# Sawtooth Software

***RESEARCH PAPER SERIES***

## Scale Constrained Latent Class

Bryan Orme
Sawtooth Software, Inc.


***1457 E 840 N  Orem, Utah***
***+1 801 477 4700***
***www.sawtoothsoftware.com***

# Scale Constrained Latent Class

Bryan Orme, Sawtooth Software
December, 2013

*Note: Scale is a complex issue within MNL and our treatment here assumes that the standard deviation of the utility vector may be used as a proxy for scale.*

## Background and Motivation

Latent Class for Multinomial Logit (MNL) is a popular procedure for finding segments of respondents with different preferences from choice data such as CBC and MaxDiff. However, one aspect of standard Latent Class analysis that may interfere with some analysts' goals is that it sometimes can form segments that mainly differ in terms of scale (response error) but don't differ much in terms of real preference patterns (Magidson, Vermunt 2007). With MNL, the larger the response error, the smaller the magnitude of the part-worth utilities; the smaller the response error, the larger the magnitude of the part-worth utilities.

As far as we know, we are the first to propose Scale Constrained Latent Class. The capability to enforce utility constraints has been a feature of our Latent Class module since its first release in the mid-1990s. Utility constraints (monotonicity constraints) avoid finding less useful solutions where segments demonstrate reversals in the part-worth utilities (logically out of order preferences). Across group scale constraints are just another kind of constraint and involve only a tiny modification to the Latent Class algorithm.

Researchers can use Latent Class to 1) find segments of respondents with different preferences for strategic segmentation and targeting purposes, 2) estimate part-worth utilities for the different segments that fit the data better than aggregate logit. Scale Constrained Latent Class is for researchers who are mainly focusing on the first purpose: strategic market segmentation. If the goal is fitting the data and making predictions regarding consumer choice using latent class results (rather than using, say, HB utilities within choice simulators), then scale constraints are counterproductive, since they slightly lower the fit to the data.

## Previous Research at the Sawtooth Software Conferences

In 2006, Louviere and Eagle presented a paper at the Sawtooth Software conference calling attention to the fact that logit-based models involve a confound between scale factor and utilities. Two respondents (or groups of respondents) could have essentially the same preferences, but appear to have different utilities just due to a uniform stretching or shrinking of the utilities (the scale factor). The Sawtooth Software community had long been aware of a related issue in the popular ACA software since the mid-1980s (where the individual-level calibration step using the ratings for calibration concepts could lead to very different scaling of different respondents). Louviere and colleagues had published on the scale factor issue in discrete choice well prior to that 2006 Sawtooth Software presentation.

In 2007, Magidson and Vermunt reported at the Sawtooth Software Conference how the Latent Gold software system can estimate latent class solutions while controlling for scale. Their approach finds latent segments that differ in terms of their part-worth utilities as well at latent subgroups within those segments that differ in terms of their scale parameter. The approach we take here is much simpler than the Latent Gold approach.

## How Scale Constrained Latent Class Analysis Works

The standard Latent Class procedure consists of the following steps:

1. Set the number of clusters (classes), and choose a random initial part-worth solution for each cluster with values in the interval -.1 to .1.

2. Use each group's logit coefficients to fit each respondent's data and (per the logit rule) 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 group sizes.) Compute the likelihood of the data, given those estimates.

5. Repeat steps 2 through 4 until the improvement in likelihood is sufficiently small.

For Scale Constrained Latent Class analysis, we insert one small step between steps 3 and 4 above. We set the utilities across the classes to have the same scale, equal to the average scale observed in that iteration. That procedure is described below.

Let's imagine that after a given iteration of the latent class procedure, the part-worth utilities for three classes are as follows:

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Attribute 1, Level 1 | +1.05 | +0.22 | -0.83 |
| Attribute 1, Level 2 | +0.33 | -0.67 | +0.52 |
| Attribute 1, Level 3 | -1.38 | +0.45 | +0.31 |
| | | | |
| Attribute 2, Level 1 | +0.35 | +1.06 | +0.79 |
| Attribute 2, level 2 | -0.35 | -1.06 | -0.79 |

There are multiple ways to think of scale factor, one of which is the dispersion of the part-worth utilities. We take the simple approach of using the standard deviation of each utility vector as a proxy for scale. The standard deviation for Class 1 in the table above is the standard deviation across the five values (+1.05, +0.33, -1.38, +0.35, -0.35) or 0.820. The standard deviations across the five part-worths for each of the three classes are:

| | |
|---|---|
| Class 1: | 0.820 |
| Class 2: | 0.768 |
| Class 3: | 0.679 |
| | |
| Average: | 0.756 |

Within each iteration of the Latent Class procedure, we constrain the scale for all three classes to the average scale of the three classes, by multiplying the vectors of part-worths by the following scale factor adjustments:

| | |
|---|---|
| Class 1: | 0.756/0.820 = 0.922 |
| Class 2: | 0.756/0.768 = 0.984 |
| Class 3: | 0.756/0.679 = 1.113 |

After multiplying the part-worth utilities for each class by its scale factor adjustment, the new scale constrained part-worth utilities are:

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Attribute 1, Level 1 | +0.97 | +0.22 | -0.92 |
| Attribute 1, Level 2 | +0.30 | -0.66 | +0.58 |
| Attribute 1, Level 3 | -1.27 | +0.44 | +0.35 |
| | | | |
| Attribute 2, Level 1 | +0.32 | +1.04 | +0.88 |
| Attribute 2, level 2 | -0.32 | -1.04 | -0.88 |
| | | | |
| Standard Deviation: | 0.756 | 0.756 | 0.756 |

After the scale adjustment, the standard deviation of the part-worth utility values for each class is 0.756. The likelihood of the solution is evaluated using the constrained utility vectors. In each subsequent iteration, the part-worths for the classes are again scale constrained, though the magnitude of the parameters (as expressed by the standard deviation) tend to increase from one iteration to the next as the updated utilities provide better fit to the data.

Note: if interaction terms or None parameters are involved, these also are used to compute the standard deviation of the vector of part-worth utilities for each class.

## Example Using Artificial Data Set

To demonstrate how Scale Constrained Latent Class works, we constructed a CBC data set with known part-worth utilities for three groups of respondents of known size. The true group sizes and part-worth utilities are as follows:

**True 3-Group Solution**

| | Group 1<br>N=100<br>33.3% | Group 2<br>N=100<br>33.3% | Group 3<br>N=100<br>33.3% |
|---|---|---|---|
| Attribute 1, Level 1 | 1 | -1 | 0 |
| Attribute 1, Level 2 | 0 | 0 | -3 |
| Attribute 1, Level 3 | -1 | 1 | 3 |
| | | | |
| Attribute 2, Level 1 | 2 | 2 | -1.5 |
| Attribute 2, Level 2 | 0 | 0 | 0 |
| Attribute 2, Level 3 | -2 | -2 | 1.5 |

We are somewhat devious with this constructed example, because in reality Group 3 is a mixture of 50 respondents with double the scale factor (half the response error) as the other 50 respondents, but the same pattern of preferences:

| | Group 3<br>High Error<br>N=50<br>16.7% | Group 3<br>Low Error<br>N=50<br>16.7% |
|---|---|---|
| Attribute 1, Level 1 | 0 | 0 |
| Attribute 1, Level 2 | -2 | -4 |
| Attribute 1, Level 3 | 2 | 4 |
| | | |
| Attribute 2, Level 1 | -1 | -2 |
| Attribute 2, Level 2 | 0 | 0 |
| Attribute 2, Level 3 | 1 | 2 |

Our expectation is that standard Latent Class will probably recover the data well for the three-group solution, but for a 4-group solution it will likely try to break out the High Error respondents from the Low Error respondents. The fact that the true utilities for the 3-group solution reflect different scale factors across the groups is at odds with the scale constraint algorithm which assumes that each group has part-worth utilities with equal magnitude. We will see whether this wrinkle poses problems for the scale constrained algorithm to recover the true nature of the 3-group solution. We will also see whether Scale Constrained Latent Class can avoid partitioning group 3 into separate groups (if a 4-group solution is requested) based mainly on scale factor differences.

We generated a CBC questionnaire for two attributes each with three levels, with 12 tasks, each task containing 3 concepts. We generated 300 robotic respondents, divided them into groups as described

above, and had those robotic respondents "answer" each choice task within the CBC questionnaire according to the true group utilities plus Gumbel error of the default standard deviation[1].

We ran a standard Latent Class solution (using Sawtooth Software's Latent Class tool), with 10 replicates where each replicate uses a different random seed (where we use the one replicate with the highest fit to the data).

Below is the 3-group standard Latent Class solution for the artificially created dataset in the left columns and the original true data are shown for reference in the right columns, shaded in grey:

**Standard Latent Class Solution (3 Groups)**

| | **Estimated Solution from Synthetic Data** | | | | **True Solution** | | |
|---|---|---|---|---|---|---|---|
| | Group 1<br>33.2% | Group 2<br>33.4% | Group 3<br>33.3% | | Group 1<br>N=100<br>33.3% | Group 2<br>N=100<br>33.3% | Group 3<br>N=100<br>33.3% |
| Attribute 1, Level 1 | 1.03 | -0.89 | 0.14 | | 1 | -1 | 0 |
| Attribute 1, Level 2 | -0.06 | 0.08 | -2.94 | | 0 | 0 | -3 |
| Attribute 1, Level 3 | -0.96 | 0.98 | 2.80 | | -1 | 1 | 3 |
| | | | | | | | |
| Attribute 2, Level 1 | 2.02 | 2.03 | -1.46 | | 2 | 2 | -1.5 |
| Attribute 2, Level 2 | 0.01 | 0.07 | -0.06 | | 0 | 0 | 0 |
| Attribute 2, Level 3 | -2.03 | -2.10 | 1.52 | | -2 | -2 | 1.5 |

LL=-2,046.16

Despite the response error that the robotic respondents used while answering the questions, standard Latent Class has recovered the three groups quite faithfully, both in terms of group size and also in terms of absolute utilities.

[1] Standard deviation π/SQRT(6), or approximately 1.2825.

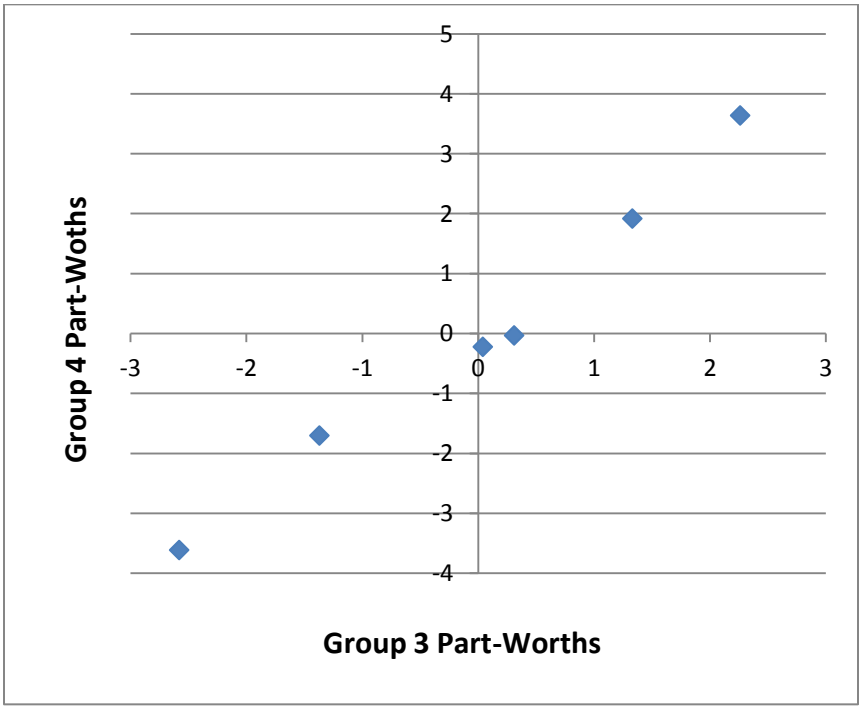
Now, we'll see what happens with the 4-group solution. Will standard Latent Class create a fourth segment by distinguishing between the high and low error respondents that we've mixed into Group 3?

**Standard Latent Class Solution (4 Groups)**

| | Estimated Solution from Synthetic Data | | | | | True Solution | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Group 1 33.2% | Group 2 33.5% | Group 3 14.9% | Group 4 18.4% | | Group 1 N=100 33.3% | Group 2 N=100 33.3% | Group 3 High Error N=50 16.7% | Group 3 Low Error N=50 16.7% |
| Attribute 1, Level 1 | 1.03 | -0.89 | 0.31 | -0.03 | | 1 | -1 | 0 | 0 |
| Attribute 1, Level 2 | -0.06 | 0.08 | -2.58 | -3.61 | | 0 | 0 | -2 | -4 |
| Attribute 1, Level 3 | -0.96 | 0.98 | 2.26 | 3.64 | | -1 | 1 | 2 | 4 |
| | | | | | | | | | |
| Attribute 2, Level 1 | 2.02 | 2.03 | -1.37 | -1.70 | | 2 | 2 | -1 | -2 |
| Attribute 2, Level 2 | 0.01 | 0.07 | 0.04 | -0.22 | | 0 | 0 | 0 | 0 |
| Attribute 2, Level 3 | -2.03 | -2.10 | 1.33 | 1.92 | | -2 | -2 | 1 | 2 |

LL=-2,042.94

Groups 1 and 2 are essentially the same as before, faithfully recovering the true group sizes and preferences. As expected, Latent Class splits Group 3 into two separate groups according to scale (representing them as groups 3 and 4)—though both the true group sizes and utilities for the low and high error groups are not as cleanly recovered as with groups 1 and 2. We've scatter-plotted the part-worth utilities for estimated groups 3 and 4 below.



The correlation between Group 3 and 4's part-worth utilities is 0.99, confirming that that these two groups have essentially the same pattern of preferences and standard Latent Class has partitioned groups 3 and 4 mainly based on scale. Most researchers would not want to see segmentation solutions like this where the main difference between two segments represented just an increase or decrease in scale (response error).

Next, we re-estimated latent class with scale constraints. As before, we used 10 replicates, each using different random starting seeds. The scale constrained 3-group solution shown alongside the true data structure is given below:

**Scale Constrained Latent Class Solution (3 Groups)**

| | **Estimated Solution from Synthetic Data** | | | | **True Solution** | | |
|---|---|---|---|---|---|---|---|
| | Group 1<br>33.2% | Group 2<br>33.4% | Group 3<br>33.3% | | Group 1<br>N=100<br>33.3% | Group 2<br>N=100<br>33.3% | Group 3<br>N=100<br>33.3% |
| Attribute 1, Level 1 | 1.17 | -1.02 | 0.10 | | 1 | -1 | 0 |
| Attribute 1, Level 2 | -0.07 | -0.10 | -2.34 | | 0 | 0 | -3 |
| Attribute 1, Level 3 | -1.11 | 1.12 | 2.24 | | -1 | 1 | 3 |
| | | | | | | | |
| Attribute 2, Level 1 | 2.31 | 2.31 | -1.17 | | 2 | 2 | -1.5 |
| Attribute 2, Level 2 | 0.01 | 0.07 | -0.05 | | 0 | 0 | 0 |
| Attribute 2, Level 3 | -2.32 | -2.38 | 1.21 | | -2 | -2 | 1.5 |

LL=-2,067.40

Notice that the Scale Constrained Latent Class procedure also captures the true solution quite faithfully in terms of group sizes and pattern of part-worth utilities, though the magnitude of the estimated part-worth utilities isn't quite the same as the original true solution. That's because Scale Constrained Latent Class forces the scale to be same across the classes, whereas the original true utilities for the three groups differed in scale (Group 3 had larger scale than Groups 1 and 2). All three vectors of part-worth utilities for the estimated solution in the table above have a constant standard deviation.

Also notice in the table above that fit is a little bit lower for Scale Constrained than for standard Latent Class. Standard latent class has a fit of LL=-2,046.16 whereas the fit for Scale Constrained Latent Class is -2,067.40 (Log-Likelihoods closer to zero mean better fit). Given that the null LL is -3,955.00, this means that the Scale Constrained Latent Class solution fits the data 99.1% as well as standard Latent Class. Similar to how utility constraints usually lead to slightly decreased fit relative to unconstrained solutions, Scale Constrained Latent Class *always* leads to a slightly decreased fit to the data.

Finally comes the real test for Scale Constrained Latent Class. We are hoping that Scale Constrained Latent Class will not try to partition group 3 along the dimension of scale (distinguishing high and low error respondents who are mixed within that group, yet have the same underlying pattern of part-worth utility preferences).

Again, recall, we're forcing a four-group solution, even though only three true groups exist in the data (aside from the scale factor differences among respondents within Group 3). The four-group solution for Scale Constrained Latent Class is:

**Scale Constrained Latent Class (4 Groups)**

| | Estimated Solution from Synthetic Data | | | | | True Solution | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Group 1 16.8% | Group 2 33.0% | Group 3 33.3% | Group 4 16.9% | | Group 1 N=100 33.3% | Group 2 N=100 33.3% | Group 3 High Error N=50 16.7% | Group 3 Low Error N=50 16.7% |
| Attribute 1, Level 1 | 1.38 | -1.02 | 0.10 | 0.85 | | 1 | -1 | 0 | 0 |
| Attribute 1, Level 2 | -0.19 | -0.10 | -2.32 | 0.06 | | 0 | 0 | -2 | -4 |
| Attribute 1, Level 3 | -1.19 | 1.12 | 2.22 | -0.91 | | -1 | 1 | 2 | 4 |
| | | | | | | | | | |
| Attribute 2, Level 1 | 2.34 | 2.29 | -1.16 | 2.17 | | 2 | 2 | -1 | -2 |
| Attribute 2, Level 2 | -0.31 | 0.06 | -0.05 | 0.42 | | 0 | 0 | 0 | 0 |
| Attribute 2, Level 3 | -2.04 | -2.36 | 1.20 | -2.58 | | -2 | -2 | 1 | 2 |

LL=-2,063.14

True Groups 2 and 3 are recovered very faithfully (though the scale is constrained to be equal across groups, so the utilities are stretched a bit from the original true utilities). Notice that Scale Constrained Latent Class keeps Group 3 respondents together (even though it represents a mixture of high and low error respondents), because they only differ by scale. But, we required that a fourth group somehow be partitioned, so something has to give. In this case, true Group 1 is split into two separate (yet fairly similar) segments (estimated groups 1 and 4 in the table above).

We wondered whether it was just luck that when forcing Scale Constrained Latent Class to create 4 segments out of 3 true segments that it happened to split a different group apart instead of Group 3. So, we repeated the analysis 9 additional times, creating 9 new artificial data sets by using different Gumbel error draws for the robotic respondents.

**"Something Has to Give"**
**How Three True Groups Are Partitioned into Four**

| | Standard Latent Class | Scale Constrained Latent Class |
|---|---|---|
| Gumbel Draws #1 | Split True Group 3 | Split True Group 1 |
| Gumbel Draws #2 | Split True Group 3 | Split True Group 2 |
| Gumbel Draws #3 | Split True Group 3 | Split True Group 2 |
| Gumbel Draws #4 | Split True Group 1 | Split True Group 2 |
| Gumbel Draws #5 | Split True Group 3 | Split True Group 1 |
| Gumbel Draws #6 | Split True Group 3 | Split True Group 1 |
| Gumbel Draws #7 | Split True Group 2 | Split True Group 2 |
| Gumbel Draws #8 | Split True Group 3 | Split True Group 2 |
| Gumbel Draws #9 | Split True Group 3 | Split True Group 1 |
| Gumbel Draws #10 | Split True Group 3 | Split True Group 2 |

We see that our first observation was very representative of what occurs for this data set when comparing standard Latent Class to Scale Constrained Latent Class. In 8 out 10 cases, standard Latent Class creates a fourth group by partitioning Group 3 principally along the dimension of scale. In no cases did Scale Constrained Latent Class partition Group 3 (which contains the mixture of high and low error respondents), instead forming a fourth group by partitioning either groups 1 or 2.

The reader may wonder about the ability of these two latent class procedures to assign respondents who answered with error back into their original true groups. Across three replicates of this data set representing independent draws of Gumbel errors, standard Latent Class has an average 97.5% hit rate recovery of the true 3-group solution. Scale Constrained Latent Class has a 97.1% hit rate recovery of the true 3-group solution. This essentially is a tie, but we should note that the Scale Constrained Latent Class was at a disadvantage due to the fact that the true group utilities did not have equal magnitude scaling (in terms of standard deviation). If the true groups' utilities all had the same magnitude scale (in terms of their standard deviation), then we'd expect that the Scale Constrained Latent Class would have a slight edge over standard Latent Class in terms of hit rate validity.

## Convergence

In this section, we report on how well Scale Constrained Latent Class converges compared to standard Latent Class. Latent Class procedures are subject to local minima and should be repeated from multiple random starting points to make sure one doesn't use a suboptimal solution. Within each replicate, it typically takes a few dozen iterations to converge on a solution with a reasonable degree of precision. For Sawtooth Software's latent class procedure, we have chosen a stopping point criterion of 0.01. If the log-likelihood fails to improve by more than 0.01 from one iteration to the next, then we stop the iterations.

For the three-group solution we have been describing, we report the log-likelihood at each iteration for 5 different random starting seeds:

**Standard Latent Class**
**History of Iterations**

| Iteration | Seed=1 LL | Seed=2 LL | Seed=3 LL | Seed=4 LL | Seed=5 LL |
|---|---|---|---|---|---|
| 0 | -3955.00 | -3955.00 | -3955.00 | -3955.00 | -3955.00 |
| 1 | -2658.11 | -2905.93 | -2879.74 | -2703.47 | -2972.98 |
| 2 | -2219.15 | -2384.17 | -2190.65 | -2246.88 | -2231.16 |
| 3 | -2100.32 | -2176.57 | -2056.37 | -2110.31 | -2079.48 |
| 4 | -2063.86 | -2119.61 | -2046.94 | -2056.64 | -2053.87 |
| 5 | -2050.49 | -2079.78 | -2046.29 | -2048.14 | -2047.75 |
| 6 | -2047.30 | -2056.52 | -2046.19 | -2046.67 | -2046.57 |
| 7 | -2046.47 | -2049.01 | -2046.17 | -2046.30 | -2046.27 |
| 8 | -2046.24 | -2046.93 | -2046.16 | -2046.20 | -2046.19 |
| 9 | -2046.18 | -2046.37 | | -2046.17 | -2046.17 |
| 10 | -2046.16 | -2046.21 | | -2046.16 | -2046.16 |
| 11 | -2046.16 | -2046.17 | | | |
| 12 | | -2046.16 | | | |
| 13 | | -2046.16 | | | |

All five starting seeds lead to essentially the same solution (this isn't always the case with real data; these data were constructed with known, clean properties).

Scale Constrained Latent Class demonstrates the following history of convergence for the same dataset and the three-group solution:

**Scale Constrained Latent Class**
**History of Iterations**

| Iteration | Seed=1 LL | Seed=2 LL | Seed=3 LL | Seed=4 LL | Seed=5 LL |
|---|---|---|---|---|---|
| 0 | -3955.00 | -3955.00 | -3955.00 | -3955.00 | -3955.00 |
| 1 | -2683.55 | -2952.64 | -2897.82 | -2716.17 | -3004.79 |
| 2 | -2220.28 | -2392.86 | -2197.56 | -2246.06 | -2239.25 |
| 3 | -2105.31 | -2182.22 | -2070.58 | -2108.64 | -2085.14 |
| 4 | -2084.55 | -2124.88 | -2067.74 | -2073.60 | -2074.50 |
| 5 | -2071.72 | -2089.64 | -2067.55 | -2068.34 | -2068.79 |
| 6 | -2068.56 | -2073.46 | -2067.45 | -2067.65 | -2067.76 |
| 7 | -2067.71 | -2069.04 | -2067.42 | -2067.47 | -2067.50 |
| 8 | -2067.49 | -2067.84 | -2067.41 | -2067.42 | -2067.43 |
| 9 | -2067.43 | -2067.52 | -2067.40 | -2067.41 | -2067.41 |
| 10 | -2067.41 | -2067.44 | | -2067.40 | -2067.41 |
| 11 | -2067.40 | -2067.41 | | | |
| 12 | | -2067.41 | | | |

Modifying the algorithm to hold the scale factor constant across classes in each iteration doesn't seem to affect the speed of convergence for this dataset. For real datasets with higher dimension solutions (e.g. 6-group, 10-group, etc.), our experience so far is that Scale Constrained Latent Class usually converges even faster than standard latent class.

We should note that CBC datasets that involve None concepts lead to less quick and consistent convergence when applying scale constraints. The None parameter can become a relatively large and decisive parameter for defining groups, so constraining the solution such that the standard deviation across all the parameters (including the None) is the same across groups leads to greater losses in model fit compared to non-constrained solutions. This is to be expected.

**Do We Lose Ability to Compare Respondents in Terms of Certainty?**

With the scale-constrained solution described here, we force the groups to have equal scale (where scale is defined as standard deviation of the utility vector). This means that if some groups of respondents were indeed more consistent in their choices than others, such information is lost within the constrained solution. Maybe the researcher would like to be able to know which respondents within a certain segment seem more informed or more confused in their choices. One possible way to obtain that information is to run HB separately and use the fit statistic (RLH, Root Likelihood) that results. This gives a continuous, individual-level fit measure. One could then examine the HB fit statistic

for each respondent involved in a target segment of interest (where that target segment was found via scale-constrained latent class).

## Summary

Latent Class analysts are already familiar with the notion of employing utility constraints (to avoid finding nonsensical segments with utility reversals). Scale constraints are for avoiding segmentation solutions where the preferences between segments differ mostly in terms of scale (response error). Of course, both utility and scale constraints can be applied within the same latent class run.

In this paper, we've demonstrated using an artificial data set that Scale Constrained Latent Class analysis tends to avoid partitioning respondents into separate groups based principally on scale differences. Researchers may find Scale Constrained Latent Class helpful when the goal of the latent class analysis is to find market segments for strategic segmentation purposes. The reduction in model fit is relatively small for constrained scale solutions (especially for CBC questionnaires without a None alternative) and the predictive validity of these constrained solutions is essentially as good as non-constrained standard latent class for the example data set we used.

## References

Louviere, Jordan and Thomas Eagle (2006), "Confound It! That Pesky Little Scale Constant Messes up our Convenient Assumptions," 2006 Sawtooth Software Conference, Sequim, WA, pp 211-228.

Magidson, Jay and Jeroen K. Vermunt (2007), "Removing the Scale Factor Confound in Multinomial Logit Choice Models to Obtain Better Estimates of Preference," 2007 Sawtooth Software Conference, Sequim, WA, pp 139-154.