example, consider a respondent whose preferences are characterized as 80% like group 1 and 20% like group 2. His contribution to the mean values reported in the group 1 column (banner point) includes some group 2 tendencies.

The differences between the within-class means reported by Latent Class and the Market Simulator are not usually very great since respondents' probabilities of membership in classes usually tend toward zero or one. The smoothing that occurs when reporting the by-segment results in the Market Simulator will probably not substantially change your interpretation of the results. If the differences concern you, you can always refer to the original Latent Class output.

## 3 Using The Latent Class System

### 3.1 Creating Your Own Datasets in .CSV Format

Most users will probably automatically prepare data files in the studyname.cho, studyname.chs, or studyname.csv formats using Sawtooth Software's SSI Web (CBC or MaxDiff) system via the export option (Data Management area for SSI Web v8). But, other datasets created in other ways can be analyzed within the Latent Class system. You can prepare these datasets in the [.cho or .chs](#) format. Or, you can use the simpler .CSV formats described below.

---

#### Single CSV Format

*(Design and Responses within Same File)*

You can save your data to a comma-separated values (CSV) file, for example, from Excel.

You may also convert existing .CHO files to the .CSV format described below using **Tools + Convert .cho to .csv**. The layout of the file is:

Column 1: Caseid (i.e. respondent number)

Column 2: Task# (i.e. question number, or set number)

Column 3: Concept# (i.e. alternative number)

Next Columns: One column per attribute.

("None" concept is coded as a row of zeros.)

Final Column: Response/choice.

(With standard CBC questionnaires, respondents pick just one concept. The chosen concept is coded as "1," and the non-chosen concepts are coded "0." For allocation-based data (e.g. constant sum), you record how many chips are allocated within the response column. The response column can also accept decimal values. For Best-Worst CBC data, the best concept is coded as "1" and the worst concept is coded as "-1".)

Below is an example, showing the first 3 tasks for respondent #1001. This questionnaire includes 4 product concepts per task, where the 4th concept is the "None" alternative. The respondent chose concept #2 in the first task, "None" in the second task, and concept #3 in the third task. Additional tasks and respondents follow in later rows.

	A	B	C	D	E	F	G	H
1	CASEID	Task#	Concept#	Att1	Att2	Att3	Att4	Response
2	1001	1	1	2	1	3	4	0
3	1001	1	2	1	2	5	1	1
4	1001	1	3	3	3	1	2	0
5	1001	1	4	0	0	0	0	0
6	1001	2	1	3	2	2	3	0
7	1001	2	2	1	1	4	2	0
8	1001	2	3	2	3	1	1	0
9	1001	2	4	0	0	0	0	1
10	1001	3	1	1	3	3	4	0
11	1001	3	2	2	2	4	3	0
12	1001	3	3	3	1	5	1	1
13	1001	3	4	0	0	0	0	0

Respondent IDs should be unique. Task# and Concept# should always be coded in ascending order. Different respondents could potentially have different numbers of tasks, and different tasks can have different numbers concepts under this layout. Missing responses are coded as "0".

By default, Latent Class assumes each attribute column contains integer values that it will need to expand via effects-coding (part-worth function). But, if you want to "take over" all or portions of the design matrix and wish to specify columns that are to be used as-is (user-specified), even potentially including decimal values, then you may do so. You will need to identify such columns as "User-Specified" coding within Latent Class's *Attribute Information* tab.

Note: Dual-Response None studies (see Appendix G) cannot be coded using the Single CSV Format.

## Dual CSV Format

*(Design and Responses in Separate Files)*

This format can be more compact than the previously described layout when just a few versions (blocks) of the questionnaire are being used. For example, if just four versions of the questionnaire were being employed (such as for a paper-and-pencil study), the four versions could be described just once in one CSV file, and then respondent answers could be given in a second file (including which version# each respondent received). The format is as follows:

*Design File:*

Column 1: Version#

Column 2: Task# (i.e. question number, or set number)

Column 3: Concept# (i.e. alternative number)

Next Columns: One column per attribute.

("None" concept is coded as a row of zeros.)

Below is an example, showing the first 3 tasks for version #1 of the CBC questionnaire. This questionnaire includes 4 product concepts per task, where the 4th concept is the "None" alternative.

Additional tasks and versions follow in later rows.

	A	B	C	D	E	F	G
1	Version#	Task#	Concept#	Att1	Att2	Att3	Att4
2	1	1	1	2	1	3	4
3	1	1	2	1	2	5	1
4	1	1	3	3	3	1	2
5	1	1	4	0	0	0	0
6	1	2	1	3	2	2	3
7	1	2	2	1	1	4	2
8	1	2	3	2	3	1	1
9	1	2	4	0	0	0	0
10	1	3	1	1	3	3	4
11	1	3	2	2	2	4	3
12	1	3	3	3	1	5	1
13	1	3	4	0	0	0	0

(Any "fixed tasks" (holdout tasks) that are constant across versions are coded at the top of the design file as Version 0.)

Task# and Concept# should always be coded in ascending order.

*Respondent Answers File:*

Column 1: Caseid (i.e. respondent number)

Column 2: Version# (i.e. block number)

Next Columns: Responses (one per task).

The response columns are coded differently, depending on the type of CBC questionnaire. When you specify on the *Data Files* tab that your data have the CSV layout (separate design and response files), the software asks you to provide more information regarding the type of responses in your study:

Response Type:

- a) Discrete choice (single response per task)
- b) Chip allocation (response for each concept)
- c) Best/Worst (best and worst responses for each task)

None Option:

- a) A 'none' option is not included
- b) A "none" option is included
- c) A "dual response none" option is included

The responses found in the Respondent Answers File must be compatible with your specification above:

*Discrete choice*

- a) If there is a "None" option in the design file, there should be one response per task (the chosen concept 1..n); integers only. Missing="0".

b) If there is not a "None" option in the design file:

- i) If *not* using dual response none, there should be one response per task (the chosen concept 1..n); integers only. Missing="0".
- ii) If using "Dual Response None" (see Appendix G), there should be two responses per task:
  - Response #1: the chosen concept 1..n or 0 if missing (integers only)
  - Response #2: 1=would\_buy, 2=would\_not\_buy, 0=missing (integers only)

*Chip allocation*

There should be one response per concept (the number of chips); decimals allowed.  
Missing="0".

*Best/Worst*

- a) if *not* using dual response none, there should be two responses per concept (integers only, missing=0):
  - Response #1: "best" concept
  - Response #2: "worst" concept
- b) If using "Dual Response None" (see Appendix J), there should be three responses per task (integers only, missing=0):
  - Response #1: "best" concept
  - Response #2: "worst" concept
  - Response #3: 1=would\_buy, 2=would\_not\_buy, 0=missing (integers only)

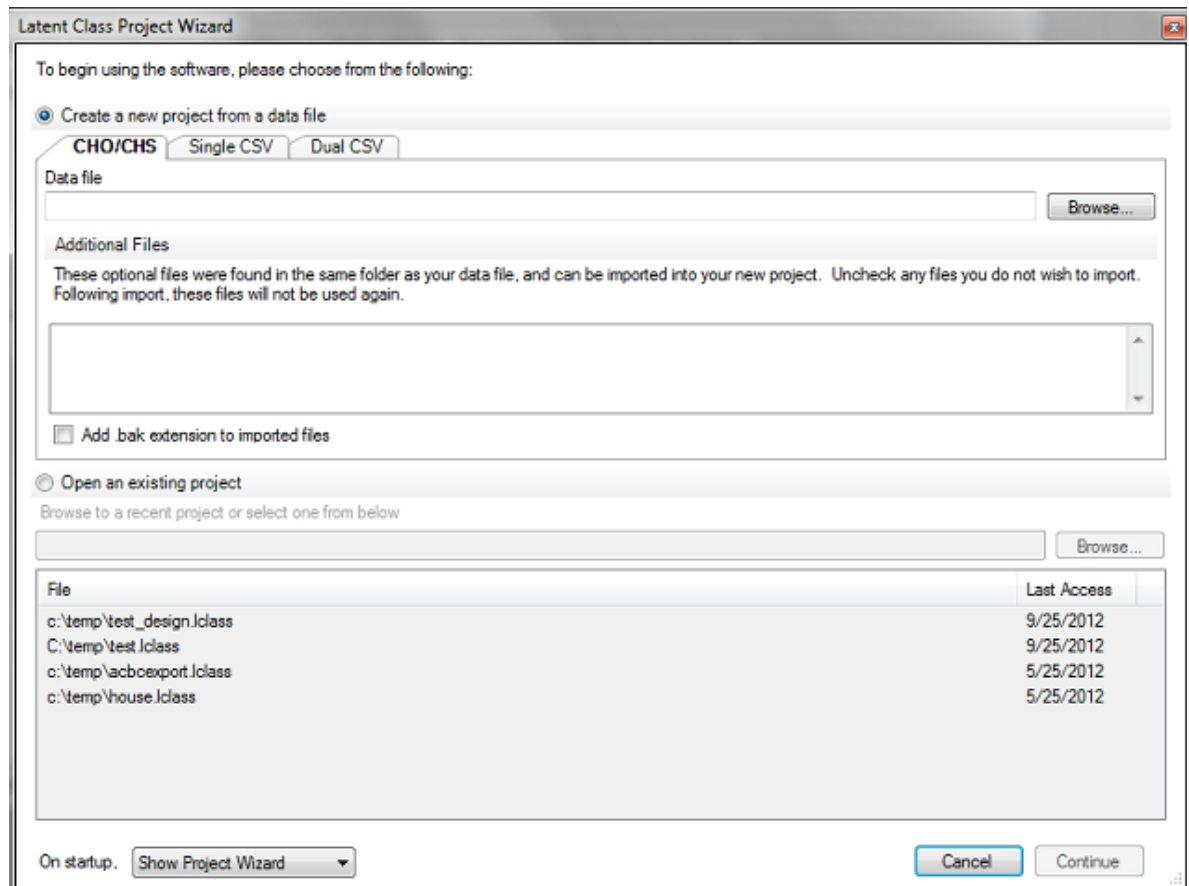
Fixed task (holdout task) responses should be first, keeping order with the design file.

## 3.2 Project Management

This section describes the operation of the Latent Class System. To start the program, click **Start / Sawtooth Software Latent Class**. You see an initial screen that identifies your license for the software.

### Creating or Opening a Project

After the initial splash screen, you see the main application and the Latent Class Project Wizard (or click **File / Open...** to access this Wizard). You can create a new project using your data file (produced by CBC or other sources), or you can open a recently opened Latent Class project file by selecting from the list of recently used projects.



The project wizard has the following options:

#### Create a new project from a data file

If you collected CBC data using Sawtooth Software's CBC system, you should use SSI Web to export a *studyname.cho* or *studyname.chs* or *studyname.csv* file (with its accompanying labels file, called *studyname.att*). A *studyname.cho* file is a text file that contains information about the product concepts shown and the answers given for choose-one (standard discrete choice) tasks. A *studyname.chs* file is a text file that contains information about product concepts shown and answers given for allocation (constant-sum) tasks. The *studyname.csv* file contains the same information as .cho and .chs file, but it can also support the the best-worst response type (not to

---

mention that it is easier to read/manipulate than .cho and .chs files).

#### **Open an existing project**

Click this option to open an existing Latent Class v4.x project with a .lclass extension.

---

### **Saving the Project**

Once you have opened a project using either of the methods above and have configured your desired settings for the Latent Class run, you can save the project by clicking **File | Save**. The settings for your Latent Class run are saved under the name **studyname.lclass**. If you want to save a copy of the project under a new name (perhaps containing different settings), click **File | Save As** and supply a new project (study) name. A new project is stored as **newstudyname.lclass**.

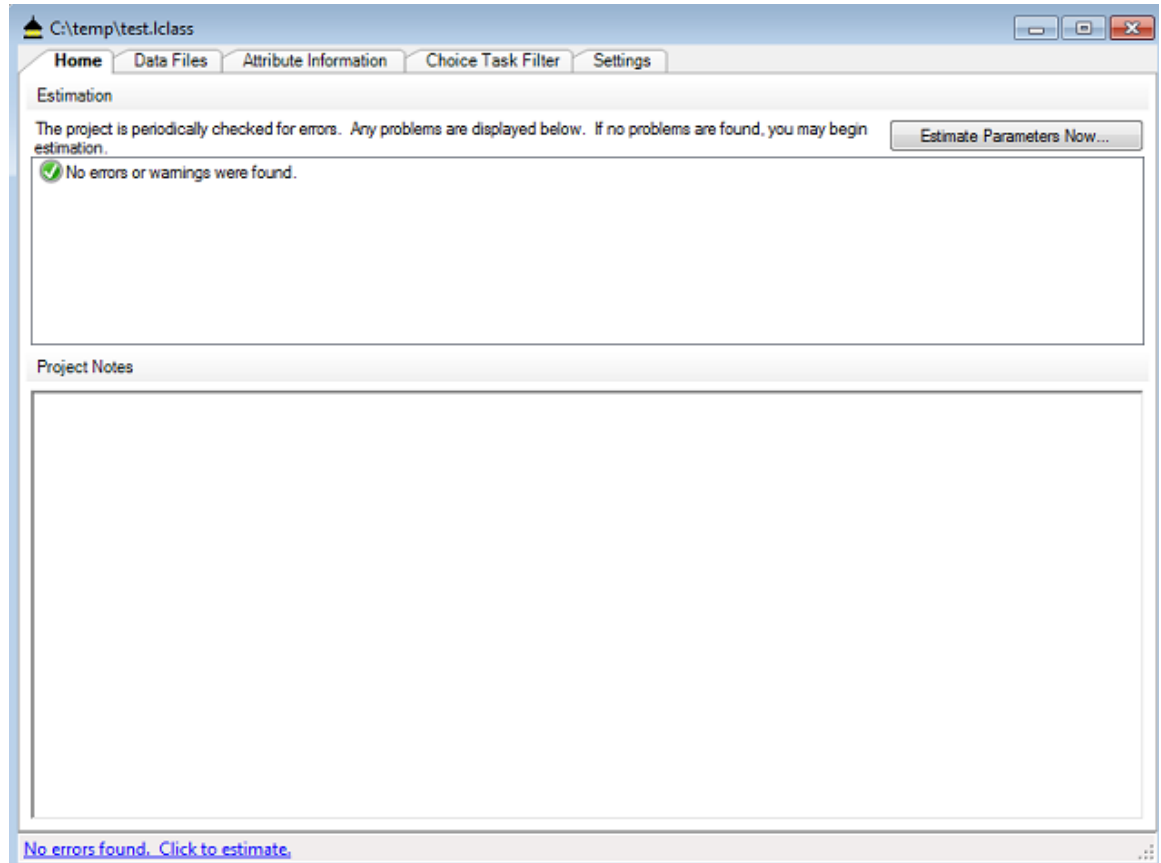
---

### **Edit | View Data File**

It is not necessary to know the layout of the studyname.cho or studyname.chs file to use Latent Class effectively. However, if you are interested, you can click the **Edit | View Data File** option. Any changes you make to this file are committed to disk (after prompting you to save changes), so take care when viewing the data file.

### 3.3 Home Tab and Estimating Parameters

After you create or open an existing project, the main project window is displayed, with five main tabs: **Home**, **Data Files**, **Attribute Information**, **Choice Task Filter**, and **Settings**.



Two panels are shown on the *Home Tab*. The first reports any error messages for the study. The second is a workspace in which you can write notes, or cut-and-paste information from your Latent Class runs for your documentation and review.

The *Home Tab* also includes the ***Estimate Parameters Now...*** button, the button you click when you are ready to perform Latent Class estimation.

---

#### Performing Latent Class Estimation

When you click ***Estimate Parameters Now...***, two things happen. First Latent Class makes temporary binary data files that can be read much faster than the original data files. Preparation of data files takes a moment, and then you see a screen like the following:



Latent Class Build Process (9/25/2012 3:04:25 PM)  
Data File: C:\temp\TV\_N250.cho

Attribute	Coding	Levels
Brand	Part Worth	3
Screen Size	Part Worth	3
Sound Quality	Part Worth	3
Channel Blockout	Part Worth	2
Picture-in-picture	Part Worth	2
Price	Part Worth	4

The number of parameters to be estimated is 11.

All tasks are included in estimation.

Build includes 250 respondents.

Total number of choices in each response category:

Concept	Number	Percent
1	889	19.76
2	887	19.71
3	965	21.44
4	864	19.2
5	895	19.89

There are 4500 expanded tasks in total, or an average of 18.0 tasks per respondent.

The build file 'C:\Users\Bryan.Bryan-THINK\AppData\Local\Temp\tmp9B51.tmp' was built successfully.

The first portion of the report identifies the source data file.

Next, the attribute list is shown, indicating the type of coding used and the number of levels for each attribute. If you want to include interactions or exclude any attributes, you may do so from the *Attribute Information* tab. If you want to treat any attributes as linear rather than as part worths, you may also make these changes from the *Attribute Information* tab.

The number of parameters to be estimated and number of respondents included is displayed. Unless you have specified interaction effects from the *Attribute Information* tab, all attributes will be included as main effects, plus a None parameter if that option was offered in the questionnaire. The number of parameters will depend on the number of attributes and levels and their coding. Effects coding is the default, and the sum of part worths within each attribute is zero. Any single level can therefore be deleted from each attribute for estimation, and recovered at the end of the computation as the negative of the sum of the included levels. We delete the final level of each attribute, and then after iterations have concluded we expand each individual's part worths to include the deleted levels.

The number of parameters shown on this screen is usually the number remaining after one level is deleted from the part worth levels for each attribute (plus an additional parameter for the "None," if present). If you include interactions, delete attributes, or use linear coding of attributes using the *Attribute Information* tab, the number of parameters to be estimated will vary accordingly.

The next information is a count of the number of times respondents selected alternatives 1 through 5 of the choice tasks. This is just incidental information about your data file that may or may not be useful.

Next are shown the total number of choice tasks and average choice tasks per respondent.

If you are satisfied with the way your data have been prepared and wish to use default settings for your Latent Class run, click ***Continue with estimation*** to begin the Latent Class iterations. If not, click ***Do not estimate now*** to return to the main menu.

## 3.4 Choice Data File Tab

You can ask Latent Class to give you a quick summary of your data file by clicking on the *Data Files* tab, and then clicking the **Browse...** dropdown control and selecting **Summary...**

After a few moments, a report like the following is displayed:

```
Analysis of 'C:\temp\TV_N250.cho'  
  Number of respondents: 250  
  Total number of tasks: 4500  
  Average tasks per respondent: 18  
  Average concepts per task: 5  
  Average attributes per concept: 6
```

Note: the full path to the data file is used so that multiple projects can point to it. If you move the location of your project or data file, it may later tell you that the data file cannot be found.

## 3.5 Attribute Information Tab

The *Attribute Information* tab displays information about the attributes in your project, how they are to be coded in the design file (part worth, linear, user-specified, or excluded), and whether you wish to model any first-order interaction effects.

In the example below, for illustration, we've changed Price to be estimated as a linear function, and have added an interaction between Brand and Price.

Attributes	Label	Coding
1	Brand	Part Worth
2	Screen Size	Part Worth
3	Sound Quality	Part Worth
4	Channel Blockout	Part Worth
5	Picture-in-picture	Part Worth
6	Price	Linear

Levels	Label	Value
1	\$300	3
2	\$350	3.5
3	\$400	4
4	\$450	4.5

Attribute	Interacts with	Expanded Parameters
Brand [Part Worth, 3 levels]	Price [Linear]	3

The attribute list was developed when Latent Class initially read the .CHO (and optional .ATT) files. If you did not have a .ATT file containing labels for attributes and levels, default labels are shown. You can edit the labels by clicking with the mouse and typing the desired labels. If you have somehow changed the .CHO or .CHS file, and the attribute list is no longer current, you can click **Other Tasks** and select **Build attribute information from data file** to scan the data file again to update the attribute information.

### Attribute Coding

There are four options for attribute coding in Latent Class:

#### Part worth

This is the standard approach used in the industry. The effect of each attribute level on choice is separately estimated, resulting in separate part worth utility value for each attribute level. Latent Class uses effects-coding to implement part worth estimation (such that the sum of part-worth utilities within each attribute is zero).

### Linear

With quantitative attributes such as price or speed, some researchers prefer to fit a single linear coefficient to model the effect of this attribute on choice. For example, suppose you have a price variable with 5 price levels. To estimate a linear coefficient for price, you provide a numeric value for each of the five levels to be used in the design matrix. This is done by highlighting the price attribute in the list, changing to *Coding to Linear*. For each price level shown in the window on the right, edit the *Value* column. Latent Class always enforces zero-centered values within attributes during estimation. If you do not provide values that sum to zero (within each attribute) within this dialog, Latent class will subtract off the mean prior to running estimation to ensure that the level values are zero-centered.

Let's assume you wish to use level values of .70, .85, 1.00, 1.15, and 1.30, which are relative values for 5 price levels, expressed as proportions of "normal price." You can specify those level values, and Latent Class converts them to (-0.3, -0.15, 0.0, 0.15, and 0.3) prior to estimation. If we had used logs of the original positive values instead, then price would have been treated in the analysis as the log of relative price (a curvi-linear function).

*Important Note: If you use Linear coding and plan to use the utilities from the Latent Class run in Sawtooth Software's SMRT program for running market simulations, you'll need to create a .VAL file prior to importing the Latent Class run into our Market Simulator tools. Simply select **Tools | Create VAL File**.*

### User-specified

This is an advanced option for supplying your own coding of attributes in the .CHO or .CHS file for use in Latent Class. For example, you may have additional variables to include in the model, such as dummy codes indicating whether an "end display" was displaying alongside a shelf-display task, which called attention to a particular brand in the choice set. There are a multitude of other reasons for advanced users to specify their own coding. Please see Appendix C for more information.

User-specified coding is also used for estimating parameters for .CHO datasets produced by our MaxDiff software.

### Excluded

Specify "excluded" to exclude the attribute altogether from estimation.

---

## Specifying Interactions

Latent Class can automatically include first-order interactions (interactions between two attributes). To add interaction terms to the model, click the **+Add...** button within the *Attribute Interactions* panel. Choose the two attributes that are to interact. For part-worth coded attributes, interactions add  $(J-1)(K-1)$  levels to the model, where J is the number of levels in the first attribute and K is the number of levels in the second attribute. However, after expanding the array of part worth utilities to include the "omitted" parameters, there are a total of JK utility values representing interaction terms written to the output file.

## 3.6 Choice Task Filter Tab

The *Choice Task Filter* tab displays a list of all available tasks in the data set. (If you have changed the data set used by your project, this list may need updating. In that case, click the ***Refresh List*** link.

With Sawtooth Software's CBC data collection systems, we often distinguish between "random" tasks and "fixed" tasks. Random tasks generally refer to those that are experimentally designed to be used in the estimation of attribute utilities. Fixed tasks are those (such as holdout tasks) that are held constant across all respondents and are excluded from analysis in Latent Class. Rather, they are used for testing the internal validity of the resulting simulation model.

You can exclude any fixed holdout tasks by unchecking the box corresponding with the task to exclude.

Some researchers also prefer to omit the first few choice tasks from estimation (treating them as warm-up tasks), so you have that option as well through this dialog.

### 3.7 Settings Tab

This tab displays the parameter values that govern the estimation. The numbers shown in each field are default values that you can change if you wish.

**Minimum and maximum number of groups:** (defaults: Minimum = 2, Maximum = 5) are the numbers of segments for which solutions should be computed. Up to a 30-group solution can be modeled. We recommend that you compare latent class results to aggregate logit results (a single group solution), if only to assess the relative gain from fitting a second group.

**Number of replications for each solution** (default: 5) lets you conduct automatic replications of each solution from different random starting points. Although the results of all replications are displayed to the screen (and saved in the *studyname.log* file), only the solution with the highest

likelihood for each number of groups is saved to the final data files (e.g. the **studyname\_solutions.csv** and **studyname\_segment\_membership.csv** files) and made available for importing into the market simulator. If your problem is relatively large, you will probably want to use only one replication in your initial investigation of the data set. However, before accepting a particular solution as optimal, we urge that you replicate that solution from several different random starting points.

**Maximum number of iterations** (default: 100) determines how long the computation is permitted to go when it has difficulty converging. To understand what happens during each iteration, it may be useful at this point to repeat how the latent class estimation process works:

1. Initially, select random estimates of each group's utility values.
2. Use each group's estimated utilities to fit each respondent's data, and estimate the relative probability of each respondent belonging to each group.
3. Using those probabilities as weights, re-estimate the logit weights for each group. Accumulate the log-likelihood over all groups.
4. Continue repeating steps 2 and 3 until the log-likelihood fails to improve by more than some small amount (the convergence limit). Each iteration consists of a repetition of steps 2 and 3.

The default iteration limit is 100, although acceptable convergence may be achieved in many fewer iterations. You may substitute any other iteration limit. If you find that you have specified more iterations than you want to wait for, you can terminate the computation at any time by clicking **Cancel**. The computation will be halted, but your results will be lost.

**Convergence limit for log-likelihood** (default: 0.01) determines how much improvement there must be in the log-likelihood from one iteration to the next for the computation to continue. We use a limit of .01 as a default, which leads to typically good precision, but you may substitute a different value.

**Total task weight:** This option is only appropriate to modify from its default value of 1 if you are using allocation-based responses rather than discrete choices. If you believe that respondents allocated ten chips independently, you should use a value of ten. If you believe that the allocation of chips within a task are entirely dependent on one another (such as if every respondent awards all chips to the same alternative) you should use a value of one. Probably the truth lies somewhere in between, and for that reason we suggest 5 as a default value when using constant sum data. **Note: you can use the .CHS file for formatting discrete choice data if you'd like. In that case, make sure to select a Total task weight of 1, and the results will be identical as when using a .CHO file (assuming the same starting seed).**

**Estimate 'None' parameter (if present)** Although you may have included a "None" alternative in your questionnaire, you may not want to include respondents' propensities to choose that alternative in the information used for segmentation.

*Note: For any respondents who choose None for all tasks, there is no information with which to classify them into clusters, and they are automatically classified into the largest cluster.*

**Tasks to include for best/worst data** If you have a dataset that includes answers of best and worst tasks within each task, then you may select which tasks to include in analysis: *Best & worst tasks, Best tasks only, or worst tasks only.*

**Constraints** accesses a dialog in which you can specify utility constraints (also known as monotonicity constraints). This can be useful for constraining part worths (or signs for linear



coefficients) to conform to rational expectations. See the next section in this documentation for more information.

**Respondent filters** accesses a dialog for selecting which respondents to include in the analysis, based on merged variables (that are included in the second line of each respondent's data in the .CHO or .CHS files).

**Respondent weighting** lets you select which variable in the demographics file (as selected from the *Data Files* tab) to use as weights. (Weights may also be supplied on the merged variables line in the .CHO or .CHS file). For example, select var10 as the weighting variable, and the 10th value on the second line of the .CHO file or .CHS file is used for the respondent weight. The values used for weights are not limited to integers, but can include decimal places of precision. When weights are used, the maximum and minimum weights are computed and reported in the output, and a message is printed saying that standard errors and t ratios may not be accurate.

**Report standard errors:** Standard errors and t ratios are only reported if this box is checked. The numerical output of Latent Class is much more voluminous than that of Logit, so we have made this information optional.

**Display re-scaled utilities and attribute importances** determines whether a table is provided within the Latent Class output in which the part worth utilities for each group are re-scaled to be more comparable from group to group (using the normalization method of "zero-centered diffs"). The logit algorithm employed in Latent Class produces utilities close to zero if members of a group are confused or inconsistent in their ratings, and produces larger values if members of a group have consistent patterns of responses. Because groups differ in the scaling of their utilities, it is often difficult to interpret differences from group to group. This table re-scales the part worth utilities for each group to make them comparable: the average range of values for each attribute (the difference between that attribute's maximum and minimum utilities) is set equal to 100. For linear attributes, we compute maximum and minimum utilities by examining the maximum and minimum values you supply within the **Attribute Information** tab. Note that even if you don't choose to include the re-scaled utilities in the output, you can later display re-scaled part worth utilities and importances by segment using the market simulator.

**Tabulate all pairs of solutions** refers to tabulations of the respondent group membership for all the solutions with one another. Although each respondent has some probability of belonging to each group, we also classify each respondent into the group to which he or she has highest probability of belonging, and tabulate each solution with those adjacent to it. That is to say, the two-group solution is tabulated against the three-group solution, the three-group solution is tabulated against the four-group solution, etc. If you check this box, then all pairs of solutions are tabulated against each other rather than just those that are adjacent.

**Random starting seed** (default: "1,") Gives you control of the random number generator that is used to provide the initial random start. The reason for giving you this control is so you can repeat a run later if you want to. Latent class analysis will probably give you a somewhat different solution each time you re-run the analysis with the same data set if you use a different starting seed. The solutions depend on the initial starting values, which are random. If you provide a value here (in the range from 1 to 10000) you will get the same solution each time, but different values will lead to different solutions. If you provide a value of zero, which is the default, then the time of day is used as a seed for the random number generator. The random seed used is printed in the .log file, in case you need to reproduce a solution.

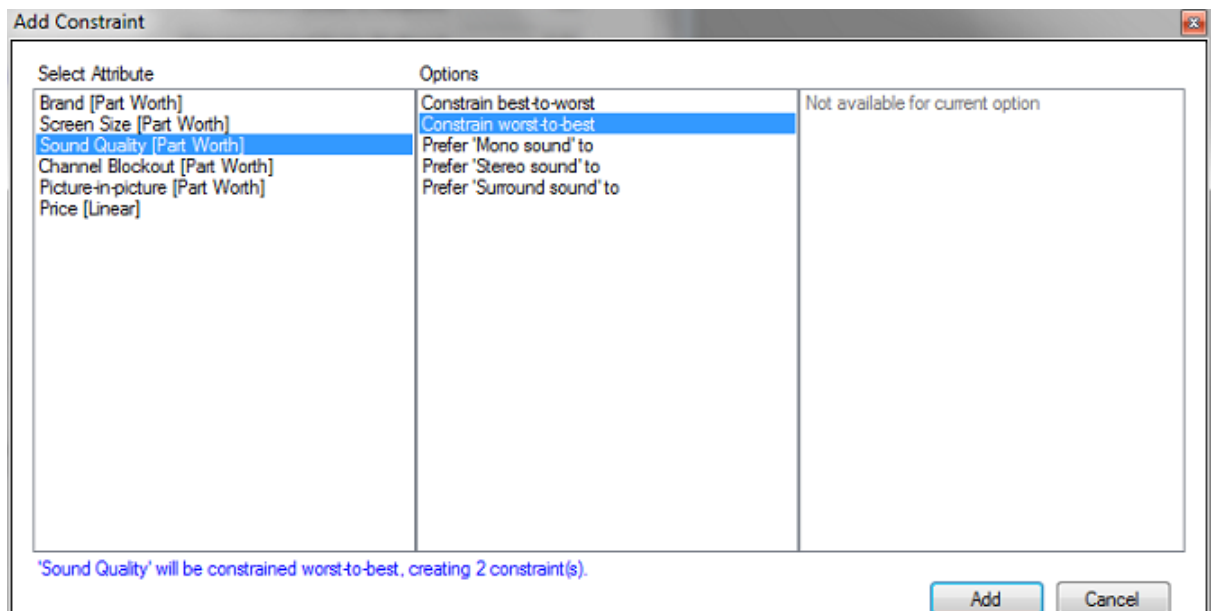
## 3.8 Monotonicity (Utility) Constraints

Sometimes there are attributes such as speed, quality, or price, where the levels have a natural order of preference. Higher speeds are preferred to lower speeds, higher quality to lower quality, and lower prices to higher prices (all else being equal). Given a good study design, execution, and adequate sample size, part worths (when summarized across groups in latent class) rarely depart from the expected rational order for such attributes. However, some studies reflect reversals in naturally ordered part worths. These reversals are usually due to inadequate sample size and can be explained as due to random error. Even so, reversals can be disconcerting for clients, and in some cases can detract from predictive accuracy. With higher-dimension solutions in latent class, reversals are quite common *within* groups, though they are typically not present when viewing the summarized results *across* groups.

Latent class provides an effective manner for "smoothing" reversals (within group) using monotonicity constraints. These constraints often improve individual-level predictions. They also tend to keep higher-dimension solutions "better behaved," helping control overfitting and leading to more meaningful and interpretable within-group preferences. But, constraints should be used with caution, as it is our experience that they can sometimes slightly reduce share prediction accuracy.

We have described the use of constraints for attributes with discrete levels, but not for attributes treated as linear, for which single parameters are estimated. Those attributes may be constrained to have positive or negative signs. For example, with price, it seems reasonable to constrain the sign of the coefficient to be negative so that higher prices correspond to larger negative utilities. With a performance attribute, it might be reasonable to constrain the sign to be positive.

You can specify monotonicity constraints by revealing the *Constraints* dialog and then clicking the **Add...** button. Then, select the attribute(s) you wish to constrain.



For example, we have specified that *Sound Quality* should be constrained from worst-to-best, meaning that the first level is worst and the last level is best (and other levels ascend in monotonically increasing order). Click **Add** to confirm the constraint and add it to the list of constraints.

---

If you employ constraints, a message to that effect appears in the output, and you are warned that standard errors and t values may not be accurate.

## 4 Numeric Results

### 4.1 The Data Set

In this section we present and describe the latent class analysis of a sample data set named "Isample.cho," that is provided in the installation. (You may open this project by browsing to the .cho file within \My Documents\Sawtooth Software\Latent Class Samples.) This example uses an artificial data set constructed as follows: First, part worth utility values were chosen for three hypothetical groups on three attributes:

#### Hypothetical Utilities for Three Segments

	Segment 1 (N = 80)	Segment 2 (N = 160)	Segment 3 (N = 240)
Brand 1	2.0	-1.0	-1.0
Brand 2	-1.0	2.0	-1.0
Brand 3	-1.0	-1.0	2.0
Pack 1	-1.0	-1.0	-1.0
Pack 2	-1.0	-1.0	2.0
Pack 3	2.0	2.0	-1.0
Price --	2.0	2.0	2.0
Price -	1.0	1.0	1.0
Avg Price	0.0	0.0	0.0
Price +	-1.0	-1.0	-1.0
Price ++	-2.0	-2.0	-2.0

One hypothetical group likes each brand best. Pack 3 is preferred by two groups and Pack 1 is preferred by no group. All agree on the desirability of paying lower prices.

From these utilities, artificial data for 480 respondents were constructed. We drew 80 simulated respondents from the first segment, 160 from the second, and 240 from the third. A CBC questionnaire was created for each individual, containing 20 tasks, each presenting three alternatives plus a "None" option. Answers for each question were simulated by first summing the respondent's utilities for each alternative, adding a random component to each sum, and then selecting the highest of the resulting values. The random components were distributed double-exponentially with variance of unity.

We have chosen an example based on simulated data rather than data from real respondents because with simulated data we know what the "right answer" is: that there are really three segments of sizes 80, 160, and 240, and the pattern of utilities for each segment.

One of the problems in interpreting logit-based utilities is that logit analysis stretches or shrinks its parameter estimates according to the goodness of its fit. If the data for a group are fit very well, its utilities tend to be far from zero. If the data for a group are fit poorly, the utilities will be closer to zero. Because some groups may be fit better than others, it can be confusing to compare results for different groups. To facilitate such comparisons, we provide the option of re-scaling each group's estimated utilities so the average range within an attribute is 100. To provide a "target" against which to later compare the estimated utilities, we scale the above hypothetical utilities in the same way:

**Hypothetical Utilities, Re-scaled for Comparability**

	Segment 1 (N = 80)	Segment 2 (N = 160)	Segment 3 (N = 240)
Brand 1	60	-30	-30
Brand 2	-30	60	-30
Brand 3	-30	-30	60
Pack 1	-30	-30	-30
Pack 2	-30	-30	60
Pack 3	60	60	-30
Price --	60	60	60
Price -	30	30	30
Avg. Price	0	0	0
Price +	-30	-30	-30
Price ++	-60	-60	-60

In conjoint analysis we sometimes summarize the relative importance of an attribute by expressing its range in utility to the sum of the ranges for all the attributes. We do that here, finding that for every segment, Brand and Pack each has importances of 30%, and Price has an importance of 40%.

**Attribute Importances**

	Segment 1	Segment 2	Segment 3
Brand	30	30	30
Pack	30	30	30
Price	40	40	40

## 4.2 A Sample Computation

The example data data set, *Isample.cho*, is provided in the ...My Documents\Sawtooth Software\Latent Class Samples folder. You can obtain your own results by running Latent Class for the tutorial data set. If you use the defaults, you will get solutions from 2 to 5 groups. Unless you use exactly the same settings as we did (including utility constraints on price and a starting seed of 1), the results below will probably differ from your own, since Latent Class does not guarantee that the same solution will be obtained every time. We start with the result for three groups, treating all attributes as discrete, and without respondent weighting. However, since we know that utilities for price should decrease with increasing prices, we constrained the price utilities accordingly.

Latent Class output is given to you in grid format, with multiple tabs (the tabs are selected along the bottom of the report). The first tab (Summary) is shown below for the *Isample* data set.

Latent Class Estimation Results

Save Results...

Latent Class Estimation (9/25/2012 3:58:32 PM)

Minimum number of groups	3
Maximum number of groups	3
Number of replications	5
Maximum number of iterations	100
Convergence limit for log likelihood	0.01000
Using constraints	
Random number seed	1
Null log-likelihood	-13308.42587

Summary of best replications

Groups	Replication	Log-likelihood	Pct Cert	AIC	CAIC	BIC	ABIC	Chi-Square	Relative Chi-Square
3	3	-3726.02314	72.00253	7510.04627	7746.96230	7717.96230	7625.80478	19164.80546	660.85536

Summary of all replications

Groups	Replication	Log-likelihood	Pct Cert	AIC	CAIC	BIC	ABIC	Chi-Square	Relative Chi-Square
3	1	-5476.56535	58.84889	11011.13069	11248.04672	11219.04672	11126.88920	15663.72104	540.12831
3	2	-5459.16679	58.97962	10976.33358	11213.24961	11184.24961	11092.09209	15698.51815	541.32821
3	3	-3726.02314	72.00253	7510.04627	7746.96230	7717.96230	7625.80478	19164.80546	660.85536
3	4	-5459.15975	58.97967	10976.31949	11213.23552	11184.23552	11092.07800	15698.53224	541.32870
3	5	-5459.16302	58.97965	10976.32605	11213.24208	11184.24208	11092.08456	15698.52569	541.32847

Summary / 3 Groups / Segment Membership

We start with estimates of respondent utilities that are just random numbers, and improve them, step by step, until the final solution fits the data acceptably well. The latent class algorithm uses a maximum likelihood criterion. Given its estimates of segment utilities and group sizes at each stage, the algorithm computes the probability that each respondent should have selected the alternative he or she did. The **log likelihood** is obtained by summing the logs of those probabilities, over all respondents and questions.

In this questionnaire there are 480 respondents, each of whom answered 20 questions, each of which had 3 product alternatives and a "none" option. Under the "null" condition of utilities of zero, any answer would have a probability of .25. The log of .25 is -1.38629. Multiplying that by the number of respondents (480) and the number of questions (20) gives the "null" log likelihood of -13308.42587.

If we were able to predict every respondent's answers perfectly, all the probabilities would be unity, and

their logs would be zero. Therefore, the most favorable possible log likelihood is zero. This computational procedure starts with a random solution that is modified iteratively, with each successive solution having a higher likelihood. For each iteration we print the resulting likelihood as well as the gain as compared to the previous iteration. The gains are large in initial iterations, and then become successively smaller in later iterations.

We requested 5 replications of the latent class run, from different starting points. The **Summary of best replications** reports that Replication #3 achieved the best fit (log-likelihood) of -3726.02314. Other measures of fit are provided.

**Pct Cert (Percent certainty)** indicates how much better the solution is than the null solution, when compared to an "ideal" solution. It is equal to the difference between the final log likelihood and the null log likelihood, divided by the negative of the null log likelihood, in this case approximately  $(-3,726 + 13,308) / 13,308$ , or 72.00. This measure was first suggested by Hauser (1978), and was used in an application similar to ours by Ogawa (1987). Although "Percent certainty" is useful for providing an idea of the extent to which a solution fits the data, it is not very useful for deciding how many segments to accept because it generally increases as more segments are included.

**Consistent Akaike Information Criterion**, (CAIC) is among the most widely used measures for deciding how many segments to accept. CAIC was proposed by Bozdogan (1987), and an application similar to ours is described in Ramaswamy et al. (1993). Like all measures we report here, CAIC is closely related to the log likelihood. Our implementation of CAIC is given by the formula:

$$\text{CAIC} = -2 \text{ Log Likelihood} + (nk + k - 1) \times (\ln N + 1)$$

where  $k$  is the number of groups,  $n$  is the number of independent parameters estimated per group, and  $N$  is the total number of choice tasks in the data set.

**Akaike Information Criterion (AIC)** is also related to the log likelihood, and is given by the formula:

$$\text{AIC} = -2 \text{ Log Likelihood} + 2 (nk + k - 1)$$

**BIC (Bayesian Information Criterion)** is very similar to CAIC, and is given by the formula:

$$\text{BIC} = -2 \text{ Log Likelihood} + (nk + k - 1) \times (\ln N)$$

**ABIC (Adjusted Bayesian Information Criterion)** is also closely related to the previous formulas, and is given by:

$$\text{ABIC} = -2 \text{ Log Likelihood} + (nk + k - 1) \times [ \ln ( (N+2) / 24 ) ]$$

Unlike the Pct Cert and Chi Square statistics, smaller values of CAIC, AIC, BIC, and ABIC are preferred. These measures are decreased by larger log likelihoods, and are increased by larger sample sizes and larger numbers of parameters being estimated. These measures are not very useful for assessing the absolute level of fit of a particular solution, but are sometimes useful when looking across alternative solutions with different numbers of groups.

*Note: in previous versions of our latent class software, we reported CAIC (but not AIC, BIC, or ABIC, which are closely related to CAIC). Because some users have requested the additional statistics, we have included them in version 4.5.*

**Chi Square** is twice the log likelihood for the solution, minus twice the log likelihood for the null solution. It can be used to test whether a solution fits significantly better than the null solution,

although that is almost always true. Chi Square is not very useful for choosing the number of segments because it tends to increase as the number of solutions increases.

**Relative Chi Square** is just Chi Square divided by the number of parameters estimated ( $nk + k - 1$ ). We know of no theoretical basis for this statistic, but Monte Carlo analyses of many data sets has led us to believe that it may be useful for choosing the number of segments. Bigger is better.

On the second tab of the report (3 Groups) the 3-group solution (for the best replication, Replication #3) is reported. At the top of the report, the history of iterations, log-likelihood fit, and size of each group is shown per each iteration.

Below that are reported the "raw" utilities, which may have been stretched or shrunk by different amounts for each group. Although the three segments are of the correct relative sizes, they are not shown in the same order as for the table of "true" utilities above.

<b>Segment Sizes</b>	50.0%	33.3%	16.7%
<b>Part Worth Utilities</b>			
Brand 1	-1.36125	-1.99435	4.97854
Brand 2	-1.39668	4.74263	-4.35689
Brand 3	2.75793	-2.74828	-0.62165
Pack A	-1.92822	-2.97450	-2.49905
Pack B	2.94556	-1.67699	-1.84212
Pack C	-1.01734	4.65148	4.34117
Price 1	2.55428	4.58037	5.82939
Price 2	2.13549	3.91768	1.36355
Price 3	-0.59954	-1.32020	1.36335
Price 4	-1.99480	-3.58882	-4.27804
Price 5	-2.09543	-3.58902	-4.27824
NONE	0.24871	0.67993	0.84798

Notice that the values for the left-hand column (for the group containing 50% of the sample) are scaled somewhat smaller than for the other two columns. Those differences make it harder to compare values in adjacent columns, a difficulty which is removed in the table below, in which each column has been re-scaled to "zero-centered diffs" so that its average range within attributes is 100.



<b>Part Worth Utilities Rescaled for Comparability</b>			
Brand 1	-29.85601	-25.69348	56.82554
Brand 2	-30.63328	61.09990	-49.72994
Brand 3	60.48929	-35.40641	-7.09560
Pack A	-42.29149	-38.32079	-28.52443
Pack B	64.60456	-21.60482	-21.02611
Pack C	-22.31307	59.92561	49.55055
Price 1	56.02256	59.00947	66.53724
Price 2	46.83746	50.47189	15.56367
Price 3	-13.14963	-17.00831	15.56139
Price 4	-43.75157	-46.23524	-48.83001
Price 5	-45.95882	-46.23782	-48.83229
NONE	5.45500	8.75958	9.67895

The true values have not been recovered perfectly, due to the random error introduced when constructing our artificial data. However, the patterns of high and low values are nearly correct, certainly close enough to characterize each group's preferences.

The re-scaled utilities immediately above are an option that you may request in the output. If you do, you will also get a table of attribute importances, like the following:

<b>Attribute Importances</b>			
Brand	30.37419	32.16877	35.51850
Pack	35.63201	32.74880	26.02499
Price	33.99379	35.08243	38.45651

The average maximum membership probability is 1.00000.

Due to weights or constraints, significance tests and standard errors may not be accurate.

We have not very accurately recovered the true importances of 30% for Brand and Pack and 40% for price for each group, but we are dealing with small sample sizes and fairly noisy data.

One indication that the solution fits the data well is that the average respondent is assigned to a group with probability 1.00. This is much higher than you will see with real data, and is only possible because we started with an artificial data set in which each individual was clearly in one group or another. Finally, if we examine which group each individual was assigned to, the recovery is perfect. We constructed a data set in which the first 80 respondents were from one group, the next 160 from another, and the last 240 from the third. All are classified properly by this solution.

We are also given a notice that since some parameters were constrained (prices) the optional significance tests and standard errors may not be accurate.

## 4.3 Choosing the Number of Segments

Next we examine what happens when comparing solutions with different numbers of clusters. We re-ran this computation five times, each time estimating solutions for from 2 to 5 clusters from different starting points (and for comparison, we ran a constrained solution using a 1-group run). For each number of clusters we retained only the solution with highest Chi Square, and those solutions are summarized below:

	PctCert	AIC	CAIC	BIC	ABIC	Chi Square	RelChiSq
1 Group	26.3	19640	19714	19705	19677	6994	777
2 Groups	57.8	11275	11430	11411	11351	15380	809
3 Groups	72.1	7480	7717	7688	7596	19195	662
4 Groups	73.3	7179	7498	7459	7335	19516	500
5 Groups	75.2	6699	7100	7051	6895	20016	408

Notice that the PctCert statistic and Chi Square both increase as we increase the number of groups, but don't increase much after the 3-group solution. This follows expectations, since in theory larger numbers of groups should produce higher values. Also note that using a different starting point, we got a slightly higher fit than for the previous run we reported for the 3-group solution.

AIC, CAIC, BIC, and ABIC are very closely related, and have a minimum for five groups, indicating that they did not work at detecting the correct number of groups. Perhaps more significant is the fact that these measures decrease dramatically until 3 groups, and then become nearly flat for larger numbers of groups. Such an inflection point is probably a better indicator of the right number of groups than its absolute magnitude. Relative Chi Square is maximized for two groups, a disappointment.

We have found that when analyzing real data from human respondents, these statistics are likely to repeat such a pattern, not providing obvious information about the best choice of number of groups. It seems that rather than choosing the solution which provides the highest absolute level of any statistic, it is more useful to look at differences. For example, for Chi Square and Pct Cert we find large increases going from one group to two, and large differences again going from two groups to three. However, beyond that the differences are minimal, suggesting that three might be the right number of groups. Likewise, AIC, CAIC, BIC, and ABIC drop sharply as we go from one to two and from two to three groups, but then stays fairl constant for solutions with more groups.

When choosing among solutions, one also has access to other information, such as their patterns of utilities and estimated group sizes. Here are the relative group sizes estimated for the five solutions above:

2 Groups	0.500	0.500			
3 Groups	0.500	0.167	0.333		
4 Groups	<b>0.035</b>	0.167	0.465	0.333	
5 Groups	0.179	0.155	0.176	0.324	0.167

The four-group solution contains a tiny group, emphasized in bold, so it would probably be disqualified on that basis.

You will probably pay close attention to the estimated utilities provided by each solution. Also, each individual can be classified into the group for which he or she has highest probability, and solutions can

be further investigated by tabulating them with one another as well as with other variables. You are automatically provided with tabulations of solutions with one another. For example, here is the way respondents were classified by adjacent two-group and three-group solutions in one of the runs.

Tabulation of 2 Group vs. 3 Group Solutions

	1	2	3	Total
1	0	0	240	240
2	80	160	0	240
Total	80	160	240	480

Every solution that you consider interpreting should be tested by rerunning from several different starting points, and seeing how similar those solutions are to one another. That can be done automatically using the "replications" parameter on the **Settings** dialog, and you can see if the part worth utilities, importances, groups' sizes, and likelihoods are similar each time. Or you can merge group membership as segmentation variables into your favorite cross-tabulation tool to cross-tab the variables to see how similarly different solutions classify respondents.

If the goal of the latent class analysis is managerial relevancy of the segmentation solution, probably the most important aspects to consider when choosing a solution for segmentation purposes are its interpretability and stability (reproducibility).

One should repeat the analysis from different starting points (or by randomly splitting the sample into two halves and running the analysis separately on each group) to observe whether similar segments and segment sizes emerge. But if the primary goal for using Latent Class is to create an accurate share simulator, then the accuracy of share predictions should take precedence over segment interpretability and stability. It is sometimes the case that a latent class solution with a larger number of groups can perform better in terms of accuracy than a lower-dimension run that may have clearer interpretability. And, share prediction accuracy may occur with solutions that have more groups than the previously discussed statistical criteria might justify. Some Latent Class users save one run for the purposes of simulations, but use another run for interpretable segment classification. The interpretable segment classification may be used as a banner point (e.g. filter), and the more accurate latent class utility run may be used for estimating shares and reporting utilities.

***Many analysts use the most interpretable latent class solution as a banner point, but will use HB utilities instead for estimating shares and reporting utilities.***

## 4.4 Practical Problems in Latent Class Analysis

We have used a sample data set for which the latent class algorithm is comparatively well behaved. However, we want to avoid giving the impression that latent class analysis is simple, or that its interpretation is straightforward. It does present several difficulties. To illustrate some of these, we reran the sample problem five more times.

**Timing:** For very large problems, Latent Class can take many minutes to run. Latent Class v4.5 is significantly faster than the first versions of Latent Class, and computer speeds continue to increase, so speed is much less an issue than just a few years ago. Generally, Latent Class is slower than logit, but faster than hierarchical Bayes (HB). However, given the time required to select among latent class solutions with different numbers of segments, HB may be faster overall when considering both the PC's computational effort and subsequent human effort.

**Optimality:** In the example described earlier, we used the default convergence limit of 0.01. For the two-group solutions, Latent Class got the same solution every time in the sense that it always classified respondents the same way.

For the three-group solutions, Latent Class got the solution reported earlier three times. However the other three times it got a different solution, similar to the 2-group solution but with a tiny splinter group. The correct solution had dramatically higher likelihood than the incorrect one, so there is no question about which is superior.

One of the most severe problems with latent class analysis is vulnerability to local maxima. ***The only way to avoid this problem is to compute several solutions from different starting points.*** Latent Class will choose different starting points for each run automatically unless you specify otherwise.

For four and five groups, different solutions were found every time--the data do not naturally support stable four- or five-group solutions. As it was, they classified respondents similarly about 80% of the time. However, nearly all of them would be rejected for containing tiny groups.

You may find that the iterative process converges rapidly. However, depending on your data or an unfortunate random starting point, you may also encounter either of the following:

**Convergence may be slow.** You may reach the limit on number of iterations before your successive gains are smaller than the convergence limit. If that happens, you receive a warning message both on screen and in your printed output. In that case, it is possible that no clear segmentation involving that number of groups exists for those data, and further exploration with that number of groups may not be worthwhile. However, we suggest that you try again, with different starting points and a higher iteration limit.

**The system may become "ill-conditioned."** This means that there is not enough information to estimate all the parameters independently of one another. This message is a symptom of too many segments, too many parameters, or too little data. (It may also indicate an unacceptable pattern of prohibitions in your questionnaire.) This message is often an indication that at least one group has become tiny. If you receive this message, try again from a different starting point. If you receive it repeatedly, but only for large numbers of segments, you may have to be satisfied with fewer segments. If you receive it with a small number of segments, it may be necessary to remove interaction terms (if you have imposed them) so as to estimate fewer parameters.

These difficulties may tempt you to abandon latent class analysis. Although we think it is important to describe the difficulties presented by Latent Class, we think it is the best way currently available to find

---

market segments with CBC-generated choice data. In the next section we describe how we came to that point of view.

## 5 Why Latent Class?

### 5.1 Clusterwise Logit vs. Latent Class

By combining elements of cluster and logit analysis, it is possible to find segments of respondents in a simple and straightforward way. Our initial investigation into this subject used an approach similar to that of Moore *et al.* (1996). We constructed a segmentation/estimation module by combining CBC's Logit module with elements of our K-Means cluster analysis product, CCA. We called this method "Klogit," which went like this:

1. Set the number of clusters, and choose an initial solution for each cluster consisting of random numbers in the interval -0.1 to 0.1.
2. Use each group's values to fit each respondent's data, using a logit model. (Of course, the initial solutions will not fit anybody at all well.) Reallocate each respondent to the group whose coefficients provide the best fit.
3. Estimate a logit solution for the respondents in each group.
4. Repeat steps 2 and 3 until no respondents change groups.

We used Monte Carlo methods to investigate Klogit's ability to recover known cluster structures. It did quite well, and we would base our CBC segmentation module on Klogit if we had not also had experience with latent class estimation.

However, because of the favorable reports on latent class methods in the references above, we further extended CBC's Logit algorithm to perform latent class analysis. Our algorithm, Latent Class, is more complicated than Klogit, but not very much so, and goes like this:

1. As with Klogit, set the number of clusters, and choose a random initial solution for each cluster with values in the interval -.1 to .1.
2. As with Klogit, use each group's logit coefficients to fit each respondent's data, and estimate the likelihood of each respondent's belonging to each class.
3. Estimate a *weighted* logit solution for each class. Each solution uses data for *all* respondents, with each respondent weighted by his or her estimated probability of belonging to that class.
4. Underlying this method is a model which expresses the likelihood of the data, given estimates of group coefficients and groups sizes. Compute the likelihood of the data, given those estimates.
5. Repeat steps 2 through 4 until the improvement in likelihood is sufficiently small.

Our implementation of the latent class algorithm is based on the description by DeSarbo, Ramaswamy, and Cohen (1995), who provide the likelihood equation that we use as well as some further details of the estimation procedure.

We do not produce the likelihood equation here because of typographic limitations. However, log-likelihood is computed as follows:

1. For each individual, compute the probability of each choice that was made, assuming the individual belonged to the first group. Multiply together those probabilities for all tasks to get a likelihood of that individual's data, assuming membership in the first group. Also make a similar computation assuming membership in each other group.
2. Weight the individual's likelihood for each group (as just described) by the current estimate of that group's size. (The group size estimates sum to unity.) Sum the products over groups, to get a total (weighted) likelihood for that individual.
3. Cumulate the logs of those likelihoods over individuals.

4. Individual probabilities of belonging to each group are obtained by percentaging the weighted likelihoods obtained by steps 1 and 2. Relative group size estimates are obtained by averaging those values over respondents.

We have compared Latent Class and Klogit in many Monte Carlo studies, with these results:

- Klogit is much faster than Latent Class, by an average factor of about 3.
- For Monte Carlo data with random response error but no heterogeneity within segment, Klogit and Latent Class do about equally well. With moderate response error they both obtain the correct solution almost every time, and they do quite well even with very large response error.
- However, when the data contain within-cluster heterogeneity, Latent Class pulls ahead. Latent Class is more reproducible when repeated solutions are computed from different starts, recovers known solutions better, and also produces groups more nearly of the right size.

One of our principles at Sawtooth Software is to try to produce tools that work every time. That leads us to select the more robust and stable Latent Class method, despite the fact that Klogit is conceptually simpler and much faster.

## 5.2 HB vs. Latent Class

As mentioned in the introduction to this documentation, hierarchical Bayes (HB) is used more often today to analyze CBC data. Still, Latent Class offers unique benefits, and it has a strong following.

Latent class may be particularly beneficial in the following circumstances:

- Often one of the goals of a CBC project is to learn more about the natural market segments that might exist in a population. A marketing manager might benefit from knowing about the different preferences across segments, especially if these segments have targetable differences beyond just preferences. Latent class provides an effective way to discover segments, and the statistics it provides for determining an appropriate number of segments are argued to be superior than those for the usual alternative of cluster analysis. Although it is possible to run cluster analysis on the results of HB analysis, we expect that analysts can achieve slightly better results if using the one-step latent class procedure.
- If respondents are really segmented into quite compact and differentiated groups in terms of preference, the discrete assumption of heterogeneity used in latent class can more accurately model individuals than HB.

HB has the following strengths:

- If respondents seem to be distributed in a more continuous rather than discrete fashion in terms of their multidimensional preferences, then HB can more accurately reflect individuals' preferences. We would argue that most data sets usually involve a relatively continuous distribution of heterogeneity, which is more in harmony with HB's assumptions. HB is also quite robust even when there are violations of the underlying assumptions. It is not surprising, therefore, that hit rates (individual-level prediction accuracy of holdout tasks) from HB almost always exceed those for Latent Class for typical CBC data sets.
- Market simulations using the individual-level part worths resulting from HB estimation are usually a bit more effective than Latent Class in reducing IIA problems.

We know of two recent studies in which individual-level estimates from HB did not improve predictive fit over aggregate logit or latent class solutions. The first study (Pinnell, Fridley 2001) demonstrated better hit rates of holdout tasks for aggregate logit than HB for some commercial partial-profile CBC data sets (HB was also superior for some partial-profile CBC data sets). In another recent study (Andrews, Ainslie, Currim 2002), better household-level prediction was achieved with latent class and logit than HB for simulated scanner choice data when the number of simulated purchases was only three per household and seven parameters were fit (with more data available at the household level, HB performed as well as latent class, with better internal fit). In both cases where HB seemed to perform worse than aggregate methods, the data were exceptionally sparse at the individual level and HB was subject to overfitting. We speculate that the narrow margin of victory for aggregate methods over HB in both cases may have been due to sub-optimal specification of the priors (prior covariance, and degrees of freedom for the prior covariance matrix). We have re-analyzed two of the data sets reported by Pinnell and Fridley, and after adjusting HB priors (assuming lower heterogeneity and a greater weight for the prior) found that HB's hit rates slightly exceeded that of aggregate logit. We haven't examined the simulated household data of Andrews, Ainslie and Currim, so we can only speculate that modified HB priors may have affected their conclusions.

We'll conclude with two more observations:

- If the principal goal of the research is to develop an accurate group-level market simulator, then



---

latent class has been shown to offer simulation accuracy that approaches (and has been reported by some authors to occasionally exceed) the accuracy of HB.

- Some researchers have learned to leverage the strengths of both techniques within their CBC analysis. They use latent class to detect segments, and use the segment membership information as "banner points" (filters) applied to simulations using underlying HB utility runs.

## 6 References

### 6.1 References

Andrews, R. L., Ainslie, A., and Currim, I. S., "An Empirical Comparison of Logit Choice Models with Discrete Versus Continuous Representations of Heterogeneity," *Journal of Marketing Research*, November 2002, p 479-487.

Bozdogan, H. (1987), "Model Selection and Akaike's Information Criterion (AIC): The General Theory and its Analytical Extensions," *Psychometrika*, 52, 345-370.

DeSarbo, W. S., V. Ramaswamy, and S. H. Cohen (1995), "Market Segmentation with Choice-Based Conjoint Analysis," *Marketing Letters*, 6, 137-148.

Hauser, J. R. (1978), "Testing and Accuracy, Usefulness, and Significance of Probabilistic Choice Models: An Information-Theoretic Approach," *Operations Research*, 26, (May-June), 406-421.

Lenk, P. J., W. S. DeSarbo, P. E. Green, and M. R. Young (1996), "Hierarchical Bayes Conjoint Analysis: Recovery of Partworth Heterogeneity from Reduced Experimental Designs," *Marketing Science*, 15 (2) 173-191.

Ogawa, K. (1987), "An Approach to Simultaneous Estimation and Segmentation in Conjoint Analysis," *Management Science*, 6, (Winter), 66-81.

Pinnell, Jon and Fridley, Lisa (2001), "The Effects of Disaggregation with Partial Profile Choice Experiments," *Sawtooth Software Conference Proceedings*, Sequim WA, 151-165.

Ramaswamy, V. , W. S. DeSarbo, D. J. Reibstein, and W. T. Robinson. (1993), "An Empirical Pooling Approach for Estimating Marketing Mix Elasticities with PIMS Data." *Marketing Science*, 12 (Winter), 103-124.

## 7 Appendices

### 7.1 Appendix A: .LCU File Format

#### Studyname.LCU File Format

The **studyname.lcu** file contains the final utilities for each respondent group (class) from Latent Class analysis. Since this file always has the same name, it is important that you rename it before doing further analyses with the same study name to avoid over-writing it.

##### *The Header:*

The file contains a header section that describes which parameters have been estimated, followed by part worth utilities for each group for each latent class solution.

Following is an example of such a header:

```

3 1 12 2 3
3 3 5
1 0 0
0 1 0
0 0 1
1 1 Brand 1
1 2 Brand 2
1 3 Brand 3
2 1 Pack A
2 2 Pack B
2 3 Pack C
3 1 Price 1
3 2 Price 2
3 3 Price 3
3 4 Price 4
3 5 Price 5
NONE

```

The first line contains the number of attributes, whether "None" is included (1 if yes, 0 if no), the total number of parameters estimated for each individual, and the minimum and maximum number of segment solutions reported below. For example, if this file contains results from 2-group to 3-group solutions, then values of 2 and 3 are shown.

The second line contains the number of levels for each attribute.

Following is a line for each attribute, each with as many entries as there are attributes. This is an attribute-by-attribute matrix of ones and zeros (or minus ones) which indicates which effects were estimated. Main effects are indicated by non-zero values on the diagonal. Interactions are indicated by non-zero values in other positions. Linear variables are indicated by negative entries on the diagonal.

Following are labels, one for each parameter estimated. These are in the same order as the parameter estimates that follow in the file, and serve to identify them. If interaction parameters were estimated, then this list will include a label for each term.

##### *Group Utility Data:*

Below the header, the part worths are listed for each solution. For example, data for the the 2-group solution are listed as follows:

2

240 0.30835 1.20565 -1.51400 -1.20315 -0.65658 1.85972 2.16634 0.81157 -0.07445 -1.45163  
-1.45183 -0.16877240 -1.36149 -1.39686 2.75835 -1.92847 2.94598 -1.01751 2.55459 2.13587 -0.59984 -1.99489  
-2.09574 0.24880

The "2" in the first line of this section indicates that the data that follow are for a 2-group solution. On the next line, the 240 indicates the weighted number of respondents included in the first group. Following that are the 11 part worth utilities plus the None utility characterizing the first group. The second group's weight and part worth parameters follow that.

Sections follow for as many solutions there are for larger numbers of groups. For example, for the 3-group solution, group weights and part worths are:

3

160 -2.01774 4.77464 -2.75690 -2.99721 -1.68169 4.67891 4.60645 3.94615 -1.32848 -3.61196  
-3.61216 0.67780240 -1.36152 -1.39685 2.75837 -1.92839 2.94592 -1.01753 2.55460 2.13594 -0.59979 -1.99497  
-2.09578 0.2488780 5.25094 -4.32132 -0.92961 -2.45805 -2.00137 4.45942 6.09424 1.33020 1.33000 -4.37712  
-4.37732 0.99309

## 7.2 Appendix B: .CHO and .CHS Formats

The Latent Class program can use a studyname.CHO or a studyname.CHS data file, which are in ASCII (text-only) format. The studyname.CHO is for discrete choice data, and is automatically created by the CBC system. The studyname.CHS file is for allocation-based, constant sum (chip) allocation CBC questionnaires. It is not necessary to have used our CBC system to create the data file. You can create any of the ASCII-format data files and supporting control files with a text editor or other data processing software.

### .CHO File Layout

The studyname.CHO file contains data from each interview, including specifications of the concepts in each task, respondent answers, the number of seconds required for each answer (optional), and total interview time in minutes (optional).

Individual data records for respondents are appended to one another; one record follows another. Following is how the single task described above would appear in a sample .CHO data file:

```

8960 2 6 12 1
4 2
3 1
2 1 2 3 2 3
3 3 3 1 3 1
4 2 1 2 1 2
2 32
.
.
.

```

We'll label the parts of each line and then discuss each part.

<b>Line 1:</b>	<b>8960</b>	<b>2</b>	<b>6</b>	<b>12</b>	<b>1</b>
	Respondent	"Extra"	Number of	Number of	None option
	Number	Variables	Attributes	Choice tasks	0=N, 1=Y

*First position (8960):* The first position on the first line of the data file contains the respondent number.

*Second position (2):* The second position contains the number of "extra" variables included in the file. These variables may include the duration of the interview, and any merged segmentation information, which can be useful for selecting subgroups of respondents for special analyses. The variables themselves appear in line two of the data file and are described below. (Most Latent Class users set the number of "extra" variables to zero and omit line 2.)

*Third position (6):* The third position contains the number of attributes in the study.

*Fourth position (12):* The fourth position contains the number of choice tasks included in the questionnaire.

*Final position (1):* The final number indicates whether the "None" option was in effect; 0 = no, 1=yes.

**Line 2:**        **4**                **2**  
                   Interview        Segmentation  
                   Duration            Variable

These "extra" variables are largely a carryover from very early versions of CBC software, and are only used in the Latent Class system for respondent filters or weights. If you specify that there are no "extra" variables on line 1, you can omit this line of data.

The remaining five lines all describe the interview's first task:

**Line 3:**        **3**                        **1**  
                   Number of                Depth of  
                   Concepts first task    Preference first task

*First position (3):* The first position gives the number of concepts in the first task, (excluding the "none" alternative).

*Second position (1):* The second position reports the depth of preference for this task. The "1" indicates the respondent was asked for his or her "first choice" only. A "2" would indicate that a first and second choice were asked for, and so on. (Latent Class only uses information for respondents' first choice.)

The next three lines describe the three concepts in the task, in the attributes' natural order. The data for each concept are unrandomized; the first attribute is always in the first position. The numbers on each line indicate which attribute level was displayed. Let's look at the first of these lines:

**Line 4:**        **2**        **1**        **2**        **3**        **2**        **3**  
                   Level Displayed for Each Attribute in First Concept

This line represents the first concept. These are always recorded in the natural order, whether attribute positions were randomized or not. So, the numbers in line 4 represent:

First position (2):	level 2 of attribute #1	(Computer B)
Second position (1):	level 1 of attribute #2	(200 MHz Pentium)
Third position (2):	level 2 of attribute #3	(5 lbs)
Fourth position (3):	level 3 of attribute #4	(16 Meg Memory)
Fifth position (2):	level 2 of attribute #5	(1.5 Gbyte hard disk)
Sixth position (3):	level 3 of attribute #6	(\$3,000)

Following are a list of the six attributes and their levels, to help in interpreting these lines from the data file:

1 Computer A  
 2 Computer B  
 3 Computer C  
 4 Computer D

1 200 MHz Pentium  
 2 166 MHz Pentium  
 3 133 MHz Pentium

1 3 lbs  
 2 5 lbs  
 3 7 lbs

1 64 Meg Memory  
 2 32 Meg Memory  
 3 16 Meg Memory

1 2 Gbyte hard disk  
 2 1.5 Gbyte hard disk  
 3 1 Gbyte hard disk

1 \$1,500  
 2 \$2,000  
 3 \$3,000

**Line 5:**       **3       3       3       1       3       1**  
                   Level Displayed for Each Attribute in Second Concept

Line five represents the second concept, and the numbers are interpreted as:

First position (3):	level 3 of attribute #1	(Computer C)
Second position (3):	level 3 of attribute #2	(133 MHz Pentium)
Third position (3):	level 3 of attribute #3	(7 lbs)
Fourth position (1):	level 1 of attribute #4	(64 Meg Memory)
Fifth position (3):	level 3 of attribute #5	(1 Gbyte hard disk)
Sixth position (1):	level 1 of attribute #6	(\$1,500)

**Line 6:**       **4       2       1       2       1       2**  
                   Level Displayed for Each Attribute in Third Concept

Line six represents the third concept, and the numbers are interpreted as:

First position (4):	level 4 of attribute #1	(Computer D)
Second position (2):	level 2 of attribute #2	(166 MHz Pentium)
Third position (1):	level 1 of attribute #3	(3 lbs)
Fourth position (2):	level 2 of attribute #4	(32 Meg Memory)
Fifth position (1):	level 1 of attribute #5	(2 Gbyte hard disk)
Sixth position (2):	level 2 of attribute #6	(\$2,000)

**Line 7:**       **2                               32**  
                   Choice                               Task Duration

*First position* (2): The first position contains the respondent's choice, which in this example is concept two.

*Second position* (32): The second position on this line contains the time it took, in seconds, for the respondent to give an answer to this task. This is optional information that doesn't affect computation, and any integer may be used if desired.

The balance of the respondent's data would consist of lines similar to the last five in our data file fragment, and those lines would have results for each of the other choice tasks.

---

### **.CHS File Layout**

Following is a description of the .CHS format, for use with allocation-based (constant-sum) discrete choice experiments. (Please note that this format may also be used to code discrete choice responses, with the entire allocation given to the item chosen.) Individual data records for respondents are appended to one another; one record follows another. Following is how the single task would appear in a sample .CHS data file:

```

8960 2 6 12 0
4 2
3
2 1 2 3 2 3 7
3 3 3 1 3 1 3
4 2 1 2 1 2 0
.
.
.

```

We'll label the parts of each line and then discuss each part.

**Line 1:**

<b>8960</b>	<b>2</b>	<b>6</b>	<b>12</b>	<b>0</b>
Respondent Number	"Extra" Variables	Number of Attributes	Number of Choice tasks	None option 0=N, 1=Y

*First position (8960):* The first position on the first line of the data file contains the respondent number.

*Second position (2):* The second position contains the number of "extra" variables included in the file. These variables may include the duration of the interview, and any merged segmentation information, which can be useful for selecting subgroups of respondents for special analyses or for weighting. The variables themselves appear in line two of the data file and are described below. (If formatting their own files, most Latent Class users set the number of "extra" variables to zero and omit line 2.)

*Third position (6):* The third position contains the number of attributes in the study.

*Fourth position (12):* The fourth position contains the number of choice tasks included in the questionnaire.

*Final position (0):* The final number indicates whether the "None" option was in effect; 0 = no, 1=yes.

**Line 2:**

<b>4</b>	<b>2</b>
Interview Duration	Segmentation Variable

These "extra" variables are largely a carryover from very early versions of CBC software, and are only used in the Latent Class system for respondent filters or weights. If you specify that there are no "extra" variables on line 1, you can omit this line of data.

The remaining four lines all describe the interview's first task:

**Line 3:**

<b>3</b>
Number of Concepts first task

Line 3 contains one number, which is the number of alternatives (rows of data) in the first task. If one of the alternatives is a "None" option, that row is specified as the final one within each task and is counted in this number.

The next three lines describe the three concepts in the task. The numbers on each line indicate



which attribute level was displayed. Let's look at the first of these lines:

**Line 4:**        **2**        **1**        **2**        **3**        **2**        **3**        **7**

Level Displayed for Each Attribute in First Concept, plus point allocation

This line represents the first concept, and at the end of the line is the respondent's point allocation (7) to this concept. So the numbers in line 4 represent:

First position (2):	level 2 of attribute #1	(Computer B)
Second position (1):	level 1 of attribute #2	(200 MHz Pentium)
Third position (2):	level 2 of attribute #3	(5 lbs)
Fourth position (3):	level 3 of attribute #4	(16 Meg Memory)
Fifth position (2):	level 2 of attribute #5	(1.5 Gbyte hard disk)
Sixth position (3):	level 3 of attribute #6	(\$3,000)
Seventh position (7):	Point allocation for this concept	

Following are a list of the six attributes and their levels, to help in interpreting these lines from the data file:

- 1 Computer A
- 2 Computer B
- 3 Computer C
- 4 Computer D
  
- 1 200 MHz Pentium
- 2 166 MHz Pentium
- 3 133 MHz Pentium
  
- 1 3 lbs
- 2 5 lbs
- 3 7 lbs
  
- 1 64 Meg Memory
- 2 32 Meg Memory
- 3 16 Meg Memory
  
- 1 2 Gbyte hard disk
- 2 1.5 Gbyte hard disk
- 3 1 Gbyte hard disk
  
- 1 \$1,500
- 2 \$2,000
- 3 \$3,000

**Line 5:**        **3**        **3**        **3**        **1**        **3**        **1**        **3**

Level Displayed for Each Attribute in Second Concept, plus point allocation

Line five represents the second concept, and the numbers are interpreted as:

First position (3):	level 3 of attribute #1	(Computer C)
Second position (3):	level 3 of attribute #2	(133 MHz Pentium)
Third position (3):	level 3 of attribute #3	(7 lbs)
Fourth position (1):	level 1 of attribute #4	(64 Meg Memory)
Fifth position (3):	level 3 of attribute #5	(1 Gbyte hard disk)
Sixth position (1):	level 1 of attribute #6	(\$1,500)
Seventh position (3):	Point allocation for this concept	

**Line 6:**      4          2          1          2          1          2                   0  
                  Level Displayed for Each Attribute in Third Concept, plus point allocation

Line six represents the third concept, and the numbers are interpreted as:

First position (4):	level 4 of attribute #1	(Computer D)
Second position (2):	level 2 of attribute #2	(166 MHz Pentium)
Third position (1):	level 1 of attribute #3	(3 lbs)
Fourth position (2):	level 2 of attribute #4	(32 Meg Memory)
Fifth position (1):	level 1 of attribute #5	(2 Gbyte hard disk)
Sixth position (2):	level 2 of attribute #6	(\$2,000)
Seventh position (0):	Point allocation for this concept	

Note: if a "None" concept is present, it is included as the last alternative in the task, with all attribute level codes as "0".

The balance of the respondent's data would consist of lines similar to the last four in our data file fragment, and those lines would have results for each of the other choice tasks.

## 7.3

## Appendix C: Directly Specifying Design Codes in the .CHO or .CHS Files

Some advanced users of Latent Class may want to control the coding of some or all of the independent variables, rather than let Latent Class automatically perform the effects coding based on the integers found in the .CHO or .CHS files. When doing this, you set attribute coding to "User-specified" within the *Attribute Information* tab for any attribute for which you are controlling the coding.

*(Note: many users will find the .csv file format even easier to work with, and the same principles covered in this Appendix also apply for coding .csv files.)*

When you specify that some attributes use "User-specified" coding, you are telling Latent Class that certain or all values found in the .CHO or .CHS files should be used as-is within the design matrix. In the example below, we'll let Latent Class code automatically all but one of the parameters to be estimated. ***Even though we'll only show an example for a .CHO file, the same procedure is followed within the relevant format for the .CHS file.***

Consider the following segment from a studyname.CHO file, representing the first of 18 tasks for respondent number 2001 (if needed, please refer to [Appendix B](#) that describes the layout for the studyname.CHO file):

```

2001 7 6 18 0
6 1 2 3 4 5 6
5 1
2 1 2 1 2 3
1 1 3 1 1 2
1 3 3 2 2 1
2 3 2 2 1 4
3 2 1 1 2 3
3 99

```

Attribute six in this example is Price (we've bolded the price codes for the five product concepts available within this task). Price currently is coded as 4 levels. Let's imagine that the prices associated with these levels are:

```

Level 1 $10
Level 2 $20
Level 3 $30
Level 4 $50

```

In our example above, the prices are \$10, \$20, \$30, \$50. To zero-center the codes, we first subtract from each value the mean of the values. The mean is 27.5. Therefore, the zero-coded values are:

```

10 - 27.5 = -17.5
20 - 27.5 = -7.5
30 - 27.5 = 2.5
50 - 27.5 = 22.5

```

Now that we have zero-coded the values for Price, we are ready to inform Latent Class regarding this coding procedure and place the values within the studyname.CHO file.

To specify the presence of user-defined coding for (in this example) the Price attribute, the user should:

1. From the *Attribute Information* tab, right-click the Price attribute label, and select **Change Coding Method / User-specified**. This tells Latent Class to use the values (after zero-centering) currently found in the .CHO or .CHS file for this attribute within the design matrix.
2. Next, the user-defined coded values for Price need to be placed within the studyname.CHO file. Recall that the default coding for the studyname.CHO file for respondent 2001 looked like:

```

2001 7 6 18 0
6 1 2 3 4 5 6
5 1
2 1 2 1 2 3
1 1 3 1 1 2
1 3 3 2 2 1
2 3 2 2 1 4
3 2 1 1 2 3
3 99

```

Substitute the coded independent variables for Price in the studyname.CHO file as follows (you'll typically use a data processing software and an automated script to do this):

```

2001 7 6 18 0
6 1 2 3 4 5 6
5 1
2 1 2 1 2 2.5
1 1 3 1 1 -7.5
1 3 3 2 2 -17.5
2 3 2 2 1 22.5
3 2 1 1 2 2.5
3 99

```

Note that all the values must be space-delimited within the studyname.CHO file. Make sure there is at least one space between all values. The values may include up to six decimal places of precision.

In this example, there are only four discrete values used for price. We did this for the sake of simplicity. However, this coding procedure can support any number of unique values representing a continuous variable in the design matrix.

## 7.4

## Appendix D: Analyzing Alternative-Specific and Partial-Profile Designs

### Introduction

The Latent Class System analyzes data from the studyname.CHO or studyname.CHS files, which are in ASCII (text-only) format. The studyname.CHO file is automatically generated by the CBC system, but you can create your own studyname.CHO or studyname.CHS files and analyze results from surveys that were designed and conducted in other ways. *(Note: many users will find the .csv file format even easier to work with, and the same principles covered in this Appendix also apply for coding .csv files.)*

### Alternative-Specific Designs

Some researchers employ a specialized type of choice-based conjoint design wherein some alternatives (i.e. brands) have their own unique set of attributes. For example, consider different ways to get to work in the city: cars vs. buses. Each option has its own set of product features that are uniquely associated with that particular mode of transportation. These sorts of dependent attributes have also been called "brand-specific attributes," though as our example illustrates, they aren't always tied to brands.

Consider the following design:

<u>Car:</u>	<u>Bus:</u>
Gas: \$1.25/ gallon Gas: \$1.50/ gallon Gas: \$1.75/ gallon	Picks up every 20 min. Picks up every 15 min. Picks up every 10 min. Picks up every 5 min.
Company-paid parking Parking lot: \$5.00/day Parking lot: \$7.00/day	25 cents per one-way trip 50 cents per one-way trip 75 cents per one-way trip \$1.00 per one-way trip
Light traffic report Moderate traffic report Heavy traffic report	

There are actually six different attributes in this design:

1. Mode of transportation (2 levels: Car/Bus)
2. Price of gas (3 levels)
3. Car parking (3 levels)
4. Traffic report (3 levels)
5. Bus frequency (4 levels)
6. Bus fare (4 levels)

Attributes two through four never appear with bus concepts, and attributes five and six never appear with car concepts.

In the studyname.CHO and studyname.CHS files (described in detail in [Appendix B](#)), there is a row of coded values describing each product concept displayed in each task. Consider a two-alternative task with the following options:

Car, Gas: \$1.25/ gallon, Parking lot: \$5.00/ day, Light traffic report  
 Bus, Picks up every 10 min., 50 cents per one-way trip

Again, there are six total attributes used to describe two different modes of transportation. The two alternatives above would be coded as follows in the studyname.cho or studyname.chs files:

1	1	2	1	0	0
2	0	0	0	3	2

Note that the attributes that do not apply to the current concept are coded as a 0 (zero).

---

## Analyzing Partial-Profile Designs

Partial-profile designs display only a subset of the total number of attributes in each task. For example, there might be 12 total attributes in the design, but only five are displayed in any given task. As with coding attribute-specific designs, we specify a level code of 0 (zero) for any attribute that is not applicable (present) in the current product concept.

---

## Analyzing More Than One Constant Alternative

Some discrete choice designs include more than one constant alternative. These alternatives are typically defined using a single statement that never varies. With the transportation example above, other constant alternatives in the choice task might be: "I'd carpool with a co-worker" or "I'd walk to work." Multiple constant alternatives can be included in the design and analyzed with the Latent Class System. If there is more than one constant alternative, one appropriate coding strategy is to represent these as additional levels of another attribute. For example, in the transportation example we've been using, rather than specifying just two levels for the first attribute (Car, Bus), we could specify four codes:

1	Car
2	Bus
3	I'd carpool with a co-worker
4	I'd walk to work

You should specify four alternatives per task to accommodate the car, bus and the two constant alternatives. To code the task mentioned in the previous example plus two constant alternatives in either the studyname.CHO or studyname.CHS files, you would specify:

1	1	2	1	0	0
2	0	0	0	3	2
3	0	0	0	0	0
4	0	0	0	0	0

It is worth noting that the advanced coding strategies outlined in this appendix are also useful within CBC's standard logit, CBC/HB and ICE systems. Though these systems cannot generate designs automatically or questionnaires for these advanced designs, they can analyze choice data files coded to reflect them.



## 7.5

## Appendix E: How Constant Sum Data Are Treated in Latent Class

### Introduction

Conjoint analysis has been an important marketing research technique for several decades. In recent years, attention has focused on the analysis of choices, as opposed to rankings or ratings, giving rise to methodologies known as "Discrete Choice Analysis" or "Choice-Based Conjoint Analysis."

Choice analysis has the advantage that experimental choices can mimic actual buying situations more closely than other types of conjoint questions. However, choices also have a disadvantage: inefficiency in collecting data. A survey respondent must read and understand several product descriptions before making an informed choice among them. Yet, after all of that cognitive processing the respondent provides very scanty information, consisting of a single choice among alternatives. There is no information about intensity of preference, which products would be runners up, or whether any other products would even be acceptable.

Many researchers favor the comparative nature of choice tasks, but are unwilling to settle for so little information from each of them. This leads to the notion of asking respondents to answer more fully by allocating "constant sum scores" among the alternatives in each choice set rather than by just picking a single one. For example, a survey respondent might be given 10 chips and asked to distribute them among alternatives in each choice set according to his/her likelihood of purchasing each. Alternatively, the respondent might be asked to imagine the next 10 purchase occasions, and to estimate how many units of each alternative would be purchased in total on those occasions. Such information can be especially informative in categories where the same individuals often choose a mix of products, such as breakfast cereals or soft drinks. Constant sum scores are particularly appropriate when it is reasonable for the respondent to treat the units as probabilities or frequencies.

Constant sum scores certainly can provide more information than mere choices, although they are not without shortcomings of their own. One disadvantage is that it takes respondents longer to answer constant sum tasks than choice tasks (Pinnell, 1999). Another is that one can't be sure of the mental process a respondent uses in providing constant sum data. The requirement of summing to 10 or some other total may get in the way of accurate reporting of relative strengths of preference. Finally, since respondents' processes of allocating points are unknown, it's not clear what assumptions should be made in analyzing the resulting data.

Our strategy for analyzing constant sum data begins with the notion that each constant sum point is the result of a separate choice among alternatives. Suppose 10 points are awarded among three alternatives, with the scores [7, 3, 0]. We could treat this as equivalent to 10 repeated choice tasks, in which the first alternative was chosen 7 times, the second chosen 3 times, and the third never chosen. But, there is a difficulty with this approach: one can't be sure that constant sum points are equivalent to an aggregation of independent choices. Perhaps this respondent is inclined always to give about 7 points to his/her first choice and about 3 points to his/her second choice. Then we don't have 10 independent choices, but something more like two.

If a respondent conscientiously makes 10 independent choices in allocating 10 points, then those data contain more information and should receive greater weight than if he/she uses some simpler method. Likewise, if a respondent were always to allocate points among products without really reflecting on the actual likelihood of choice, those data contain less information, and should be given less weight in estimation of his/her values.



We deal with this problem by asking the analyst to estimate the amount of weight that should be given to constant sum points allocated by respondents. We provide a default value, and our analysis of synthetic data sets shows that we do a creditable job of estimating respondent part worths when using this default value, although the analysis can be sharpened if the user can provide a more precise estimate of the proper weight.

### How Constant Sum Data Are Coded

*In earlier versions of Latent Class, we used a less efficient process for estimating part worths from allocation-based CBC data. It involved expanding the number of choice tasks to be equal to the number of product alternatives that had received allocation of points. We are indebted to Tom Eagle of Eagle Analytics for showing us an equivalent procedure that is much more computationally efficient and therefore considerably faster.*

First, although we have spoken of "constant sum data," that is something of a misnomer. There is no requirement that the number of points allocated in each task have the same sum. During estimation, the data from each task are automatically normalized to have the same sum, so each task will receive the same weight regardless of its sum. However, to avoid implying that their sums must be constant, we avoid the term "constant sum" in favor of "chip allocation" in the balance of this appendix.

Latent Class reads the **studyname.CHS** file (or data from a .CSV file) which contains chip allocation data in text format and produces a binary file for faster processing. The simplest of procedures might treat chip allocation data as though each chip were allocated in a separate choice task. If, for example, the chips allocated to alternatives A, B, and C were [A = 7, B = 3, C = 0] then we could consider that 10 repeated choice tasks had been answered, with seven answered with choice of A and three answered with choice of B.

In the latent class procedure we compute the likelihood of each respondent's data, conditional on the current estimates of the part worths utilities for each of the classes as well as the sizes of the classes. This likelihood consists of a series of probabilities multiplied together, each being the probability of a particular choice response. If the chips allocated within a task have the distribution [A = 7, B = 3, C = 0], then the contribution to the likelihood **for that task** is

$$P_a * P_a * P_a * P_a * P_a * P_a * P_a * P_b * P_b * P_b$$

which may be rewritten as:

$$P_a^7 * P_b^3 \tag{1}$$

where  $P_a$  is the likelihood of choosing alternative a from the set and  $P_b$  is the likelihood of choosing alternative b from the set. According to the logit rule:

$$P_a = \exp(U_a) / \text{SUM}(\exp(U_j)) \tag{2}$$

and

$$P_b = \exp(U_b) / \text{SUM}(\exp(U_j)) \tag{3}$$

where  $U_a$  is the total utility for concept a,  $U_b$  is the total utility for concept b, and j is the index for each of the concepts present in the task.

Substituting the right-hand side of equations 2 and 3 into equation 1, we obtain an alternate form for expressing the likelihood of our example choice task where 7 chips are given to A and 3 chips to B:

$$(\exp(U_a) / \text{SUM}(\exp(U_j)))^7 * (\exp(U_b) / \text{SUM}(\exp(U_j)))^3$$

And, an equivalent expression is:

$$\exp(U_a)^7 * \exp(U_b)^3 / \text{SUM}(\exp(U_j))^{10} \quad (4)$$

There is a potential problem when so many probabilities are multiplied together (equivalently, raising the probability of the alternative to the number of chips given to that alternative). If the respondent really does answer by allocating each chip independently, then the likelihood *should* be the product of all those probabilities. But if the data were really generated by some simpler process, then it would seem more appropriate to recognize this.

For this reason we give the user a parameter which we describe as "Total task weight." If the user believes that respondents allocated ten chips independently, he should use a value of ten. If he believes that the allocation of chips within a task are entirely dependent on one another (such as if every respondent awards all chips to the same alternative) he should use a value of one. Probably the truth lies somewhere in between, and for that reason we suggest 5 as a default value.

We use the Task Weight in the following way.

Rather than assume that each chip represents an independent choice event, we first normalize the number of chips allocated within each task by dividing the exponents in equation 4 by the total number of chips allocated. This simplifies the formula to:

$$\exp(U_a)^{0.7} * \exp(U_b)^{0.3} / \text{SUM}(\exp(U_j))$$

We can then apply the task weight to appropriately weight the task. Assuming the researcher wishes to apply a task weight of 5, the new formula to represent the probability of this task is:

$$[ \exp(U_a)^{0.7} * \exp(U_b)^{0.3} / \text{SUM}(\exp(U_j)) ]^5$$

Which may be rewritten as:

$$\exp(U_a)^{(0.7*5)} * \exp(U_b)^{(0.3*5)} / \text{SUM}(\exp(U_j))^5$$

Mathematically, this is identical to the likelihood expression based on expanded tasks that we used in earlier versions of our latent class software, but it avoids expanding the tasks and is more efficient computationally.

## 7.6

## Appendix F: Utility Constraints for Attributes Involved in Interactions

Latent Class can constrain utilities to conform to user-specified monotonicity constraints within each individual. Constraints can also be used for attributes involved in interaction terms.

### When Both Attributes Are Categorical:

Consider two attributes both with known preference order ( $\text{level1} < \text{level2} < \text{level3}$ ) involved in an interaction effect. Main effects and first-order interaction effects may be estimated in Latent Class. Effects coding results in zero-centered main effects that are independent of the zero-centered first-order effects.

To impose monotonicity constraints, for each individual, construct a table containing the joint utilities when two levels from each attribute combine. In the joint effects table below, A is equal to the main effect of Att1\_Level1 plus the main effect of Att2\_Level1 plus the interaction effect of Att1\_Level1 x Att2\_Level1.

	Att2_Level1	Att2_Level2	Att2_Level3
Att1_Level1	A	B	C
Att1_Level2	D	E	F
Att1_Level3	G	H	I

Given that these two attributes have known *a priori* rank order of preference from "worst to best," we expect the following utility relationships:

A<B<C  
 D<E<F  
 G<H<I  
 A<D<G  
 B<E<H  
 C<F<I

For any pair of joint utilities that violates these preference orders, we tie the values in the joint effects table by setting both offending elements equal to their average. We recursively tie the values, because tying two values to satisfy one constraint may lead to a violation of another. The algorithm cycles through the constraints repeatedly until they are all satisfied.

After constraining the values in the joint table, the new row and column means represent the new constrained main effects. For example, Let J equal the mean of (A, B, C); J is the new main effect for Att1\_Level1. Let M equal the mean of (A, D, G); M is the new main effect for Att2\_Level1.

Finally, we compute the constrained first-order interactions. Subtract the corresponding constrained main effects from each cell in the joint effects table to arrive at the constrained interaction effect. For example, assume that J is the constrained main effect for Att1\_Level1 and M is the constrained main effect for Att2\_Level1. The constrained interaction effect Att1\_Level1 x Att2\_Level1 is equal to A-J-M.

The example above assumed full rank-order constraints within both attributes. The same methodology is applied for constraining selected relationships within the joint utility table. For example, if the only constraint was Att1\_Level1>Att1\_Level2, then the only joint effects to be constrained are A>D, B>E, and C>F.

---

### For Categorical x Linear Attributes:

Assume two attributes, one categorical (with three levels) and one linear term. Assume the following constraints are in place:

Att1\_Level1>Att1\_Level2  
Att2 is negative

The main effects for the categorical attribute may be considered (and constrained) independently of the effects involving the linear term (we can do this because the elements in the X matrix for Att2 are zero-centered). Constrain the main effects for the categorical levels of Att1, by tying offending items (by setting offending values equal to their average).

Next, we build an effects table, representing the effect of linear attribute Att2, conditional on levels of Att1 (and independent of the main effect for Att1):

	Att2
Att1_Level1	A
Att1_Level2	B
Att1_Level3	C

For example, A is equal to the linear term main effect of Att2 plus the interaction effect Att1\_Level1 x Att2. In other words, A is the level-specific linear effect of Att2 for Att1\_Level1. (Note that we do not add the main effect of categorical term Att1\_Level1 to A).

Next, we constrain any elements A, B, C that are positive to zero.

We re-compute the constrained linear main effect for Att2 as the average of the column. (Example: Let D equal the mean of (A, B, C); the constrained linear main effect for Att2 is equal to D.)

Finally, estimate the constrained interaction effects by subtracting the constrained linear main effect for Att2 from each element. (Example: the constrained interaction effect for Att1\_Level1 x Att2 is equal to A-D. Repeat in similar fashion for all rows).

---

### For Linear x Linear Attributes:

Assume two attributes Att1 and Att2, both estimated as linear terms. Assume the following constraints are in place:

Att2 is negative

In this case, if Att2 is found to be positive, we simply constrain Att2 to be zero. No action is taken with the interaction effect.

If both main effects are constrained, we similarly only apply constraints to main effects.



## 7.7 Appendix G: Estimation for Dual-Response "None"

### Introduction

Some researchers have advocated asking the "None" choice as a second-stage question in discrete choice questionnaires (see "Beyond the Basics: Choice Modelling Methods and Implementation Issues" (Brazell, Diener, Severin, and Uldry) in ART Tutorial 2003, American Marketing Association). The "Dual-Response None" technique is an application of the "buy/no buy" response that previous researchers (including McFadden, Louviere, and Eagle) have used as an extension of standard discrete choice tasks since at least the early 1990s, and have modeled with nested logit.

The "Dual-Response None" approach is as follows. Rather than ask respondents to choose among, say, four alternatives {A, B, C and None}; respondents are first asked to choose among alternatives {A, B, and C}, and then next asked if they really would buy the alternative they selected in the first stage (yes/no).

The dual-response None dramatically increases the propensity of respondents to say "None," which many researchers would argue better reflects actual purchase intentions than when the "None" is asked in the standard way. But, no information is lost due to the selection of the "None," since respondents are first asked to discriminate among available products. Thus, the "Dual-Response" none can provide a "safety net": we can estimate a "None" parameter without worrying about the potential decrease in precision of the other parameters if the incidence of "None" usage is quite high.

The "Dual-Response None" has its drawbacks. It adds a little bit more time to each choice task, since respondents must provide two answers rather than one. But, the additional time requirement is minimal, since respondents have already invested the time to become familiar with the alternatives in the choice task. It is also possible that asking the "None" as a separate question may bias the parameters of interest.

Brazell et al. have suggested that the benefits of the dual response seem to outweigh any negatives. They have conducted split-sample experiments with respondents that demonstrate that the parameters (other than the "None") are not significantly different when using the standard "None" vs. "Dual-Response None" formats.

---

### Modeling Dual-Response None in Latent Class

We do not claim to know the absolute best method for modeling choice tasks that use the "Dual-Response None." However, the method we offer in Latent Class seems to work quite well based on recent tests with actual respondent data and holdout choice tasks.

Our approach is quite simple: we model the two choices (the forced choice among alternatives, and the buy/no buy follow-up) as independent choice tasks. In the first task, the choice is among available products (without a "None" alternative available). In the second task, the choice is among available products and the "None" alternative. Failure to pick the "None" alternative in the second stage (a "buy" indication) results in a redundant task. All information may be represented using just the second stage choice task. With that one task, we can indicate the available options, which option the respondent chose, and the fact that the respondent rejected the "None" alternative. Therefore, we omit the redundant first-stage task.

---

## Data Setup in the Latent Class File

We already introduced the .CHO file layout in [Appendix B](#). If you are using a .CHO file generated by the CBC/Web v6 system (or later), then the .CHO file will already include the below modifications, and no additional re-formatting is required. If using another system for implementing "Dual-Response None" questionnaires, make the following modifications:

1. Each respondent records begins with a header that has five values. Set the fifth value equal to "2."
2. In the standard .CHO layout, the coding of each task is completed by a line with two values: "Choice" and "Task Duration." With the "Dual-Response None" format, each choice task is completed by a line with four values, such as:

<b>3</b>	<b>27</b>	<b>1</b>	<b>4</b>
1st stage	Task	Buy=1	Task
Choice	Duration	No Buy=2	Duration

The first two values contain information about the first-stage task (the choice among available alternatives) and the time (in seconds) to make that choice. This respondent chose the third alternative, and it took 27 seconds to make that selection. The second two values contain information about the "Dual-Response None" question. The first of those values is coded as a 1 (I would buy the product chosen in the first stage) or a 2 (I would not buy the product chosen in the first stage). This is followed by the time (in seconds) to make that choice.

Task Duration is not used at all in the estimation, so you can use an arbitrary integer if you wish.

## 7.8 Appendix H: Estimation for MaxDiff Experiments

Latent Class software may be used for estimating utilities for MaxDiff (best/worst) experiments. MaxDiff experiments are useful for scaling multiple items and performing segmentation research (Sa Lucas 2004, Cohen 2003). In MaxDiff experiments, researchers measure often twenty or more total items, and respondents evaluate choice sets involving typically four to six items at a time, selecting which item is "best" (or most important) and which item is "worst" (or least important). An example is given below:

Please consider dining experiences in fast food restaurants. Among these attributes, which is the most and the least important to you?		
Which is <u>Most</u> Important?		Which is <u>Least</u> Important?
<input type="radio"/>	Good tasting food	<input type="radio"/>
<input type="radio"/>	Offers healthy selections	<input type="radio"/>
<input type="radio"/>	Friendly service	<input type="radio"/>
<input type="radio"/>	Fun atmosphere	<input type="radio"/>

Each respondent typically completes a dozen or more sets (tasks) like the one above, where the items within tasks vary according to an experimental design plan. Across the questionnaire, all items being studied are represented often multiple times for each respondent. If developing individual-level utilities using HB, we'd generally recommend that each item be displayed three times or more for each respondent (Orme 2005).

### Coding the .CHO File for MaxDiff Experiments

*(Note: many users will find the .csv file format even easier to work with, and the same principles covered in this Appendix also apply for coding .csv files.)*

If using Sawtooth Software's products for MaxDiff analysis, an appropriate .CHO file can be generated automatically. For the interested reader, the format of that file is given below. If you are conducting your own MaxDiff experiment using another tool, you will need to format the data as described below for use in Latent Class software.

Consider a MaxDiff study with the following specifications:

- 8 total items in the study
- 10 sets (tasks) per respondent
- 4 items per set

What sets best/worst data apart from traditional conjoint/choice data is that each set is coded twice: once to represent the item chosen as "best" and once for the item selected "worst." Thus, in our example, even though there are 10 total sets in the study, we code these as 20 separate sets. Each respondent's data occupies multiple lines in the file, and the next respondent follows on the line directly beneath the previous (NO blank lines between respondents).



Assume respondent number 1001 received items 7, 8, 3, and 2 in the first of ten sets. Further assume that item 7 was selected as this respondent's "best" and item 3 as the "worst." The first few lines of this file representing the coding for respondent 1001's first set, should look something like:

```

1001 0 7 20 0
4 1
0 0 0 0 0 0 1
0 0 0 0 0 0 0
0 0 1 0 0 0 0
0 1 0 0 0 0 0
1 99
4 1
0 0 0 0 0 0 -1
0 0 0 0 0 0 0
0 0 -1 0 0 0 0
0 -1 0 0 0 0 0
3 99
(etc. for 9 more sets)

```

The exact spacing of this file doesn't matter. Just make sure it is in text-only format and that you have arranged the data on separate lines, and that the values are separated by at least one space. We describe each line as follows:

**Line 1:**

<b>1001</b>	<b>0</b>	<b>7</b>	<b>20</b>	<b>0</b>
Respondent number 1001	No segmentation variables	7 attributes	20 total sets	No "None"

Lines 2 through 7 reflect the information for the "best" item chosen from set #1.

**Line 2:**

<b>4</b>	<b>1</b>
Next follows a set with 4 items	One selection from this set

**Line 3:**

0 0 0 0 0 0 1

Dummy codes representing the first item in the first set (item 7 in our example). Each value represents an item (less the last item, which is omitted in the coding). The dummy codes are "0" if the item is not present, and "1" if the item is present. Since this row represents item 7, the 7<sup>th</sup> value is specified as 1.

**Line 4:**

0 0 0 0 0 0 0

Dummy codes representing item 8. In our study, if the 8<sup>th</sup> item is present, all seven values are at 0.

Lines 5 and 6 follow the same formatting rules for dummy coding, to code items 3 and 2 in our example. Next follows the line in which the respondent's "best" item is indicated.

**Line 7:**

<b>1</b>	<b>99</b>
The item in row	A filler value (time) of

1 of this set is 99 to be compatible  
best with .CHO format

Lines 8 through 13 reflect the information for the "worst" item chosen from set #1.

**Line 8:**

4 1  
Here follows One selection from  
a set with 4 this set  
items

Lines 9 through 12 reflect the dummy codes (inverted) for the items shown in set one, considered with respect to the "worst" item selected. All values that were "1" in the previous task are now "-1".

**Line 13:**

3 99  
The item in row A filler value (time) of  
3 of this set is 99 to be compatible  
best with .CHO format

---

## Estimating the Model using Latent Class

Open the appropriate .CHO file. On the *Attribute Information* tab, modify all attributes to have "User-specified coding." (Note that there will be k-1 total attributes in the study, representing your k total items in the MaxDiff design.)

Click **Estimate Parameters Now...** from the *Home* tab. Utility values are written to the .LCU and .CSV files. Remember, the utility of the omitted value for each group is 0, and the other items are measured with respect to that omitted item.

---

## A Suggested Rescaling Procedure

The raw utilities from Latent Class estimation are logit-scaled, and typically include both positive and negative values. Furthermore, the spread (scale) of the utilities for each segment differs, depending on the consistency of each segments' choices. It may be easier to present the data to management and also may be more appropriate when using the data in subsequent multivariate analyses if the data are rescaled to range from 0 to 100, following "probability" scaling.

1. Insert the score for the omitted item for each segment (score of 0).
2. Zero-center the weights for each segment by subtracting the average item weight from each weight.
3. To convert the zero-centered raw weights to the 0-100 point scale, perform the following transformation for each item score for each segment:

$$e^{U_i} / (e^{U_i} + a - 1) * 100$$

Where:

$U_i$  = zero-centered raw logit weight for item i

$e^{U_i}$  is equivalent to taking the antilog of  $U_i$ . In Excel, use the formula =EXP( $U_i$ )

a = Number of items shown per set

Finally, as a convenience, we rescale the transformed item scores by a constant multiplier so that they sum to 100.

The logic behind the equation above is as follows: We are interested in transforming raw scores (developed under the logit rule) to probabilities true to the original data generation process (the counts). If respondents saw 4 items at a time in each MaxDiff set, then the raw logit weights are developed consistent with the logit rule and the data generation process. Stated another way, the scaling of the weights will be consistent within the context (and assumed error level) of choices from quads. Therefore, if an item has a raw weight of 2.0, then we expect that the likelihood of it being picked within the context of a representative choice set involving 4 items is (according to the logit rule):

$$e^{2.0}/(e^{2.0} + e^0 + e^0 + e^0)$$

Since we are using zero-centered raw utilities, the expected utility for the competing three items within the set would each be 0.0. Since  $e^0 = 1$ , the appropriate constant to add to the denominator of the rescaling equation above is the number of alternatives minus 1.

## Index

### - A -

ABIC 32  
 AIC 32  
 Alternative-specific attributes 55  
 ATT file 16, 22, 47  
 Attribute information tab 22

### - B -

Batch mode 16  
 Best/worst scaling 66  
 BIC 32

### - C -

CAIC 32  
 Capacity limitations 3  
 CBC/HB 55  
 Chi Square 32  
 Chip allocation 58  
 CHO file 16, 18, 21, 22, 45, 53  
 CHO file format 47  
 Choice Data File tab 21  
 Choice task filter 24  
 CHS file 16, 18, 21, 22, 45, 53  
 CHS file format 47  
 Clusterwise logit 40  
 Consistent Akaike Information Criterion 32  
 Constant-sum data 58  
 Constraints 25, 28  
 CSV file 9, 45  
 CSV File Format 12

### - D -

Dual-response "None" 64

### - E -

Effects coding 18, 22  
 Exclude attribute 22

### - H -

Hierarchical Bayes 11, 42, 55  
 Holdout tasks 24  
 Home tab 18

### - I -

ICE 55  
 ID file 9  
 Ill-conditioned design warning 38  
 Interaction effects 6, 7, 22  
 Iterations 5, 25, 32

### - L -

LCU file 9  
 LCU file format 45  
 Level values 7  
 Likelihood of the data 55  
 Linear coding 7, 22  
 Local minima 38  
 LOG file 9, 25  
 Log-likelihood 32

### - M -

Main effects 6, 18  
 Market simulations 1, 6, 11  
 MaxDiff scaling 22, 66  
 Monotonicity constraints 25, 28

### - N -

None alternative 18, 25, 45, 47, 64

### - O -

Opening a project 16  
 Output files 9

### - P -

Part worth coding 22  
 Partial-profile designs 55  
 Percent certainty 32  
 PXX file 9

### - Q -

Quantitative attributes 22  
 Quickstart instructions 1

### - R -

Random number seed 25  
 Relative Chi Square 32  
 Replications 25

---

Respondent filter 25

Reversals 7, 28

**- S -**

Segments (choosing number of) 36

**- T -**

Tabulate pairs of solutions 25

Task weight 58

**- U -**

User-specified coding 22, 53

Utility constraints 25, 61

**- W -**

Weights 25

What's new in v4.5 4