# Sawtooth Software

*TECHNICAL PAPER SERIES*

# The CBC/HB System
# Technical Paper V5.6

Sawtooth Software, Inc.

*© Copyright 2021, Sawtooth Software, Inc.*
*3210 N. Canyon Rd., Provo, Utah*
*+1 801 477 4700*
*www.sawtoothsoftware.com*

# The CBC/HB System for Hierarchical Bayes Estimation

**Version 5.6**
(Updated March 2021)

## Introduction

The CBC/HB System is software for estimating part worths for Choice-Based Conjoint (CBC) questionnaires. It can use either discrete choices or constant sum (chip) allocations among alternatives in choice sets. Other advanced options include the ability to estimate first-order interactions, linear terms, covariates in the upper-level model, and the ability to handle MaxDiff data.

CBC/HB uses data files that can be automatically exported from Sawtooth Software's Lighthouse Studio or Discover systems. It can also use data collected in other ways, so long as the data conform to the conventions of the .csv or text-only format input files, as described in the appendices of the CBC/HB user manual.

The earliest methods for analyzing choice-based conjoint data (e.g., the 70s and 80s) usually did so by combining data across individuals. Although many researchers realized that aggregate analyses could obscure important aspects of the data, methods for estimating robust individual-level part worth utilities using a reasonable number of choice sets didn't become available until the 90s.

The Latent Class MNL Segmentation Module was offered as the first add-on to our CBC software in the mid-90s, permitting the discovery of groups of individuals who respond similarly to choice questions.

Landmark articles by Allenby and Ginter (1995) and Lenk, DeSarbo, Green, and Young (1996) described the estimation of individual part worths using Hierarchical Bayes (HB) models. This approach seemed extremely promising, since it could estimate reasonable individual part worths even with relatively little data from each respondent. However, it was very intensive computationally. The first applications required as much as two weeks of computational effort, using the most powerful computers available to academics!

In 1997 Sawtooth Software introduced the ICE Module for Individual Choice Estimation, which also permitted the estimation of part worths for individuals and was much faster than HB. In a 1997 paper describing ICE, we compared ICE solutions to those of HB, observing:

> "In the next few years computers may become fast enough that Hierarchical Bayes becomes the method of choice; but until that time, ICE may be the best method available for other than very small data sets."

Over the next few years, computers indeed became faster, and our CBC/HB software soon could handle even relatively large-sized problems in an hour or less. Today, most datasets will take about 10 minutes or less for HB estimation.

HB has been described favorably in many journal articles. Its strongest point of differentiation is its ability to provide estimates of individual part worths given only a few choices by each individual. It does this by "borrowing" information from population information (means and covariances) describing the preferences of other respondents in the same dataset. Although ICE also makes use of information from other individuals, HB does so more effectively, and requires fewer choices from each individual.

Latent Class MNL is also a valuable method for analyzing choice data. Because Latent Class can identify segments of respondents with similar preferences, it is an additional valuable method. Recent research suggests that default HB MNL is actually faster for researchers to use than Latent Class MNL, when one

considers the decisions that should be made to fine-tune Latent Class models and select an appropriate number of classes to use (McCullough 2009).

Our software estimates an HB MNL model using a Monte Carlo Markov Chain algorithm. In the material that follows we describe the HB model and the estimation process. We also provide timing estimates, as well as suggestions about when the CBC/HB System may be most appropriate.

We at Sawtooth Software are not experts in Bayesian data analysis. In producing this software we have been helped by several sources listed in the References. We have benefited particularly from the materials provided by Professor Greg Allenby in connection with his tutorials at the American Marketing Association's Advanced Research Technique Forum, and from correspondences with Professor Peter Lenk.

### Capacity Limitations

Because we anticipate that the CBC/HB System may be used to analyze data from sources other than our Lighthouse Studio or Discover programs, it can handle data sets that are larger than the limits imposed by CBC questionnaires. The CBC/HB System has these limitations:

- The maximum number of parameters to be estimated for any individual 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 CBC/HB System requires a fast computer and a generous amount of storage space, as offered by most every PC that can be purchased today.

There is a great deal of activity writing to the hard disk and reading back from it, which is greatly facilitated by Windows' ability to use extra RAM as a disk cache. The availability of RAM may therefore be almost as critical as sheer processor speed.

## Understanding the CBC/HB System

This section attempts to provide an intuitive understanding of the Hierarchical Bayes method as applied to the estimation of conjoint part worths for this standalone system as well as for HB MNL integrated within Lighthouse Studio. For those desiring a more rigorous treatment, we suggest “Bayesian Data Analysis” (1996) by Gelman, Carlin, Stern, and Rubin.

### Bayesian Analysis

In statistical analysis we consider three kinds of concepts: data, models, and parameters.

- In our context, data are the choices that individuals make.

- Models are assumptions that we make about data. For example, we may assume that a distribution of data is normally distributed, or that variable y depends on variable x, but not on variable z.

- Parameters are numerical values that we use in models. For example, we might say that a particular variable is normally distributed with mean of 0 and standard deviation of 1. Those values are parameters.

Often in conventional (non-Bayesian) statistical analyses, we assume that our data are described by a particular model with specified parameters, and then we investigate whether the data are consistent with those assumptions. In doing this we usually investigate the *probability distribution of the data*, given the assumptions embodied in our model and its parameters.

In Bayesian statistical analyses, we turn this process around. We again assume that our data are described by a particular model and do a computation to see if the data are consistent with those assumptions. But in Bayesian analysis, we investigate the *probability distribution of the parameters*, given the data. To illustrate this idea we review a few concepts from probability theory. We designate the probability of an event A by the notation **p(A)**, the probability of an event B by the notation **p(B)**, and the *joint probability* of both A and B by the notation **p(A,B)**.

Bayesian analysis makes much use of *conditional probability*. Feller (1957) illustrates conditional probability with an example of sex and colorblindness. Suppose we select an individual at random from a population. Let A indicate the event of that individual being colorblind, and let B indicate the event of that individual being female. If we were to do many such random draws, we could estimate the probability of a person being both female and colorblind by counting the proportion of individuals found to be both females and colorblind in those draws.

We could estimate the probability of a female's being colorblind by dividing the number of colorblind females obtained by the number of females obtained. We refer to such a probability as "conditional;" in this case the probability of a person being colorblind is conditioned by the person being female. We designate the probability of a female's being colorblind by the symbol **p(A|B)**, which is defined by the formula:

$$\mathbf{p(A|B) = p(A,B) / p(B).}$$

That is to say, the probability an individual's being colorblind, given that she is female, is equal to the probability of the individual being both female and colorblind, divided by the probability of being female.

Notice that we can multiply both sides of the above equation by the quantity **p(B)** to obtain an alternate form of the same relationship among the quantities:

$$\mathbf{p(A|B)\ p(B) = p(A,B).}$$

We may write a similar equation in which the roles of A and B are reversed:

$$\mathbf{p(B|A)\ p(A) = p(B,A).}$$

and, since the event **(B,A)** is the same as the event **(A,B)**, we may also write:

$$\mathbf{p(B|A)\ p(A) = p(A,B).}$$

The last equation will be used as the model for a similar one below.

Although concrete concepts such as sex and colorblindness are useful for reviewing the concepts of probability, it is helpful to generalize our example a little further to illustrate what is known as "Bayes theorem." Suppose we have a set of data that we represent by the symbol **y**, and we consider alternative hypotheses about parameters for a model describing those data, which we represent with the symbols $\mathbf{H_i}$, with i = 1, 2, ….

We assume that exactly one of those alternative hypotheses is true. The hypotheses could be any set of mutually exclusive conditions, such as the assumption that an individual is male or female, or that his/her age falls in any of a specific set of categories.

Rather than expressing the probability of the data given a hypothesis, Bayes' theorem expresses the probability of a particular hypothesis, $\mathbf{H_i}$, given the data. Using the above definition of conditional probability we can write

$$\mathbf{p(H_i \mid y) = p(H_i, y) / p(y).}$$

But we have already seen (two equations earlier) that:

$$p(H_i, y) = p(y \mid H_i)\, p(H_i)$$

Substituting this equation in the previous one, we get

$$p(H_i \mid y) = p(y \mid H_i)\, p(H_i) / p(y)$$

Since we have specified that exactly one of the hypotheses is true, the sum of their probabilities is unity. The **p(y)** in the denominator, which does not depend on i, is a normalizing constant that makes the sum of the probabilities equal to unity. We could equally well write

$$p(H_i \mid y) \propto p(y \mid H_i)\, p(H_i)$$

where the symbol $\propto$ means "is proportional to."

This expression for the conditional probability of a hypothesis, given the data, is an expression of "Bayes theorem," and illustrates the central principle of Bayesian analysis:

- The probability $p(H_i)$ of the hypothesis is known as its "prior probability," which describes our belief about that hypothesis before we see the data.
- The conditional probability $p(y \mid H_i)$ of the data, given the hypothesis, is known as the "likelihood" of the data, and is the probability of seeing that particular collection of values, given that hypothesis about the data.
- The probability $p(H_i \mid y)$ of the hypothesis, given the data, is known as its "posterior probability." This is the probability of the hypothesis, given not only the prior information about its truth, but also the information contained in the data.

The posterior probability of the hypothesis is proportional to the product of the likelihood of the data under that hypothesis, times the prior probability of that hypothesis. Bayesian analysis therefore provides a way to update estimates of probabilities. We can start with an initial or prior estimate of the probability of a hypothesis, update it with information from the data, and obtain a posterior estimate that combines the prior information with information from the data.

In the next section we describe the hierarchical model used by the CBC/HB System. Bayesian updating of probabilities is the conceptual apparatus that allows us to estimate the parameters of that model, which is why we have discussed the relationship between priors, likelihoods, and posterior probabilities.

In our application of Bayesian analysis, we will be dealing with continuous rather than discrete distributions. Although the underlying logic is identical, we would have to substitute integrals for summation signs if we were to write out the equations. Fortunately, we shall not find it necessary to do so.

## The Hierarchical Model

The Hierarchical Bayes model used by the CBC/HB System is called "hierarchical" because it has two levels.

- At the higher level, we assume that individuals' part worths are described by a multivariate normal distribution. Such a distribution is characterized by a vector of means and a matrix of covariances.
- At the lower level we assume that, given an individual's part worths, his/her probabilities of choosing particular alternatives are governed by a multinomial logit model.

To make this model more explicit, we define some notation. We assume individual part worths have the multivariate normal distribution,

$$\mathbf{ß}_i \sim \mathbf{Normal}(\alpha, \mathbf{D})$$

where:

$\mathbf{ß}_i$ = a vector of part worths for the ith individual

$\alpha$ = a vector of means of the distribution of individuals' part worths

**D** = a matrix of variances and covariances of the distribution of part worths across individuals
At the individual level, choices are described by a multinomial logit model. The probability of the ith individual choosing the kth alternative in a particular task is

$$\mathbf{p}_k = \mathbf{exp}(\mathbf{x}_k' \, \mathbf{ß}_i) \, / \sum_j \mathbf{exp}(\mathbf{x}_j' \, \mathbf{ß}_i)$$

where:

$\mathbf{p}_k$ = the probability of an individual choosing the kth concept in a particular choice task

$\mathbf{x}_j$ = a vector of values describing the jth alternative in that choice task

In words, this equation says that to estimate the probability of the ith person's choosing the kth alternative (by the familiar process used in many conjoint simulators) we:

1. add up the part worths (elements of $\mathbf{ß}_i$) for the attribute levels describing the kth alternative (more generally, multiply the part worths by a vector of descriptors of that alternative) to get the ith individual's utility for the kth alternative
2. exponentiate that alternative's utility
3. perform the same operations for other alternatives in that choice task, and
4. percentage the result for the kth alternative by the sum of similar values for all alternatives.

The parameters to be estimated are the vectors $\mathbf{ß}_i$ of part worths for each individual, the vector $\alpha$ of means of the distribution of worths, and the matrix **D** of the variances and covariances of that distribution.

## Iterative Estimation of the Parameters

The parameters **ß,** $\alpha$**,** and **D** are estimated by an iterative process. That process is quite robust, and its results do not appear to depend on starting values. We take a conservative approach by default, setting the elements of **ß,** $\alpha$**,** and **D** equal to zero.

Given the initial values, each iteration consists of these three steps:

- Using present estimates of the betas and **D**, generate a new estimate of $\alpha$. We assume $\alpha$ is distributed normally with mean equal to the average of the betas and covariance matrix equal to **D** divided by the number of respondents. A new estimate of $\alpha$ is drawn from that distribution (see Appendix A for details).
- Using present estimates of the betas and $\alpha$, draw a new estimate of **D** from the inverse Wishart distribution (see Appendix A for details).

- Using present estimates of $\alpha$ and **D**, generate new estimates of the betas. This is the most interesting part of the iteration, and we describe it in detail below. A procedure known as a "Metropolis Hastings Algorithm" is used to draw the betas. Successive draws of the betas generally provide better and better fit of the model to the data, until such time as increases are no longer possible. When that occurs we consider the iterative process to have converged.

In each of these steps we re-estimate one set of parameters ($\alpha$, **D** or the betas) conditionally, given current values for the other two sets. This technique is known as "Gibbs sampling," and converges to the correct distributions for each of the three sets of parameters.

Another name for this procedure is a "Monte Carlo Markov Chain," deriving from the fact that the estimates in each iteration are determined from those of the previous iteration by a constant set of probabilistic transition rules. This Markov property assures that the iterative process converges.

This process is continued for a large number of iterations, typically several thousand or more. After we are confident of convergence, the process is continued for many further iterations, and the actual draws of beta for each individual as well as estimates of $\alpha$ and **D** are saved to the hard disk. The final values of the part worths for each individual, and also of $\alpha$ and **D**, are obtained by averaging the values that have been saved.

## The Metropolis Hastings Algorithm

We now describe the procedure used to draw each new set of betas, done for each respondent in turn. We use the symbol $\mathbf{ß}_o$ (for "beta old") to indicate the previous iteration's estimation of an individual's part worths. We generate a trial value for the new estimate, which we shall indicate as $\mathbf{ß}_n$ (for "beta new"), and then test whether it represents an improvement. If so, we accept it as our next estimate. If not, we accept or reject it with probability depending on how much worse it is than the previous estimate.

To get $\mathbf{ß}_n$ we draw a random vector **d** of "differences" from a distribution with mean of zero and covariance matrix proportional to **D**, and let $\mathbf{ß}_n = \mathbf{ß}_o + \mathbf{d}$.

We calculate the probability of the data (or "likelihood") given each set of part worths, $\mathbf{ß}_o$ and $\mathbf{ß}_n$, using the formula for the logit model given above. That is done by calculating the probability of each choice that individual made, using the logit formula for $\mathbf{p}_k$ above, and then multiplying all those probabilities together. Call the resulting values $\mathbf{p}_o$ and $\mathbf{p}_n$, respectively.

We also calculate the relative density of the distribution of the betas corresponding to $\mathbf{ß}_o$ and $\mathbf{ß}_n$, given current estimates of parameters $\alpha$ and **D** (that serve as "priors" in the Bayesian updating). Call these values $\mathbf{d}_o$ and $\mathbf{d}_n$, respectively. The relative density of the distribution at the location of a point **ß** is given by the formula

$$\textbf{Relative Density} = \exp[-1/2(\mathbf{ß} - \alpha)' \, \mathbf{D}^{-1}[(\mathbf{ß} - \alpha)]$$

Finally we then calculate the ratio:

$$\mathbf{r} = \mathbf{p}_n \mathbf{d}_n / \mathbf{p}_o \mathbf{d}_o$$

Recall from the discussion of Bayesian updating that the posterior probabilities are proportional to the product of the likelihoods times the priors. The probabilities $\mathbf{p}_n$ and $\mathbf{p}_o$ are the likelihoods of the data given parameter estimates $\mathbf{ß}_n$ and $\mathbf{ß}_o$, respectively. The densities $\mathbf{d}_n$ and $\mathbf{d}_o$ are proportional to the probabilities of drawing those values of $\mathbf{ß}_n$ and $\mathbf{ß}_o$, respectively, from the distribution of part worths, and play the role of priors. Therefore, **r** is the ratio of posterior probabilities of those two estimates of beta, given current estimates of $\alpha$ and **D**, as well as information from the data.

If **r** is greater than or equal to unity, $\beta_n$ has posterior probability greater than or equal to that of $\beta_o$, and we accept $\beta_n$ as our next estimate of beta for that individual. If **r** is less than unity, then $\beta_n$ has posterior probability less than that of $\beta_o$. In that case we use a random process to decide whether to accept $\beta_n$ or retain $\beta_o$ for at least one more iteration. We accept $\beta_n$ with probability equal to **r**.

As can be seen, two influences are at work in deciding whether to accept the new estimate of beta. If it fits the data much better than the old estimate, then $\mathbf{p}_n$ will be much larger than $\mathbf{p}_o$, which will tend to produce a larger ratio. However, the relative densities of the two candidates also enter into the computation, and if one of them has a higher density with respect to the current estimates of $\alpha$ and **D**, then that candidate has an advantage.

If the densities were not considered, then betas would be chosen solely to maximize likelihoods. This would be similar to conducting logit estimation for each individual separately, and eventually the betas for each individual would converge to values that best fit his/her data, without respect to any higher-level distribution. However, since densities are considered, and estimates of the higher-level distribution change with each iteration, there is considerable variation from iteration to iteration. Even after the process has converged, successive estimations of the betas are still quite different from one another. Those differences contain information about the amount of random variation in each individual's part worths that best characterizes them.

We mentioned that the vector **d** of differences is drawn from a distribution with mean of zero and covariance matrix proportional to **D**, but we did not specify the proportionality factor. In the literature, the distribution from which **d** is chosen is called the "jumping distribution," because it determines the size of the random jump from $\beta_o$ to $\beta_n$. This scale factor must be chosen well because the speed of convergence depends on it. Jumps that are too large are unlikely to be accepted, and those that are too small will cause slow convergence.

Gelman, Carlin, Stern, and Rubin (p 335) state: "A Metropolis algorithm can also be characterized by the proportion of jumps that are accepted. For the multivariate normal distribution, the optimal jumping rule has acceptance rate around 0.44 in one dimension, declining to about 0.23 in high dimensions… This result suggests an *adaptive* simulation algorithm."

We employ an adaptive algorithm to adjust the average jump size, attempting to keep the acceptance rate near 0.30 (though you may change that value). The proportionality factor is arbitrarily set at 0.1 initially. For each iteration we count the proportion of respondents for whom $\beta_n$ is accepted. If that proportion is less than 0.3, we reduce the average jump size by ten percent. If that proportion is greater than 0.3, we increase the average jump size by ten percent. As a result, the average acceptance rate is kept close to the target of 0.30.

The iterative process has two stages. During the first stage, while the process is moving toward convergence, no attempt is made to save any of the results. During the second stage we assume the process has converged, and results for hundreds or thousands of iterations are saved to the hard disk. For each iteration there is a separate estimate of each of the parameters. We are particularly interested in the betas, which are estimates of individuals' part worths. We produce point estimates for each individual by averaging the results from many iterations. We can also estimate the variances and covariances of the distribution of respondents by averaging results from the same iterations.

Readers with solid statistical background who are interested in further information about the Metropolis Hastings Algorithm may find the article by Chib and Greenberg (1995) useful.

## How Long Does It Take?

The CBC/HB System is one of the most computationally-intensive software systems that Sawtooth Software has ever produced. It requires a fast computer with plenty of RAM and hard disk storage.

A couple decades ago, run times for CBC/HB v1 commonly were from 3 to 24 hours. With today's faster PCs, run times should take about 10 minutes or less for most data sets. This is a dramatic improvement indeed.

After a few iterations, the CBC/HB System provides quite an accurate estimate of the amount of time that will be required for any particular problem.

---

# Using the CBC/HB System

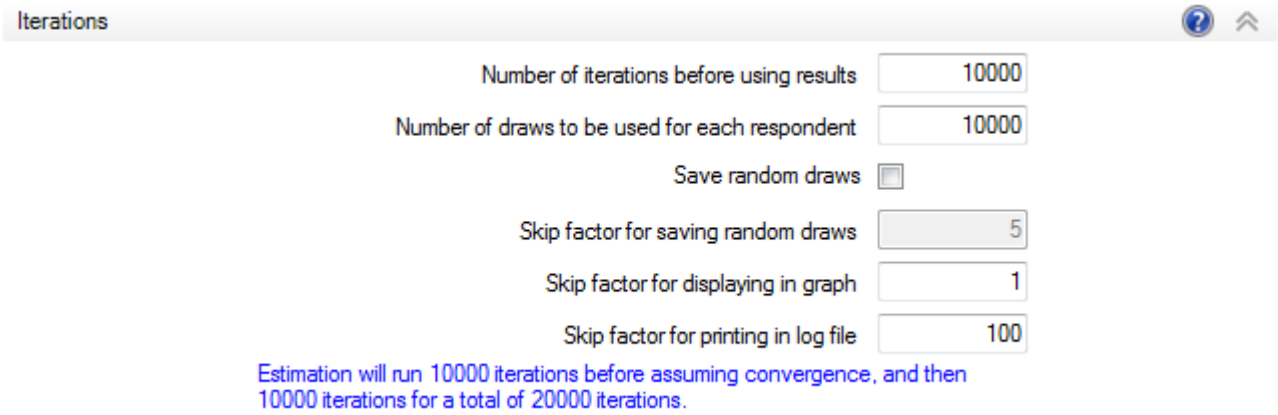
## Selecting a Data File

CBC/HB supports .CSV, .CHO, or .CHS files that are automatically produced by our Lighthouse Studio or Discover systems. You may create these files using your own tools (such as Excel), and details are provided in the documentation to do so. The first step in using the software is to browse to and select this file.

---

## Setting Parameters

The next step is to set parameter values that govern the estimation. A dialog is provided:




The numbers shown in each field are default values that you can change if you wish.

**Number of iterations before using results** is the number of iterations that will be done before convergence is assumed. The default value is 10,000, but we have seen data sets where fewer iterations were required, and others that required many more (such as with very sparse data relative to the number of parameters to estimate at the individual level). One strategy is to accept this default but to monitor the progress of the computation, and halt it earlier if convergence appears to have occurred. Information for making that judgment is provided on the screen as the computation progresses, and a history of the computation is saved in a file named studyname.log. The computation can be halted at any time and then restarted from that point.

**Number of draws to be used for each respondent** is the number of iterations used in analysis, such as for developing point estimates. If not saving draws (described next), we recommend accumulating draws across 10,000 iterations for developing the point estimates. If saving draws, you may find that using more than about 1,000 draws can lead to truly burdensome file sizes.

**Save random draws:** Check this box to save random draws to disk, in which case point estimates of respondents' betas are automatically computed by averaging each respondent's draws after iterations have finished. The default is not to save random draws, but rather to have the means and

standard deviations for each respondent's draws accumulated as iteration progresses. In that case the means and standard deviations are available immediately following iterations, with no further processing. We believe that their means and standard deviations summarize almost everything about them that is likely to be important to you. If you elect not to save the draws you will be spared computer storage problems that can occur with very large files.

**Skip factor for using draws (if saving draws)** is only applicable when saving draws to disk. The skip factor is a way of compensating for the fact that successive draws of the betas are not independent. A skip factor of k means that results will only be used for each kth iteration. Recall that only about 30% of the "new" candidates for beta are accepted in any iteration; for the other 70% of respondents, beta is the same for two successive iterations. This dependence among draws decreases the precision of inferences made from them, such as their variance. If you are saving draws to disk, because file size can become critical, it makes sense to increase the independence of the draws saved by conducting several iterations between each two for which results are saved. If 1,000 draws are to be saved for each respondent and the skip factor is 10, then 10,000 iterations will be required to save those 1,000 draws.

We do not skip any draws when draws are "not saved," since skipping draws to achieve independence among them is not a concern if we are simply collapsing them to produce a point estimate. It seems wasteful to skip draws if the user doesn't plan to separately analyze the draws file. We have advocated using the point estimates as we believe that draws offer little incremental information for the purposes of running market simulations and summarizing respondent preferences. However, if you plan to save the draws file and analyze them, we suggest using a skip factor of 10. In that case, you will want to use a more practical number of draws per person (such 1,000 rather than the default 10,000 when not saving draws), to avoid extremely large draws files.

**Skip factor for displaying in graph** controls the amount of detail that is saved in the graphical display of the history of the iterations. If using a large number of iterations (such as >50,000), graphing the iterations can require significant time and storage space. It is recommended in this case to increase the number to keep estimation running efficiently.

**Skip factor for printing in log file** controls the amount of detail that is saved in the *studyname.log* file to record the history of the iterations. Several descriptive statistics for each iteration are printed in the log file. But since there may be many thousand iterations altogether, it is doubtful that you will want to bother with recording every one of them. We suggest only recording every hundredth. In the case of a very large number of iterations, you might want to record only every thousandth.

## Advanced Settings

Most users will probably never change the defaults on the Advanced Settings screen. However, we've provided additional settings to provide more flexibility to deal with extreme types of data sets and to give advanced users greater control over estimation

**Use effects/dummy coding:** With effects coding, the last level within each attribute is "omitted" to avoid linear dependency, and is estimated as negative the sum of the other levels within the attribute. With dummy coding, the last level is also "omitted," but is constrained to zero, with the other levels estimated with respect to that level's zero parameter.

Since the release of CBC v1 in 1993, we have used effects-coding for estimation of parameters for CBC studies. Effects coding and dummy coding produce identical results (within an additive constant) for OLS or logit estimation. But, the part worths estimated using effects coding are generally easier to interpret than for dummy coding, especially for models that include interaction terms, as the main effects and interactions are orthogonal (and separately interpretable). For HB analysis (as Rich Johnson pointed out in his paper "The Joys and Sorrows of Implementing HB Methods for Conjoint Analysis,") the results can depend on the design coding procedure, when there is limited information available at the unit of analysis relative to the number of parameters to estimate. Even though we have introduced negative prior correlations in the off-diagonal elements

of the prior covariance matrix to reduce or eliminate the problem with effects coding and the "omitted" parameter for extreme data sets, there may be cases in which some advanced analysts still prefer to use dummy coding. This is a matter of personal preference rather than a choice whether one method is substantially better than the other.

**Prior degrees of freedom:** This value is the additional degrees of freedom for the prior covariance matrix (not including the # parameters to be estimated) and can be set from 2 to 100,000. The higher the value, the greater the influence of the prior variance and more data are needed to change that prior. The scaling for degrees of freedom is relative to the sample size. If you use 50 and you only have 100 subjects, then the prior will have a big impact on the results. If you have 1000 subjects, you will get about the same result if you use a prior of 5 or 50. As an example of an extreme case, with 100 respondents and a prior variance of 0.1 with prior degrees of freedom set to the number of parameters estimated plus 50, each respondent's resulting part worths will vary relatively little from the population means. We urge users to be careful when setting the prior degrees of freedom, as large values (relative to sample size) can make the prior exert considerable influence on the results.

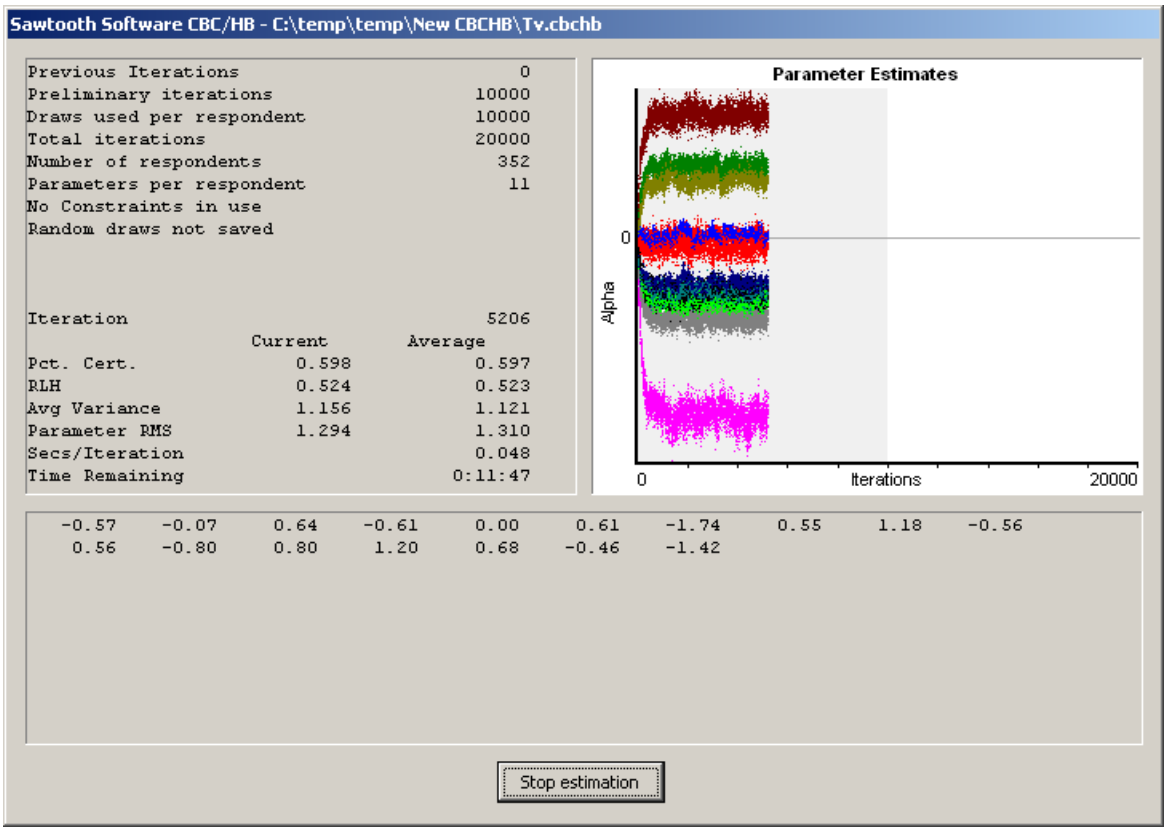
**Prior variance:** The default is 1 for the prior variance for each parameter, but users can modify this value. You can specify any value from 0.1 to 999. Increasing the prior variance tends to place more weight on fitting each individual's data and places less emphasis on "borrowing" information from the population parameters. The resulting posterior estimates are relatively insensitive to the prior variance, except 1) when there is very little information available within the unit of analysis relative to the number of estimated parameters, and 2) the prior degrees of freedom for the covariance matrix (described above) is relatively large.

**Use custom prior covariance matrix:** CBC/HB uses a prior covariance matrix that works well for standard CBC studies (see Appendix C for more information). Some advanced users may wish to specify their own prior covariance matrix (for instance, for analysis of MaxDiff data sets). Check this box and click the Edit... button to supply your own prior covariance matrix containing an NxN matrix of values, where N is equal to the number of parameters to be estimated. The user-specified prior covariance matrix overrides the default prior covariance matrix (see Appendix C) as well as the prior variance setting on the Advanced Estimation Settings tab.

**Random starting seed:** In early versions of HB, a random seed was drawn based on the computer's clock. But, users can now specify a specific seed to use (integers from 1 to 32000), so that results are repeatable. When using different random seeds, the posterior estimates will vary, but insignificantly, assuming convergence has been reached and many draws have been used.

## Monitoring the Computation

While the computation is in progress, information summarizing its current status and recent history is provided on a screen like the example below:




These are results for an actual data set, obtained relatively early in the computation. The information at the top of the screen describes the settings that were chosen before the computation was begun. This run uses the default settings of 10,000 initial iterations, followed by 10,000 further iterations during which each iteration is used, but the random draws themselves are not saved to disk.

At the time this screen print was made, the 5,206th iteration had just been completed. A graphic shows a history of the estimates of respondent parameters (elements of alpha) to this point in the computation. This graphic is useful for assessing whether convergence has been reached. The graphic is divided into two regions, a gray region at the left, which represents the initial "burn in" iterations, prior to assuming convergence, and the white region at the right in which the subsequent draws are used to create point estimates of the parameters for each respondent.

The information in the two columns in the middle-left of the screen provides a detailed summary of the status of the computation, and we shall examine those values in a moment. Also, an estimate of the time remaining is shown. 11 minutes and 47 seconds are required to complete this computation. This information is updated continuously.

At the bottom of the screen is the Stop estimation button. When this is pressed, the current iteration is finished and the current status of the computation is saved to disk for potential re-starting later. If the Stop estimation button is clicked during the second stage of estimation (the gray region of the graphic, after 10,000 iterations in this case) after we've assumed convergence and begun to use subsequent draws, the run will be halted and the current status saved, but the results from previous iterations will be deleted. When the computation is restarted all of the iterations during which results are to be used will be repeated.

We now describe the statistics displayed on the screen. There are two columns for most. In the first column is the actual value for the previous iteration. The second column contains an exponential moving average for each statistic. At each iteration the moving average is updated with the formula:

new average = .01*(new value) + .99*(old average)

The moving average is affected by all iterations of the current session, but the most recent iterations are weighted more heavily. The most recent 100 iterations have about 60% influence on the moving averages, and the most recent 500 iterations have about 99% influence. Because the values in the first column tend to jump around quite a lot, the average values are more useful.

On the left are four statistics indicating "goodness of fit" that are useful in assessing convergence. The "Pct. Cert." and "RLH" measures are derived from the likelihood of the data. We calculate the probability of each respondent choosing as he/she did on each task, by applying a logit model using current estimates of each respondent's part worths. The likelihood is the product of those probabilities, over all respondents and tasks. Because that probability is an extremely small number, we usually consider its logarithm, which we call "log likelihood."

"Pct. Cert." is short for "percent certainty" and also is known as McFadden's Rho-squared, and indicates how much better the solution is than chance, as compared to a "perfect" solution. It is equal to the difference between the final log likelihood and the log likelihood of a chance model, divided by the negative of the log likelihood for a chance model. It typically varies between zero and one, with a value of zero meaning that the model fits the data at only the chance level, and a value of one meaning perfect fit. The value of .598 for Pct. Cert. on the screen above indicates that the log likelihood is 59.8% of the way between the value that would be expected by chance and the value for a perfect fit.

RLH is short for "root likelihood," and measures the goodness of fit in a similar way. To compute RLH we simply take the nth root of the likelihood, where n is the total number of choices made by all respondents in all tasks. RLH is therefore the geometric mean of the predicted probabilities. If there were k alternatives in each choice task and we had no information about part worths, we would predict that each alternative would be chosen with probability 1/k, and the corresponding RLH would also be 1/k. RLH would be one if the fit were perfect. RLH has a value of .524 on the screen shown above. This data set has five alternatives per choice task, so the expected RLH value for a chance model would be 1/5 = .2. The actual value of .524 for this iteration would be interpreted as just better than two and a half times the chance level.

The Pct. Cert. and RLH measures convey essentially the same information, and both are good indicators of goodness of fit of the model to the data. The choice between them is a matter of personal preference.

The final two statistics, "Avg Variance" and "Parameter RMS," are also indicators of goodness of fit, though less directly so. With a logit model the scaling of the part worths depends on goodness of fit: the better the fit, the larger the estimated parameters. Thus, the absolute magnitude of the parameter estimates can be used as an indirect indicator of fit. "Avg Variance" is the average of the current estimate of the variances of part worths, across respondents. "Parameter RMS" is the root mean square of all part worth estimates, across all part worths and over all respondents.

As iterations progress, all four values (Pct. Cert., RLH, Avg Variance, and Parameter RMS) tend to increase for a while and then level off, thereafter oscillating randomly around their final values. Their failure to increase may be taken as evidence of convergence. However, there is no good way to identify convergence until long after it has occurred. For this reason we suggest planning a large number of initial iterations, such as 10,000 or more, and then examining retrospectively whether these four measures have been stable for the last several thousand iterations.

The studyname.log file contains a history of these measures, and may be inspected after the iterations have concluded, or at any time during a run by clicking Stop estimation to temporarily halt the iterative process. If values for the final few thousand iterations are larger than for the preceding few thousand, that should be considered as evidence that more iterations should be conducted before inferences are made about the parameters.

At the bottom of the screen are current estimates of average part worths. The entire "expanded" vector of part worths is displayed (up to the first 100 part worths), including the final level of each attribute that is not counted among the parameters estimated directly.

## Using Constraints

Conjoint studies frequently include product attributes for which almost everyone would be expected to prefer one level to another. However, estimated part worths sometimes turn out not to have those expected orders. This can be a problem, since part worths with the wrong slopes, or coefficients with the wrong signs, are likely to yield nonsense results and can undermine users' confidence.

CBC/HB provides the capability of enforcing constraints on orders of part worths within attributes, and on signs of linear coefficients. The same constraints are applied for all respondents, so constraints should only be used for attributes that have unambiguous a-priori preference orders, such as quality, speed, price, etc.

Evidence to date suggests that constraints can be useful when the researcher is primarily interested in the prediction of individual choices, as measured by hit rates for holdout choice tasks. However, constraints appear to be less useful, and indeed can be harmful, if the researcher is primarily interested in making aggregate predictions, such as predictions of shares of choices.

Wittink (2000) pointed out that constraints can be expected to reduce variance at the expense of increasing bias. He observed that hit rates are sensitive to both bias and variance, so trading a large amount of variance for a small amount of bias is likely to improve hit rates. He also observed that aggregate share predictions are mostly sensitive to bias since random error is likely to average out, and share predictions are therefore less likely to be improved by constraints.

In a paper available on the Sawtooth Software website (Johnson, 2000) we explored several ways of enforcing constraints among part worths in the HB context. Realizing that most CBC/HB users are probably interested in predicting individual choices as well as aggregate shares, we examined the success of each method with respect to both hit rates and share predictions. Two methods which seemed most consistently successful are referred to in that paper as "Simultaneous Tying" and "Tying After Estimation." We have implemented both of them in CBC/HB. We call the first method "Simultaneous" because it applies constraints during estimation, so the presence of the constraints affects the estimated values. The second procedure is a less formal method of simply tying offending values of saved draws from estimation done without constraints. Although it appears to work nearly as well in practice, it has less theoretical justification.

## Simultaneous Tying

This method features a change of variables between the "upper" and "lower" parts of the HB model. For the upper model, we assume that each individual has a vector of (unconstrained) part worths, with distribution:

$$\beta_i \sim \text{Normal}(\alpha, D)$$

where:

$\beta_i$ = unconstrained part worths for the ith individual
$\alpha$= means of the distribution of unconstrained part worths
$\mathbf{D}$ = variances and covariances of the distribution of unconstrained part worths

For the lower model, we assume each individual has a set of constrained part worths, $\mathbf{b}_i$ where $\mathbf{b}_i$ is obtained by recursively tying each pair of elements of $\beta_i$ that violate the specified order constraints, and the probability of the ith individual choosing the kth alternative in a particular task is

$$p_k = \exp(x_k' b_i) / \sum_j \exp(x_j' b_i)$$

With this model, we consider two sets of part worths for each respondent: unconstrained and constrained. The unconstrained part worths are assumed to be distributed normally in the population, and are used in the upper model. However, the constrained part worths are used in the lower model to evaluate likelihoods.

We speak of "recursively tying" because, if there are several levels within an attribute, tying two values to satisfy one constraint may lead to the violation of another. The algorithm cycles through the constraints repeatedly until they are all satisfied.

When constraints are in force, the estimates of population means and covariances are based on the unconstrained part worths. However, since the constrained part worths are of primary interest, we report the average of the constrained part worths on-screen, and a history of their average during iterations is available in the **studyname.meanbeta.csv** file. Also, final averages of both constrained and unconstrained part worths as well as the unconstrained population covariances are given in the **studyname.summary.txt** file.

When constraints are employed, two kinds of changes can be expected in the on-screen output:

Measures of fit (Pct. Cert. and RLH) will be ***decreased***. Constraints always decrease the goodness-of-fit for the sample in which estimation is done. This is accepted in the hope that the constrained solution will work better for predictions in new choice situations.

Measures of scale (Avg. Variance and Parameter RMS), which are based on unconstrained part worths, will be ***increased***. The constrained part worths have less variance than the unconstrained part worths, because they are produced by tying unconstrained values. Since constrained part worths are used to assess the fit of the model to the data (by computing likelihood), the constrained values take on the "correct" scaling, and the unconstrained values therefore have greater variance.

You may impose constraints on either categorical or linear attributes.

Please note that if using Simultaneous Tying, the estimates of alpha (population means) will wander significantly and not seem to reach convergence. This is expected and we do not recommend paying much attention to the estimates of alpha if using Simultaneous Tying.

## Using Covariates

Covariates are additional explanatory variables, such as usage, behavioral/attitudinal segments, demographics, etc. that can enhance the way HB leverages information from the population in estimating part worths for each individual. Rather than assuming respondents are drawn from a single, multivariate normal distribution (as has been the case in earlier versions of CBC/HB), covariates map respondents to characteristic-specific locations within the population distribution. When covariates are used that are predictive of respondent preferences, this leads to Bayesian shrinkage of part worth estimates toward locations in the population distribution that represent a larger density of respondents that share the same or similar values on the covariates. Using high quality external variables (covariates) that are predictive of respondent preferences can add new information to the model (that wasn't already available in the choice data) that improves the quality and predictive ability of the part worth estimates. One particularly sees greater discrimination between groups of respondents on the posterior part worth parameters relative to the more generic HB model where no covariates are used.

To use covariates, one first associates a .CSV file of demographics with the project on the *Data Files* tab. Respondent number must be in the first column. Covariates follow in subsequent columns. Covariates can be categorical (e.g. small_company=1, medium_company=2, large_company=3) or continuous (amount expect to pay for next automobile). Categorical covariates are coded in the Z matrix as dummy-coding, with the final level omitted as the reference zero. The covariates model is a regression-type model, where the population mean part worth parameters are estimated as a function of a matrix Z defining the respondent characteristics and a set of weights (Theta) associated with each respondent descriptor variable.

A set of weights (Theta) associated with the intercept of the population estimates of part worths as well as adjustments to the population part worth means due to characteristics of the covariates is written to the

studyname_alpha.csv file. For example, if a 3-level categorical covariate were being used, the first columns of the studyname_alpha.csv file would contain estimates for each of the part worth utilities associated with the intercept of the coded covariates matrix Z (in the case of categorical coding, the intercept would be the population mean part worth estimates associated with respondents taking on the final level of the categorical covariate). Then, the next columns would contain of set of regression weights (Theta) for the adjustments to the population estimates for each of the part worth utilities associated with respondents taking on the 1st level of the categorical covariate, followed by a set of estimates for adjustments to alpha for respondents taking on the 2nd level of the categorical covariate. The columns are clearly labeled in the .CSV file. For example, if an estimate for the level "red" for respondents taking on characteristic 2 of Variable2 was equal to 0.75, then this would indicate that respondents with characteristic 2 of Variable2 had a mean part worth utility for Red that was 0.75 utiles higher than respondents taking on characteristic 3 of Variable2 (since the final column is coded as the omitted, zero, state).

One typically examines the weights in the _alpha.csv file associated with covariates to help determine the usefulness of the covariates. Only the "used" draws should be examined. For example, if your HB run includes 10,000 burn-in iterations followed by 10,000 used iterations, then only the final 10,000 rows of the _alpha.csv file should be examined. One can examine the mean of these draws, as well as the percentage of the draws that are positive or negative. The means should have face validity (make sense from a behavioral standpoint, based on what you know about the respondent characteristics on the covariates). If the percent of draws (associated with a part worth utility) that have the same sign is 95% or greater, this is often taken as evidence that this realization of the covariate has a significant effect (90% confidence level, two-tailed test) on the part worth utility estimate. If a relatively large number of columns for a covariate have significant weights, then this gives evidence that the covariate is useful.

More detail on the usefulness of covariates in CBC/HB is provided in the white paper, "Application of Covariates within Sawtooth Software's CBC/HB Program: Theory and Practical Example" available for downloading from our Technical Papers library at www.sawtoothsoftware.com.

# How Good Are the Results?

## Background

Several articles have discussed the application of Hierarchical Bayes (HB) to the estimation of individual conjoint part worths.

- Allenby, Arora, and Ginter (1995) showed how HB could be used advantageously to introduce prior information about monotonicity constraints in individual part worths.

- In quite a different application, Allenby and Ginter (1995) showed that HB could be used to estimate individual part worths for choice data, even with relatively little data from each respondent.

- Lenk, DeSarbo, Green, and Young (1996) showed that HB could estimate individual part worths effectively even when each individual provided fewer answers than the number of parameters being estimated.

These results were impressive and suggested that HB might become the preferred method for estimation of individual part worths. In the past few years, this seems to have been the case. However, HB computation takes longer than methods such as latent class and logit, which led some to doubt about its feasibility in real-world applications in the mid-1990s. The Allenby and Ginter example used 600 respondents but estimated only 14 parameters for each. The Lenk *et al.* example used only 179 respondents, also with 14 parameters per respondent. Many commercial applications involve much larger data sets.

When Sawtooth Software introduced ICE in 1997 as a method for estimating individual part worths for choice data, we observed that computers were not yet fast enough to make HB feasible for other than small data sets. Since then computers have become faster, and it is now possible to do HB estimation within a relatively short amount of time for even relatively large data sets.

## A Close Look at CBC/HB Results

We shall now examine CBC/HB results from a study especially designed to investigate the quality of individual part worth estimates. This is the third data set examined by Huber *et al*. and we first describe it in more detail.

A total of 352 respondents answered CBC questionnaires about TV preferences (this data set is available as a "tutorial" study within the Lighthouse Studio Platform from Sawtooth Software). Each respondent answered 18 customized choice questions consisting of 5 alternatives with no "None" option. There were 6 conjoint attributes having a total of 17 levels in all. The data were coded as part worths, so 17 – 6 = 11 parameters were estimated for each respondent. The respondents were randomly divided into four groups, and those in each group answered 9 holdout tasks, each with 5 alternatives. The first and eighth tasks were identical to permit an estimate of reliability. The percentage of reliable choices for the repeated task ranged from 69% to 89%, depending on version, with an average of 81%.

The holdout tasks contained some alternatives that were very similar and sometimes identical to each another. This was done to present a challenge to conjoint simulators based on the logit model and having IIA properties.

To be absolutely sure of convergence, 100,000 iterations were done with the CBC/HB System before saving any results. We then investigated several aspects of the estimates.

## Estimation with Few Tasks

The first property examined was the ability to predict holdout choices using part worths estimated from small numbers of tasks. Six sets of part worths were estimated for each respondent, based on these numbers of tasks: all 18, 9 even-numbered, 9 odd-numbered, 6, 4, and 2. (The last three conditions used tasks distributed evenly throughout the questionnaire.) Each set of part worths was obtained by doing 1000 additional HB iterations and saving results of each 10$^{th}$ iteration. Each set of part worths was evaluated in two ways:

- Point estimates of each individual's part worths were obtained by averaging the 100 random draws, and those estimates were used in a first-choice conjoint simulator to measure hit rates.

- The random draws were also used individually in 100 separate first-choice simulations for each respondent, and accumulated over respondents to measure MAE (mean absolute error) in predicting choice shares.

With first-choice simulators, adding Gumbel-distributed random error to the summed utilities flattens share predictions in the same way that logit simulations are flattened by multiplying utilities by a number less than unity. With Gumbel error scaled to have unit standard deviation, the optimal proportion to be added was about 0.1, and this did not differ systematically depending on the number of tasks used in estimation.

Here are the resulting Hit Rate and MAE statistics for the several sets of part worths:

**Holdout Prediction with Subsets of Tasks**

| # Tasks | Hit Rate | MAE |
|---|---|---|
| 18 | 0.660 | 3.22 |
| 9 odd | 0.605 | 3.72 |
| 9 even | 0.602 | 3.52 |
| 6 | 0.556 | 3.51 |
| 4 | 0.518 | 4.23 |
| 2 | 0.446 | 5.31 |

Hit Rate and MAE for all 18 tasks are both slightly better than those reported by Huber *et al.* This may be partly due to our having achieved better convergence with the large number of iterations. Our MAEs have also been aided slightly by tuning with Gumbel error.

The important thing to notice is that performance is excellent and remains quite good as the number of choice tasks is reduced. With only 9 tasks per respondent the hit rate is about 90% as good as with all 18, and the MAE is only about 15% higher. Dropping to only 4 tasks per respondent produces a reduction in hit rate of only about 20%, and an increase in MAE of only about 30%. This does not seem to us to be a strong argument for using shorter questionnaires, because improvements from using 18 tasks instead of 9 seem worth having. But these results do give comforting evidence of robustness.

## Distribution of Replicates within Individuals

Another 10,000 iterations were computed using data from all 18 tasks, and each 10$^{th}$ replicate was saved for each respondent. Those replicates were then examined to see how the 1,000 random draws for each individual were distributed. This was investigated by first subtracting each individual's mean part worths from those of each replicate to obtain a vector of deviations. Several things were done with those 352,000 vectors of deviations.

First, the 17 x 17 matrix of pooled within-individual covariances was examined. Effects coding guarantees that the sum of variances and covariances within each attribute must be zero, so the sum of covariances for levels within each attribute must be the negative of the sum of the variances for that attribute. That naturally leads to negative covariances among the levels of each attribute. However, the covariances for all pairs of levels from different attributes were close to zero. This meant that the information about within-respondent distributions could be assessed by separate examination of each part worth element.

Next, pooled within-individual variances were examined for each level, and they did differ substantially among the 17 levels, with a ratio of approximately 4 to 1 for the maximum and minimum.

Next, skewness was also computed for each level. Skewness is zero for a symmetric distribution. For those 17 levels, 9 had slight negative skewness and 8 had slight positive skewness. All in all, the distributions were nearly symmetric.

Finally, kurtosis was computed for each level. Kurtosis indicates the relative thickness of the tails of a distribution. Values larger than 3.0 indicate thicker tails than the normal distribution. That was true for all 17 levels, with the minimum and maximum values being 3.1 and 4.1.

Therefore we can conclude that with this data set the many random draws for each individual were distributed around that individual's mean (a) independently, (b) symmetrically, and (c) with slightly thicker-than-normal tails. The regularity of these distributions suggests that little information will be lost using individuals' mean part worths rather than preserving all the individual draws. However, the individual part worths do have different variances, to which we shall again refer in a moment.

## Distributions across Individuals

Separate analyses were done for the even- and odd-numbered tasks. An additional 100,000 iterations were done initially for each and then a final 10,000 iterations for which each 10th was saved. The purpose of this analysis was to examine the estimates of covariances across individuals, rather than within individuals as described above. The covariance estimates obtained by averaging those 1000 random draws of covariance estimates were compared, as were covariance matrices obtained directly from the final point estimates of the part worths. Again, the only covariances examined were those involving levels from different attributes.

In neither case did the covariances seem very different from zero. As a check on this, we counted the number of times corresponding covariances had the same sign. For the population estimates this was only 69%, and there was only one case where the corresponding correlation had absolute values greater than .2 with similar signs in both tables. Thus, as with the within-individual covariances, there does not appear to be much structure to the distribution across individuals.

However, for both halves of the data there were large differences among the between-respondent variances, with ratios of maximum to minimum of more than 10 to 1. Also, these differences in variance were quite reliable, having a correlation between the two data sets of .87. Interestingly, the between-respondent variances were also highly correlated with the within-respondent variances, each set being correlated more than .90 with the within-respondent variances.

## Conclusions

To summarize the findings with this data set:

- The individual point estimates do an excellent job of predicting holdout concepts, and produce high hit rates using a first-choice model. Similarly, the random draws from which they are derived also do an excellent job of predicting choice shares.

- For the random draws, data for the conjoint levels appear to be distributed independently, symmetrically, and with slightly thicker-than normal tails. They differ in variances, which are approximately proportional to the across-respondent variances.

- The formal estimates of across-individual covariances do not appear to contain much information, except for the variances, among which there are strong differences.

- Similar analyses with other data sets will be required to confirm this conclusion, but it appears that nearly all the information produced by CBC/HB is captured in the point estimates of individual part worths, and little further useful information is available in the numerous random draws themselves, or in the covariances across individuals. This will be welcome news if confirmed, because it may point the way to a simpler model that works just as well with less computational effort.

## References


Allenby, G. M., Arora, N., and Ginter, J. L. (1995) "Incorporating Prior Knowledge into the Analysis of Conjoint Studies," *Journal of Marketing Research*, 32 (May) 152-62.

Allenby, G. M., Arora, N., and Ginter, J. L. (1998) "On the Heterogeneity of Demand," *Journal of Marketing Research*, 35, (August) 384-389.

Allenby, G. M. and Ginter, J. L. (1995) "Using Extremes to Design Products and Segment Markets," *Journal of Marketing Research*, 32, (November) 392-403.

Chib, S. and Greenberg, E. (1995) "Understanding the Metropolis-Hastings Algorithm," American Statistician, 49, (November) 327-335.

Gelman, A., Carlin, J. B., Stern H. S. and Rubin, D. B. (1995) "Bayesian Data Analysis," Chapman & Hall, Suffolk.

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.

Huber, J., Orme B., and Miller, R. (1999) "Dealing with Product Similarity in Conjoint Simulations," *Sawtooth Software Conference Proceedings*, Sawtooth Software, Sequim.

Huber, J., Arora, N., and Johnson, R. (1998) "Capturing Heterogeneity in Consumer Choices," *ART Forum*, American Marketing Association.

Johnson, R. M. (1997) "ICE: Individual Choice Estimation," Sawtooth Software, Sequim.

Johnson, R. M. (1999) "The Joys and Sorrows of Implementing HB Methods for Conjoint Analysis," Technical Paper available at www.sawtoothsoftware.com.

Johnson, R. M. (2000) "*Monotonicity Constraints in Conjoint Analysis with Hierarchical Bayes*," Technical Paper available at www.sawtoothsoftware.com.

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

McCullough, Richard Paul (2009) "Comparing Hierarchical Bayes and Latent Class Choice: Practical Issues for Sparse Data Sets," Sawtooth Software Conference Proceedings, Sawtooth Software, Sequim, WA.

Pinnell, Jon (1999), "Should Choice Researchers Always Use 'Pick One' Respondent Tasks?" Sawtooth Software Conference Proceedings, Sawtooth Software, Sequim, WA.

Sentis, K. & Li, L. (2000) "HB Plugging and Chugging: How Much Is Enough?" *Sawtooth Software Conference Proceedings*, Sawtooth Software, Sequim.

Sentis, K. & Li, L. (2001) "One Size Fits All or Custom Tailored: Which HB Fits Better?" *Sawtooth Software Conference Proceedings*, Sawtooth Software, Sequim.

Wittink, D. R., (2000) "Predictive Validity of Conjoint Analysis," *Sawtooth Software Conference Proceedings*, Sawtooth Software, Sequim.

# Appendix A

# Computational Procedures

## Introduction

We previously provided an intuitive explanation of the HB estimation process, and to avoid complexity it omits some details that we shall provide here.

With each iteration we re-estimate $\alpha$, a vector of means of the distribution of individuals, the covariance matrix **D** of that distribution, and a vector $ß_i$ of part worths or other parameters for each individual. Previously, we described the estimation of the betas in some detail. Here we provide details for the estimation of $\alpha$ and **D**.

## Random Draw from a Multivariate Normal Distribution:

Often in the iterative computation we must draw random vectors from multivariate normal distributions with specified means and covariances. We now describe a procedure for doing this.

Let $\alpha$ be a vector of means of the distribution and **D** be its covariance matrix. **D** can always be expressed as the product **T T'** where **T** is a square, lower-triangular matrix. This is frequently referred to as the Cholesky decomposition of **D**.

Consider a vector **u** and another vector **v** = **T u**. Suppose the elements of **u** are normal and independently distributed with means of zero and variances of unity. Since for large **n**, $1/\mathbf{n}\ \sum_n \mathbf{u\ u'}$ approaches the identity, $1/\mathbf{n}\ \sum_n \mathbf{vv'}$ approaches **D** as shown below:

$$1/\mathbf{n}\ \sum_n \mathbf{vv'} = 1/\mathbf{n}\ \sum_n \mathbf{Tu\ u'T'} = \mathbf{T}\ (1/\mathbf{n}\ \sum_n \mathbf{u\ u'})\mathbf{T'} \Rightarrow \mathbf{T\ T'} = \mathbf{D}$$

where the symbol => means "approaches."

Thus, to draw a vector from a multivariate distribution with mean $\alpha$ and covariance matrix **D**, we perform a Cholesky decomposition of **D** to get **T**, and then multiply **T** by a vector of **u** of independent normal deviates. The vector $\alpha + \mathbf{T\ u}$ is normally distributed with mean $\alpha$ and covariance matrix **D**.

## Estimation of Alpha:

If there are **n** individuals who are distributed with covariance matrix **D**, then their mean, $\alpha$, is distributed with covariance matrix $1/\mathbf{n}\ \mathbf{D}$. Using the above procedure, we draw a random vector from the distribution with mean equal to the mean of the current betas, and with covariance matrix $1/\mathbf{n}\ \mathbf{D}$.

## Estimation of D:

Let **p** be the number of parameters estimated for each of **n** individuals, and let **N** = **n** + **p**. Our prior estimate of **D** is the identity matrix **I** of order **p**. We compute a matrix **H** that combines the prior information with current estimates of $\alpha$ and $ß_i$

$$\mathbf{H} = \mathbf{pI} + \sum_n (\alpha - ß_i)\ (\alpha - ß_i)'$$

We next compute $\mathbf{H}^{-1}$ and the Cholesky decomposition

$$\mathbf{H}^{-1} = \mathbf{T\ T'}$$

Next we generate **N** vectors of independent random values with mean of zero and unit variance, $\mathbf{u}_i$, multiply each by **T**, and accumulate the products:

$$\mathbf{S} = \sum_N (\mathbf{T}\,\mathbf{u}_i)\,(\mathbf{T}\,\mathbf{u}_i)'$$

Finally, our estimate of **D** is equal to $\mathbf{S}^{-1}$.

# Appendix B

# How Constant Sum Data Are Treated in CBC/HB

## 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.

The CBC/HB 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.

Bayesian analysis provides superior results by combining data from each respondent with information from others when estimating values for that respondent. These two sources of information are combined in a way that reflects the relative strength of each. 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.

With the CBC/HB module 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 CBC/HB does 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 CBC/HB**

*In earlier versions of CBC/HB, 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.

CBC/HB 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 our hierarchical Bayes estimation procedure we compute the likelihood of each respondent's data, conditional on the current estimate of that respondent's part worths. 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:

$$P_a^7 * P_b^3 \qquad (1)$$

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:

$$P_a = \exp(U_a) / \mathrm{SUM}(\exp(U_j)) \qquad (2)$$

and

$$P_b = \exp(U_b) / \mathrm{SUM}(\exp(U_j)) \qquad (3)$$

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:

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

And, an equivalent expression is:

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

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). The HB estimation algorithm combines data from each respondent with data from others, and the relative weight given to the respondent's own data is affected by the number of probabilities multiplied together. 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 if we multiply all those probabilities together, we will in effect be giving too much weight to the respondent's own data and too little to information from other respondents.

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 CBC/HB software, but it avoids expanding the tasks and is more efficient computationally.

# Appendix C

## Specifying the Prior Covariance Matrix

In earlier versions of CBC/HB, the prior variance-covariance matrix was assumed to be the identity matrix. That is to say, prior variances for all part worths (heterogeneity values) were assumed to be unity, and all parameters were assumed to be uncorrelated. This assumption is less reasonable with effects coding of categorical attributes, where the sum of parameters for any attribute is zero, which implies negative covariances within attributes. And with dummy coding, this assumption is also less reasonable since parameters within a categorical attribute are positively correlated.

Though not rigorously correct, the assumption of zero prior covariances served well. Most data sets have enough respondents and enough tasks for each respondent that the priors have little effect on the posterior estimates for either effects coding or dummy coding.

When using the identity matrix as the prior covariance matrix, variances for the omitted levels of each attribute were overstated, and for dummy coding the variances for omitted levels (after zero-centering) were (to a lesser degree) understated, as was pointed out by Johnson (1999) in a paper available on the Sawtooth Software web site. However, for most CBC/HB data sets, this had been of little consequence, and it had not seemed worthwhile to increase the complexity of the software to deal with this situation.

However, recently we have increased the maximum number of levels permitted per attribute. We have noticed that when there are many levels, estimation of the omitted level of each attribute is less accurate, and the inaccuracy increases as the number of levels increases. This problem can be traced to the incorrect assumption of independence in the priors. Accordingly, we have changed the software so that the prior covariance matrix is specified more appropriately when either effects or dummy coding is employed. We have made several related changes.

- From the *Advanced Estimation Settings* tab, the user can specify the prior variances rather than having to assume they are equal to unity. The default value is 2.0.

- The user can also specify the prior degrees of freedom from that same tab, which must be at least equal to two more than the number of parameters being estimated. The default number is equal to the number of parameters plus 5. A larger number causes the priors to be weighted more heavily in computation of posterior parameter estimates.

- The user can specify their own prior covariance matrix from the Advanced Estimation Settings tab by clicking *Use custom prior covariance matrix*. Then, specify a text-only, space-delimited rectangular prior covariance matrix in the supplied text field, which overrides the automatic specification of the prior covariance matrix as described in this appendix.

If you are curious regarding the prior covariance matrix that has been used for your most recent run, please refer to the .PRC file, which is one of the default output files. This is a text-only file containing the prior covariance matrix used for the HB run.

**Prior Covariance Matrix under Effects Coding**

If effects coding is used, the prior covariances are automatically given appropriate negative values. Consider two attributes, a with I levels ($a_1$, $a_2$, … $a_I$) and b with J levels ($b_1$, $b_2$, … $b_J$). Actually, only I-1 plus J-1 parameters will be estimated. If there is an interaction term, denote it as $c_{ij}$, for which (I-1)*(J-1) parameters will be estimated. Denote the common variance as v.

Then the prior variances are:

$Var(a_i) = (I-1)*v/(I)$
$Var(b_j) = (J-1)*v/(J)$
$Var(c_{ij}) = (I-1)*(J-1)*v/(I*J)$

The effects between attributes are uncorrelated:

$Cov(a_i, b_j) = 0$
$Cov(a_i, c_{ij}) = 0$
$Cov(b_j, c_{ij}) = 0$

Within an attribute, the effects are correlated:

$Cov(a_i, a_k) = -v/(I)$ for i not equal to k
$Cov(b_j, b_l) = -v/(J)$ for j not equal to l
$Cov(c_{ij}, c_{kl}) = +v/(I*J)$ for i not equal to k and j not equal to l
$Cov(c_{ij}, c_{il}) = -(I-1)v/(I*J)$ for j not equal to l
$Cov(c_{ij}, c_{kj}) = -(J-1)v/(I*J)$ for i not equal to k

As a numerical example, consider two attributes having 3 and 4 levels, respectively. The prior covariance matrix for main effects is equal to the prior variance multiplied by:

| a1 | a2 | b1 | b2 | b3 | |
|---|---|---|---|---|---|
| 2/3 | -1/3 | 0 | 0 | 0 | a1 |
| -1/3 | 2/3 | 0 | 0 | 0 | a2 |
| 0 | 0 | 3/4 | -1/4 | -1/4 | b1 |
| 0 | 0 | -1/4 | 3/4 | -1/4 | b2 |
| 0 | 0 | -1/4 | -1/4 | 3/4 | b3 |

The interaction between these two attributes involves 2 * 3 = 6 variables. The prior covariance matrix is proportional to the following, with proportionality constant equal to the common variance divided by 12:

| c11 | c12 | c13 | c21 | c22 | c23 | |
|---|---|---|---|---|---|---|
| 6 | -2 | -2 | -3 | 1 | 1 | c11 |
| -2 | 6 | -2 | 1 | -3 | 1 | c12 |
| -2 | -2 | 6 | 1 | 1 | -3 | c13 |
| -3 | 1 | 1 | 6 | -2 | -2 | c21 |
| 1 | -3 | 1 | -2 | 6 | -2 | c22 |
| 1 | 1 | -3 | -2 | -2 | 6 | c23 |

**Prior Covariance Matrix under Dummy Coding**

If dummy coding is used, the prior covariances are automatically given appropriate positive values. Consider two attributes, a with I levels ($a_1, a_2, \ldots a_I$) and b with J levels ($b_1, b_2, \ldots b_J$). Actually, only I-1 plus J-1 parameters will be estimated. Denote the common variance as v.

Then the prior variances are:

$Var(a_i) = 2*v$
$Var(b_j) = 2*v$

The effects between attributes are uncorrelated:

$Cov(a_i, b_j) = 0$

Within an attribute, the effects are correlated:

$Cov(a_i, a_k)$ = v for i not equal to k
$Cov(b_j, b_l)$ = v for j not equal to l

As a numerical example, consider two attributes having 3 and 4 levels, respectively. The prior covariance matrix for main effects is equal to the prior variance multiplied by:

| a1 | a2 | b1 | b2 | b3 | |
|---|---|---|---|---|---|
| 2 | 1 | 0 | 0 | 0 | a1 |
| 1 | 2 | 0 | 0 | 0 | a2 |
| 0 | 0 | 2 | 1 | 1 | b1 |
| 0 | 0 | 1 | 2 | 1 | b2 |
| 0 | 0 | 1 | 1 | 2 | b3 |

A proper prior covariance matrix for dummy-coded models with interaction effects is not available in CBC/HB. If you specify an interaction when using with dummy coding, CBC/HB software reverts to a "default" prior covariance matrix (the identify matrix), unless a STUDYNAME.mtrx file is supplied. Dummy coding with interactions poses significant difficulties for determining an appropriate prior covariance matrix. One simple solution is to "collapse" two attributes involved in a first-order interaction into a "super attribute," coded as a single attribute in the .CHO file. Then, the super attribute may be treated as a main effect, with prior covariance structure as specified above.

# Appendix D

## Utility Constraints for Attributes Involved in Interactions

CBC/HB can constrain utilities to conform to user-specified monotonicity constraints within each individual. Whether dummy-coding or effects-coding is in place, main effect parameters may be constrained. Constraints can also be used for attributes involved in interaction terms if effects-coding is employed.

**When Both Attributes Are Categorical:**

Consider two attributes both with known preference order (level1 < level2 < level3) involved in an interaction effect. Main effects and first-order interaction effects may be estimated under effects coding in CBC/HB. 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.