CBC/HB v5

Software for Hierarchical Bayes Estimation for CBC Data

(Updated April 21, 2016)

Sawtooth Software, Inc.
Orem, UT
http://www.sawtoothsoftware.com

Bryan Orme, Editor

About Technical Support

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

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

For customer support, contact our Orem, Utah office at 801/477-4700 or email support@sawtoothsoftware.com.

Table of Contents

Foreword

Getting Started

Introduction 3
Capacity Limitations and Hardware Recommendations 5
What's New in Version 5.5? 6

Understanding the CBC/HB System

Bayesian Analysis 9
The Hierarchical Model 12
Iterative Estimation of the Parameters 13
The Metropolis Hastings Algorithm 14

Using the CBC/HB System

Opening and Creating New Projects 17
Creating Your Own Datasets in .CSV Format 19
Home Tab and Estimating Parameters 23
Monitoring the Computation 25
Restarting 28
Data Files 29
Attribute Information 31
Choice Task Filter 34
Estimation Settings 35
- Iterations 36
- Data Coding 38
- Respondent Filters 40
- Constraints 42
- Utility Constraints 44
- Miscellaneous 47

Advanced Settings 48
- Covariance Matrix 49
- Alpha Matrix 51
- Covariates 53

Using the Results 55

How Good Are the Results?

Background 56
A Close Look at CBC/HB Results 57

References

References 60

Appendices

Appendix A: File Formats 62

Appendix B: Computational Procedures 65
Appendix C: .CHO and .CHS Formats 67
Appendix D: Directly Specifying Design Codes in the .CHO or .CHS Files 73
Appendix E: Analyzing Alternative-Specific and Partial-Profile Designs 76
Appendix F: How Constant Sum Data Are Treated in CBC/HB 78
Appendix G: How CBC/HB Computes the Prior Covariance Matrix 81
Appendix H: Generating a .CHS File 84
Appendix I: Utility Constraints for Attributes Involved in Interactions 85
Appendix J: Estimation for Dual-Response "None" 88
Appendix K: Estimation for MaxDiff Experiments 90
Appendix L: Hardware Recommendations 94
Appendix M: Calibrating Part-Worths for Purchase Likelihood 96
Appendix N: Scripting in CBC/HB 98

Index **106**

1 Foreword

This Windows version of the hierarchical Bayes choice-based conjoint module fits in the best tradition of what we have come to expect from Sawtooth Software. From its inception, Sawtooth Software has developed marketing research systems that combine appropriate ways to collect information with advanced methods of analysis and presentation. Their products also have been remarkably user-friendly in guiding managers to generate useful, managerially relevant studies. One of the reasons for their success is that their programs encompass the wisdom generated by the research community generally and a very active Sawtooth users group.

In the case of hierarchical Bayes, Sawtooth Software has led in offering state-of-art benefits in a simple easy-to-use package. Hierarchical Bayes allows the marketing researcher to estimate parameters at the individual level with less than 12 choices per person. This provides an enormous value to those who leverage these partworth values for segmentation, targeting and the building of what-if simulations. Other methods that build individual choice models were tested, and indeed offered by Sawtooth Software. Latent class provided a good way to deal with heterogeneity, and Sawtooth Software's ICE (Individual Choice Estimation) generated individual estimates from that base. However, in tests, hierarchical Bayes has proven more stable and more accurate in predicting both the item chosen and the choice shares. This benefit occurs because conditioning a person's actual choice by the aggregate distribution of preferences leads to better choice predictions, and that a distribution of coefficients for each individual is both more realistic and more informative than a point estimate.

Thus, as choice based conjoint becomes increasingly popular, this CBC/HB System provides a way for the marketing research community to have it both ways-to combine the validity of a choice-based task with ease and flexibility of individual level analysis marketing researchers have long valued in traditional conjoint.

--Joel Huber, Duke University

2 Getting Started

2.1 Introduction

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

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

Quick Start Instructions:

1. Prepare the .CHO or .CHS file that contains choice data to be analyzed.

    a. From CBC/Web (Lighthouse or SSI Web System), select ***File | Export Data*** and choose the .CHO or .CHS options.

    b. From ACBC (Adaptive CBC), select ***File | Export Data*** and choose the .CHO option.

2. Start CBC/HB, by clicking ***Start | Programs | Sawtooth Software | Sawtooth Software CBC/HB***.

3. From the CBC/HB Project Wizard (or using ***File | Open***) browse to the folder containing a .CHO file, and click ***Continue***. (Wait a few moments for CBC/HB to read the file and prepare to perform analysis.)

4. To perform a default HB estimation, click ***Estimate Parameters Now…***. When complete, a file containing the individual-level part worths called studyname_utilities.CSV (easily opened with Excel) is saved to the same folder as your original data file. A text-only file named studyname.HBU is also created with the same information. If using the HB utilities in the market simulator (SMRT software), within SMRT click ***Analysis | Run Manager | Import*** and browse to the studyname.HBU.

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 Segmentation Module was offered as the first add-on to CBC 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 early 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 15 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 analysis 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 is actually faster for researchers to use than LC, 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 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.

2.2

Capacity Limitations and Hardware Recommendations

Because we anticipate that the CBC/HB System may be used to analyze data from sources other than our CBC or CBC/Web software 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. By today's standards, a PC with a 2.8 GHz processor, 2GB RAM, and 200 GBytes of storage space is very adequate to run CBC/HB for most problems.

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. See Appendix L for more information.

2.3 What's New in Version 5.5?

The latest edition of CBC/HB offers a number of improvements to the interface and also to the functionality of the software:

1. **The default Prior Variance for CBC studies has been changed from 2.0 to 1.0.** Recent work presented at the 2016 SKIM/Sawtooth Software Conference by Orme & Williams found that prior variance of 2.0 tends to overfit CBC datasets. Their findings were based on extensive analysis covering about 50 commercial data sets and using a jackknifing procedure to holdout out choice tasks for individual-level hit rate validation. None of the data sets found the optimal prior variance to be as high as 2.0. The average optimal prior variance was found to be a little bit less than 1.0, though the results depended on the characteristics of the data sets.

2. **New option to export coded design matrix.** From the Attributes tab, you can export the coded design matrix (using your current attribute settings) to a .csv file, which can be opened in a spreadsheet editor.

3. **Draw output file now a .csv file.** When saving random draws, the file format has been changed from the text-based .dra file to a .csv file format. This change allows you to directly open the file in a spreadsheet or other software for analyzing the draws.

4. **Change in the scale of RLH.** In prior versions of CBC/HB, the RLH parameter was always written to the output files on a 0-1000 scale rather than the standard 0-1 scale. To be more consistent with standard practice, the .csv output files now writes the RLH on a 0-1 scale. However, the .hbu file keeps RLH on a 0-1000 scale for compatibility with Sawtooth Software's SMRT software.

The following improvements were previously introduced in v5:

1. **Additional input file formats**. Previously, only .CHO and .CHS formats were supported. With v5, two additional input formats make it easier to supply data for HB estimation:

- CSV Layout (design and respondent data in a single file)
- CSV Layout (separate files for design and respondent data)

An optional demographics data file may be supplied in .csv format (you associate this data file with your project from the *Data Files* tab of the project window). The demographics file contains respondent IDs followed by numeric values describing demographic/segmentation variables that can be used as filters (for running HB within selected groups of respondents) or covariates (a more sophisticated model for estimating upper-level population parameters).

2. **Output files have been re-formatted to make them easier to read and manage**. Files that used to be written out with multiple rows per respondent (such as the .hbu and .dra files) are now one row per respondent. Many of the files that used to be written out as text-only files are now being written to .csv file format. They are written as one row per record, rather than sometimes divided across multiple lines (as previously could be the case). Additionally, we have provided column labels in most of the .csv files, which makes it much easier to identify the variables for analysis.

The old file names and new names are given below:

V4 File Name	V5 File Name
studyname.cov	studyname_covariances.csv

studyname.alp	studyname_alpha.csv
studyname.bet	studyname_meanbeta.csv
studyname.csv	studyname_utilities.csv
studyname.std	studyname_stddev.csv
studyname.prc	studyname_priorcovariances.csv
studyname.sum	studyname_summary.txt

3. **Ability to use covariates** (external explanatory variables, such as usage, behavioral/attitudinal segments, demographics, etc.) to 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, 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.

4. **Faster allocation-based response data handling**. Thanks to a tip by Tom Eagle of Eagle Analytics, we have implemented a mathematically identical method for processing allocation-based responses that is much faster than offered in previous CBC/HB versions (see Appendix F for details) . We've tried v5 on three small-to-moderate sized allocation-based CBC data sets we have in the office, and get between a 33% to 80% increase in speed over version 4, depending on the data set.

5. **Calibration tool for rescaling utilities to predict stated purchase likelihood (purchase likelihood model)**. Some researchers include ratings-based purchase likelihood questions (for concepts considered one-at-a-time) alongside their CBC surveys (similar to those in the ACA system). The calibration routine is used for rescaling part-worths to be used in the Purchase Likelihood simulation model offered in Sawtooth Software's market simulator. It rescales the part-worths (by multiplying them by a slope and adding an intercept) so they provide a least-squares fit to stated purchase likelihoods of product concepts. The calibration tool is available from the Tools menu of CBC/HB v5.

6. **64-bit processing supported**. If you have 64-bit processing for your computer configuration, CBC/HB v5 takes advantage of that for faster run times.

7. **Ability to specify prior alpha mean and variance**. By default, the mean prior alpha (population part-worth estimates) is 0 and prior variance was infinity in past versions of the software. Advanced users may now change those settings. However, we have seen that they have little effect on the posterior estimates. This may only be of interest for advanced users who want CBC/HB to perform as close as possible to HB runs done in other software.

8. **Ability to create projects using scripts run from the command line or an external program such as Excel.** You can set up projects, set control values, and submit runs through a scripting protocol. This is useful if automating HB runs and project management.

3 Understanding the CBC/HB System

3.1 Bayesian Analysis

This section attempts to provide an intuitive understanding of the Hierarchical Bayes method as applied to the estimation of conjoint part worths. 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:

$$p(A|B) = p(A,B) / p(B).$$

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:

$$p(A|B)\ p(B) = p(A,B).$$

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

$$p(B|A)\ p(A) = p(B,A).$$

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

$$p(B|A)\ p(A) = p(A,B).$$

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

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

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 $\mathbf{p(H_i\ )}$ of the hypothesis is known as its "prior probability," which describes our

belief about that hypothesis before we see the data.

- The conditional probability $\mathbf{p(y \mid H_i)}$ 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 $\mathbf{p(H_i \mid y)}$ 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.

3.2 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,

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

where:

β_i = a vector of part worths for the ith individual

α = 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

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

where:

p_k = the probability of an individual choosing the kth concept in a particular choice task

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 β_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 β_i of part worths for each individual, the vector α of means of the distribution of worths, and the matrix **D** of the variances and covariances of that distribution.

3.3 Iterative Estimation of the Parameters

The parameters **ß, α,** 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**, **α**, 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 **α.** We assume **α** 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 **α** is drawn from that distribution (see Appendix B for details).
- Using present estimates of the betas and **α**, draw a new estimate of **D** from the inverse Wishart distribution (see Appendix B for details).
- Using present estimates of **α** and **D**, generate new estimates of the betas. This is the most interesting part of the iteration, and we describe it in the next section. 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 (**α**, **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 **α** and **D** are saved to the hard disk. The final values of the part worths for each individual, and also of **α** and **D**, are obtained by averaging the values that have been saved.

3.4 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 β_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 β_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 β_n we draw a random vector **d** of "differences" from a distribution with mean of zero and covariance matrix proportional to **D**, and let $\beta_n = \beta_o + \mathbf{d}$.

We calculate the probability of the data (or "likelihood") given each set of part worths, β_o and β_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 p_k described in the previous section, and then multiplying all those probabilities together.

Call the resulting values p_o and p_n respectively.

We also calculate the relative density of the distribution of the betas corresponding to β_o and β_n, given current estimates of parameters α and **D** (that serve as "priors" in the Bayesian updating). Call these values d_o and d_n, respectively. The relative density of the distribution at the location of a point β is given by the formula

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

Finally we then calculate the ratio:

$$r = p_n d_n / p_o d_o$$

Recall from the discussion of Bayesian updating that the posterior probabilities are proportional to the product of the likelihoods times the priors. The probabilities p_n and p_o are the likelihoods of the data given parameter estimates β_n and β_o, respectively. The densities d_n and d_o are proportional to the probabilities of drawing those values of β_n and β_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 α and **D**, as well as information from the data.

If **r** is greater than or equal to unity, β_n has posterior probability greater than or equal to that of β_o, and we accept β_n as our next estimate of beta for that individual. If **r** is less than unity, then β_n has posterior probability less than that of β_o. In that case we use a random process to decide whether to accept β_n or retain β_o for at least one more iteration. We accept β_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 p_n will be much larger than 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 α 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 β_o to β_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. The proportionality factor is arbitrarily set at 0.1 initially. For each iteration we count the proportion of respondents for whom β_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 may be 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.

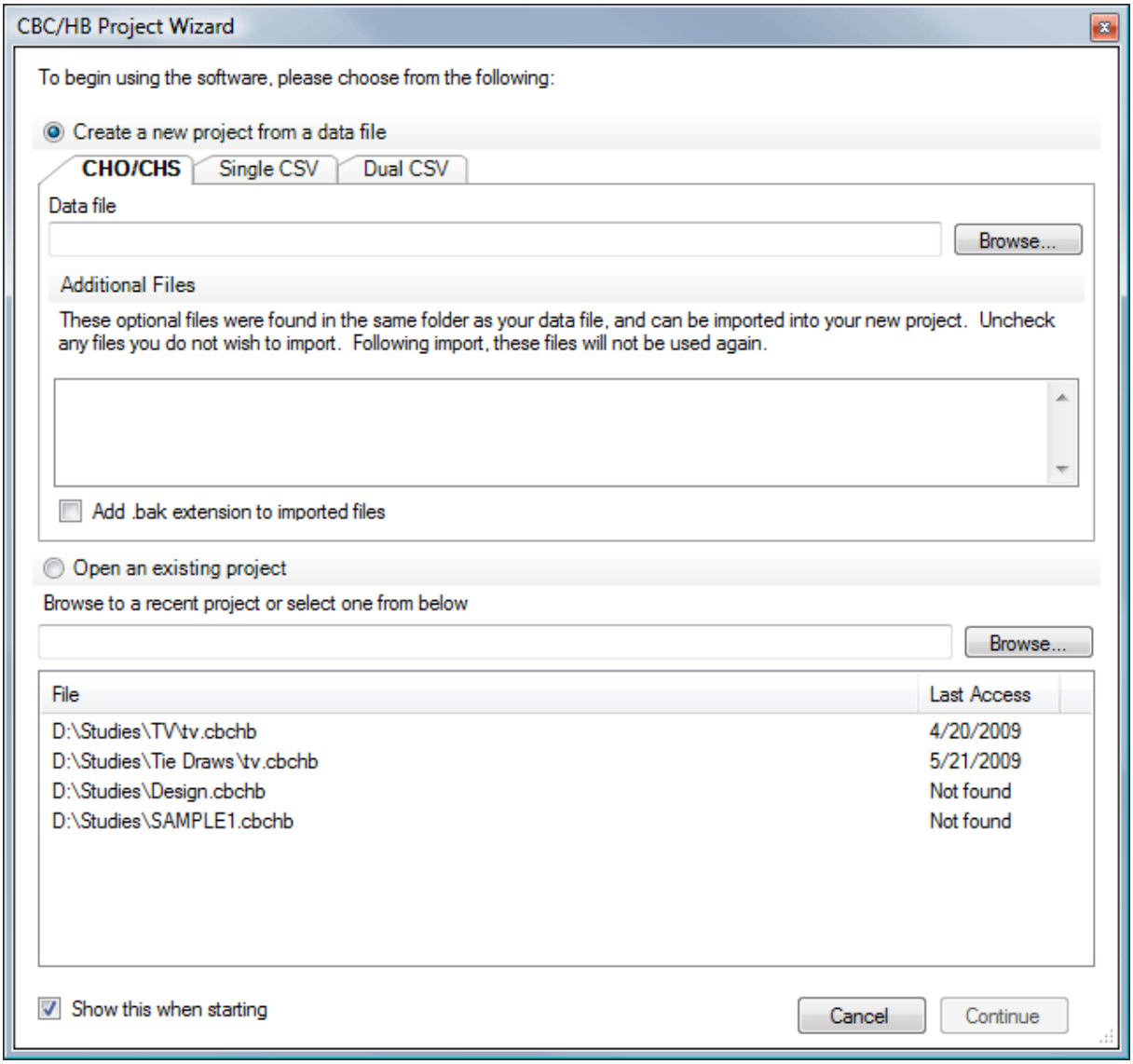
4 Using the CBC/HB System

4.1 Opening and Creating New Projects

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

Creating or Opening a Project

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



The project wizard has the following options:

Create a new project from a data file
If you collected CBC data using Sawtooth Software's CBC or ACBC systems, you should use Lighthouse Studio (formerly known as SSI Web--the platform containing those programs) to export a *studyname.cho* or *studyname.chs* file (with its accompanying labels file, called *studyname.att*). A *studyname.cho* file is a text file that contains information about the product concepts shown and the answers given for choose-one (standard discrete choice) tasks. A *studyname.chs* file is a text file that contains information about product concepts shown and answers given for allocation (constant-sum) tasks. From Lighthouse Studio (formerly known as SSI Web), click ***File | Export Data | Prepare CBC Data Files*** to generate the *studyname.cho* or *studyname.chs* file.

Note: If you supplied any earlier CBC/HB control files (.EFF, .VAL, .CON, .SUB, .QAL, .MTRX), these files are also available for import into your new CBC/HB project (you may uncheck any you do not wish to import).

Open an existing project
Click this option to open an existing CBC/HB v5 project with a .cbchb extension. Projects created with CBC/HB v4 will be updated to the new version (NOTE: updated projects can no longer be opened with v4).

Saving the Project

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

*Note: You may find it useful to use the **File | Save As** feature to create multiple project files containing different project settings, but all utilizing the same data set. That way, you can submit the multiple runs in batch mode using **Analysis | Batch Estimation...**.*

Edit | View Data File

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

4.2 Creating Your Own Datasets in .CSV Format

Most users will probably automatically prepare data files in the studyname.cho or studyname.chs format using Sawtooth Software's CBC or ACBC systems. But, other datasets created in other ways can be analyzed within the CBC/HB system. You can prepare these datasets in the .cho or .chs format. Or, you can use the simpler .CSV formats described below.

Single CSV Format
(Design and Responses within Same File)

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

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

Column 1: Caseid (i.e. respondent number)
Column 2: Task# (i.e. question number, or set number)
Column 3: Concept# (i.e. alternative number)
Next Columns: One column per attribute.
("None" concept is coded as a row of zeros.)
Final Column: Response/choice.
(With standard CBC questionnaires, respondents pick just one concept. The chosen concept is coded as "1," and the non-chosen concepts are coded "0." For allocation-based data (e.g. constant sum), you record how many chips are allocated within the response column. The response column can also accept decimal values. For Best-Worst CBC data, the best concept is coded as "1" and the worst concept is coded as "-1".)

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

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

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

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

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

Dual CSV Format
(Design and Responses in Separate Files)

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

Design File:

Column 1: Version#
Column 2: Task# (i.e. question number, or set number)
Column 3: Concept# (i.e. alternative number)
Next Columns: One column per attribute.
("None" concept is coded as a row of zeros.)

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

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

(Any "fixed tasks" (holdout tasks) that are constant across versions are coded at the top of the design

file as Version 0.)

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

Respondent Answers File:

Column 1: Caseid (i.e. respondent number)
Column 2: Version# (i.e. block number)
Next Columns: Responses (one per task).

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

Response Type:

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

None Option:

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

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

Discrete choice
a) If there is a "None" option in the design file, there should be one response per task (the chosen concept 1..n); integers only. Missing="0".
b) If there is not a "None" option in the design file:

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

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

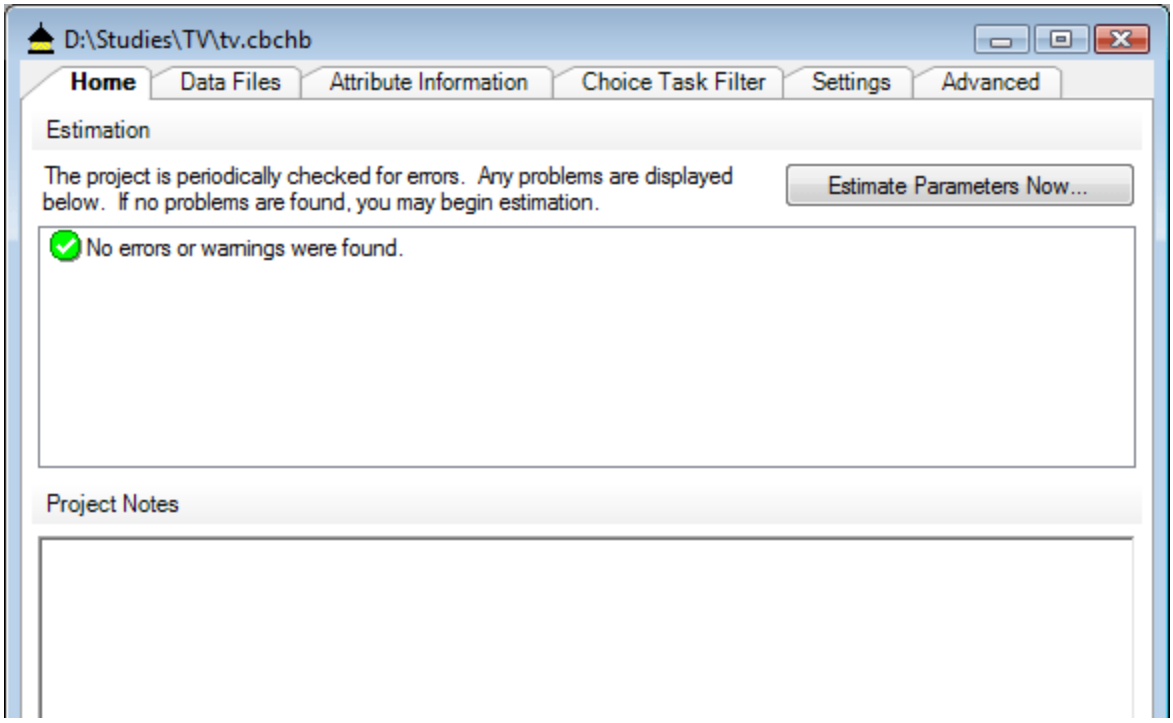
Best/Worst

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

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

4.3 Home Tab and Estimating Parameters

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



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

The *Home* tab also includes the ***Estimate Parameters Now...*** button, which is the button you click when you are ready to perform HB estimation.

Performing HB Estimation

When you click ***Estimate Parameters Now...***, two things happen. First CBC/HB makes temporary binary data files that can be read much faster than the text-based .cho or .chs data files. Preparation of data files takes a moment, and then you see a screen like the following:

```
CBC/HB Build Process (5/18/2009 12:47:44 PM)
=====================================================
Data File: D:\Studies\Tv.cho

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

```
The number of parameters to be estimated is 11.

All tasks are included in estimation

Build includes 352 respondents

Total number of choices in each response category:
Category   Number  Percent
-----------------------------------------------------
       1    1253   19.78%
       2    1275   20.12%
       3    1345   21.23%
       4    1228   19.38%
       5    1235   19.49%

There are 6336 expanded tasks in total, or an average of 18.0 tasks per respondent
```

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

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

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

The number of parameters shown on this screen is usually the number remaining after one level is deleted from the part worth levels for each attribute. If you include interactions, delete attributes, or use linear coding of attributes using the *Attribute Information* tab, the number of parameters to be estimated will vary accordingly.

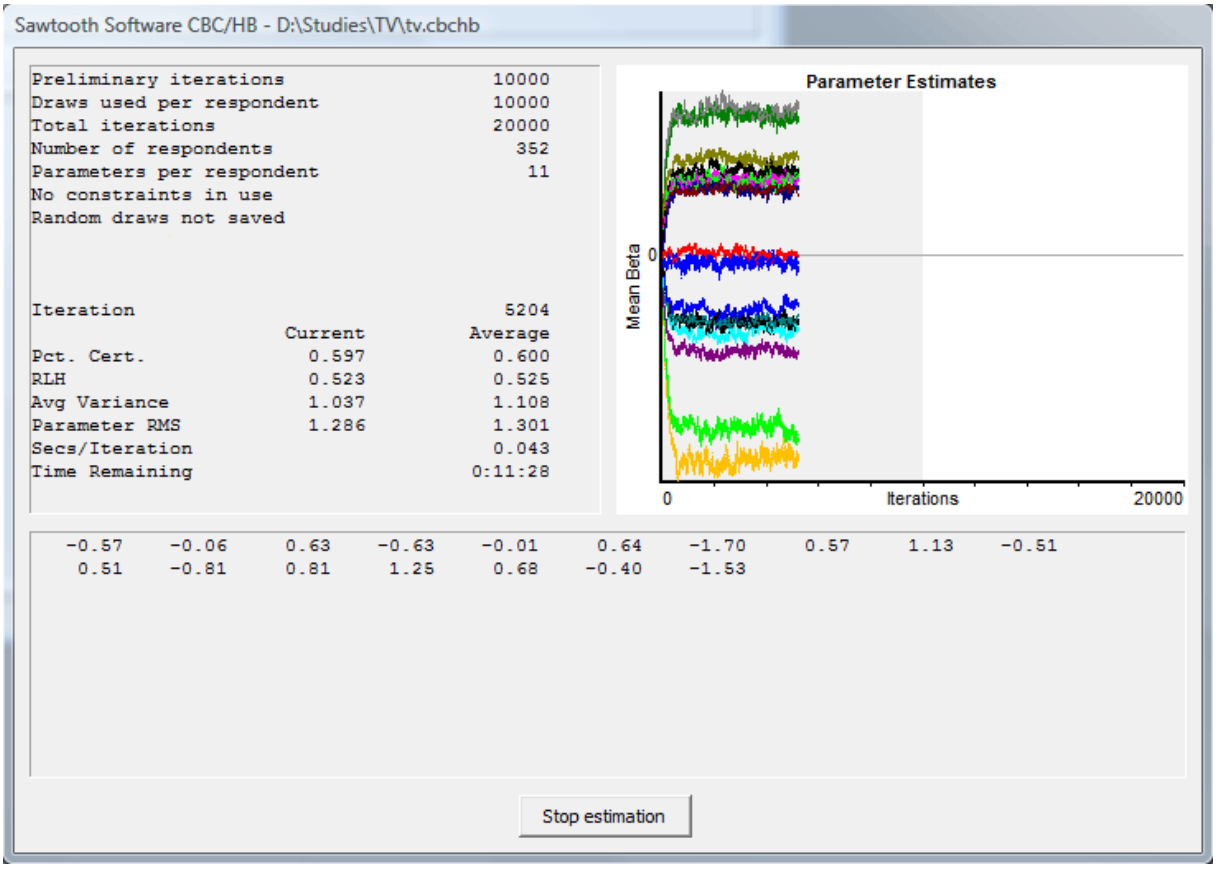
The next information is a count of the number of times respondents selected alternatives 1 through 5 of the choice tasks. This is just incidental information about your data file that may or may not be useful.

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

If you are satisfied with the way your data have been prepared, click ***Continue with estimation*** to begin the HB iterations. If not, click ***Do not estimate now*** to return to the main menu.

4.4 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 preliminary 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 5,204th iteration had just been completed. A graphic shows a history of the estimates of respondent parameters (elements of the average betas) 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 preliminary "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 28 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 selected, 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 indicates how much better the solution is than chance, as compared to a "perfect" solution. This measure was first suggested by Hauser (1978). 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 .600 for Pct. Cert. on the screen above indicates that the log likelihood is 60.0% 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 **n**th 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 .525 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 .525 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.

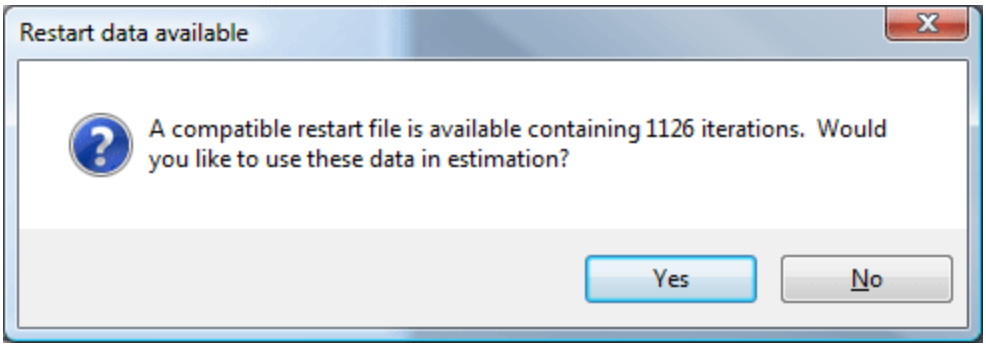
4.5 Restarting

The computation may be thought of as having three stages:

- The preliminary iterations before convergence is assumed, and during which iterations are not used for later analysis (the first 10,000 in the previous example)
- The final iterations, during which results are used for later analysis (the final 10,000 in the previous example)
- If random draws are saved to disk, the time after iterations have concluded in which averages are computed to obtain part worth estimates for each respondent and an estimate of the variances and covariances across respondents.

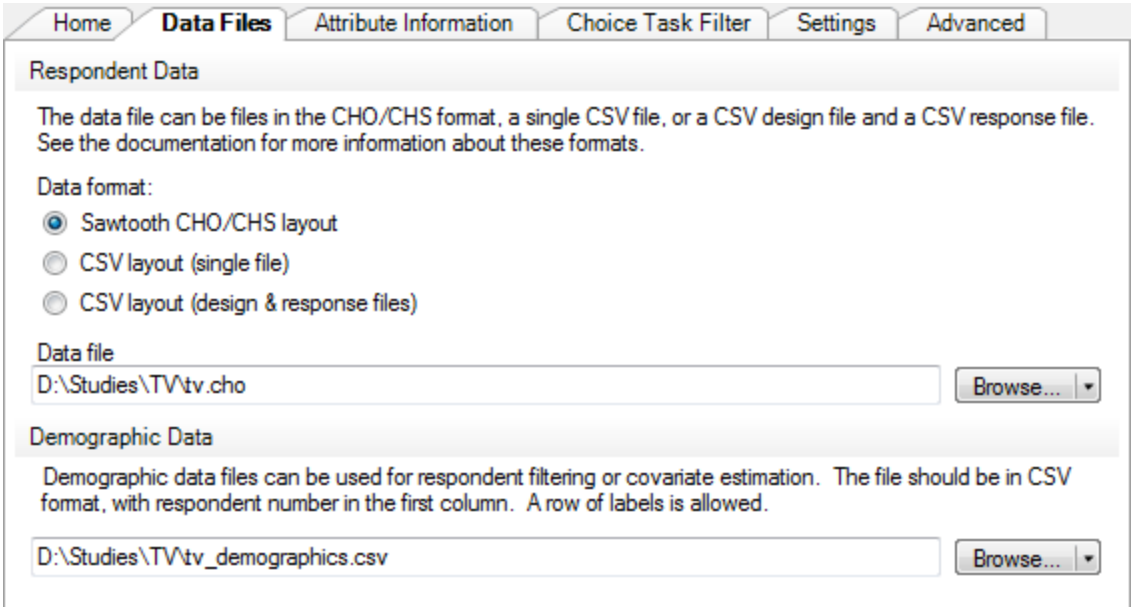
During the first two stages you may halt the computation at any time by clicking ***Stop estimation***. The current state of the computation will be saved to disk and you will later be able to restart the computation automatically from that point. If you click ***Stop estimation*** during the second stage, any results already saved or accumulated will be lost, and you will have to do more iterations to replace those results.

Later, when you select ***Estimate Parameters now...***, you will see a dialog like the one below:



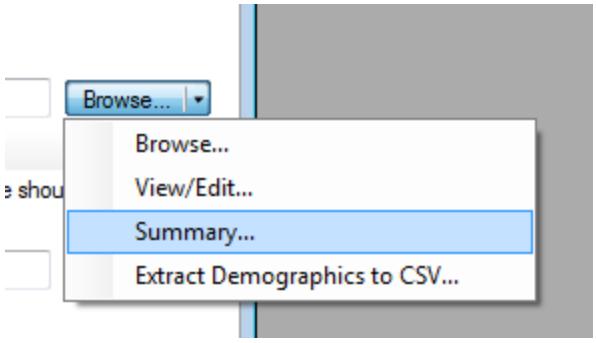
You are given the choice of restarting from that point or beginning again. Unless you have changed the specifications for the preparation of data (e.g. changes to the Attribute Information or Estimation Settings tabs) you should almost always conserve the work already done by restarting from that point.

4.6 Data Files



This tab shows you the respondent data file for your current project. A project can use data provided in a .cho or .chs file, or alternatively comma-separated values files using a single file or a combination of design and response files. You can change the data file used for the current project by browsing and selecting a new one. Note: the full path to the data file is used so that multiple projects can point to it. If you move the location of your project or data file, CBC/HB may later tell you that the data file cannot be found.

When the drop icon at the right of the **Browse** button is clicked, a menu appears.



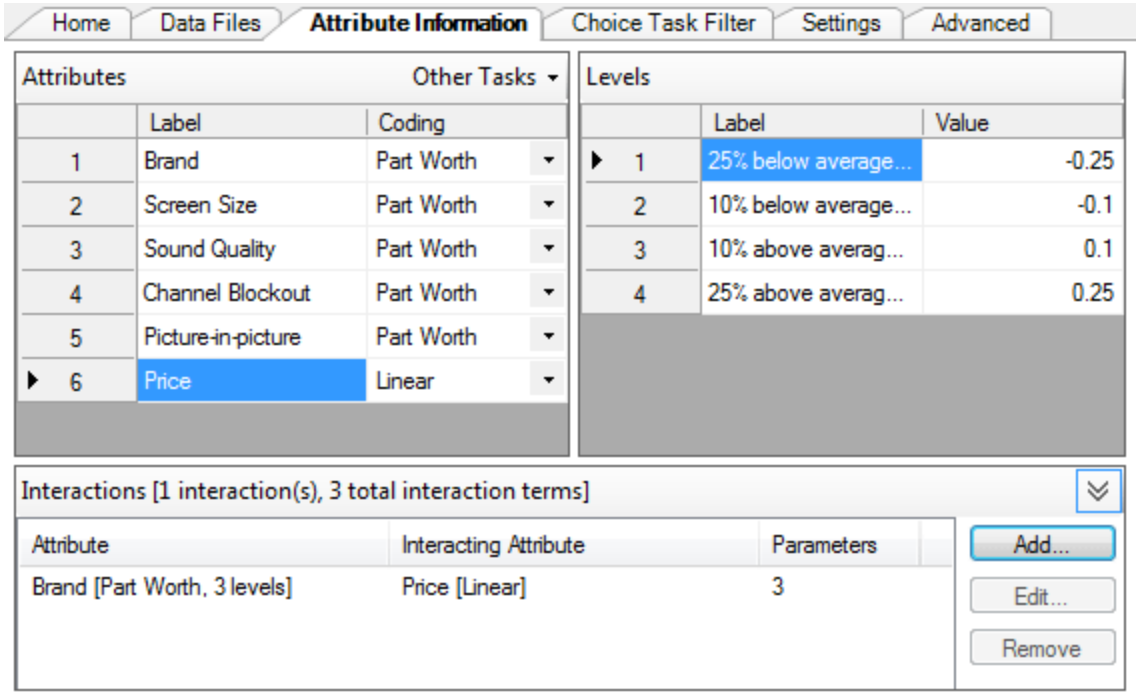
You can view a quick summary of the data in the data file by clicking ***Summary***. After a few moments, a report like the following is displayed:

```
Analysis of 'D:\Studies\TV\Tv.cho'
       Number of respondents: 352
       Total number of tasks: 6336
       Average tasks per respondent: 18
       Average concepts per task: 5
       Average attributes per concept: 6
```

You may view and also modify the data file by clicking ***View/Edit***, however with large files it may be easier to edit them with a more full-featured editor such as Notepad or Excel.

In addition to the data file, you may specify a comma-separated values file containing demographic variables. The variables in this file can be used as respondent filters or included as covariates in estimation. If you already have demographics in a .cho or .chs file (as "extra" values on line 2 for each respondent) and would like to modify them or add other data, you can select ***Extract Demographics to CSV*** from the data file **Browse** button menu. This takes the extra demographic information available in the .cho or .chs files and writes them to a .csv file.

4.7 Attribute Information



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

Generally, most projects will require few if any modifications to the defaults on this tab. Part worth (categorical effects- or dummy-coding) estimation is the standard in the industry, and HB models often do not require additional first-order interaction effects to perform admirably. For illustration, in the example above we've changed Price to be estimated as a linear function and have added an interaction between Brand and Price.

The attribute list was created when CBC/HB initially read the data (and optional) files. If you did not have an .att file containing labels for attributes and levels, default labels are shown. You may edit the labels by typing or pasting text from another application. If you have somehow changed the data file and the attribute list is no longer current, you can update it by clicking on the ***Other Tasks*** drop-down and selecting ***Build attribute information from data file***. CBC/HB will then scan the data file to update the attribute information (any labels you typed in will be discarded).

The ***Other Tasks*** drop-down can also be used to change the coding of all attributes at the same time. This is helpful, for example, when using .cho files for MaxDiff studies, where all attributes need to be changed to *User Specified*.

Attribute Coding

There are four options for attribute coding in CBC/HB:

Part worth

This is the standard approach used in the industry. The effect of each attribute level on choice is separately estimated, resulting in separate part worth utility value for each attribute level. By default, CBC/HB uses effects-coding to implement part worth estimation (such that the sum of part-worth utilities within each attribute is zero), though you can change this to dummy coding (in which the final level is set to zero) on the *Estimation Settings* tab.

Linear

With quantitative attributes such as price or speed, some researchers prefer to fit a single linear coefficient to model the effect of this attribute on choice. For example, suppose you have a price variable with 5 price levels. To estimate a linear coefficient for price, you provide a numeric value for each of the five levels to be used in the design matrix. This is done by selecting ***Linear*** from the dropdown in the ***Coding*** column. The values for each level can be typed in or pasted from another application into the ***Value*** column of the level list. CBC/HB always enforces zero-centered values within attributes during estimation. If you do not provide values that sum to zero (within each attribute) within this dialog, CBC/HB will subtract off the mean prior to running estimation to ensure that the level values are zero-centered.

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

When specifying level values for Linear coding, you should be aware that their scaling can dramatically affect the results. The range of scale values matters rather than their absolute magnitudes, as CBC/HB automatically zero-centers the coded values for linear terms, so values of 3003, 3005, and 3010 are automatically recoded as -3, -1 and +4. You should avoid using values with large ranges since proper convergence may only occur (especially with relatively sparse data sets) if the columns in the design matrix have similar variance. Part worth coded attributes have values of 1, 0 or -1 in the design matrix. For best results, we also recommend that you scale your values for linear attributes such that when zero-centered, their range is about +1 to -1. That said, we have found reasonable convergence even when values have a range of five or ten. But, ranges in the hundreds or especially thousands of units or more can result in serious convergence problems.

Important Note: If you use Linear coding and plan to use the utilities from the HB run in Sawtooth Software's SMRT program for running market simulations, you'll need to create a .VAL file prior to importing the HB run into SMRT. Simply select ***Tools | Create VAL File****. You should use the same relative values to specify products in simulations as were coded for the HB run, otherwise the beta (utility) coefficient will be applied inappropriately.*

User-specified

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

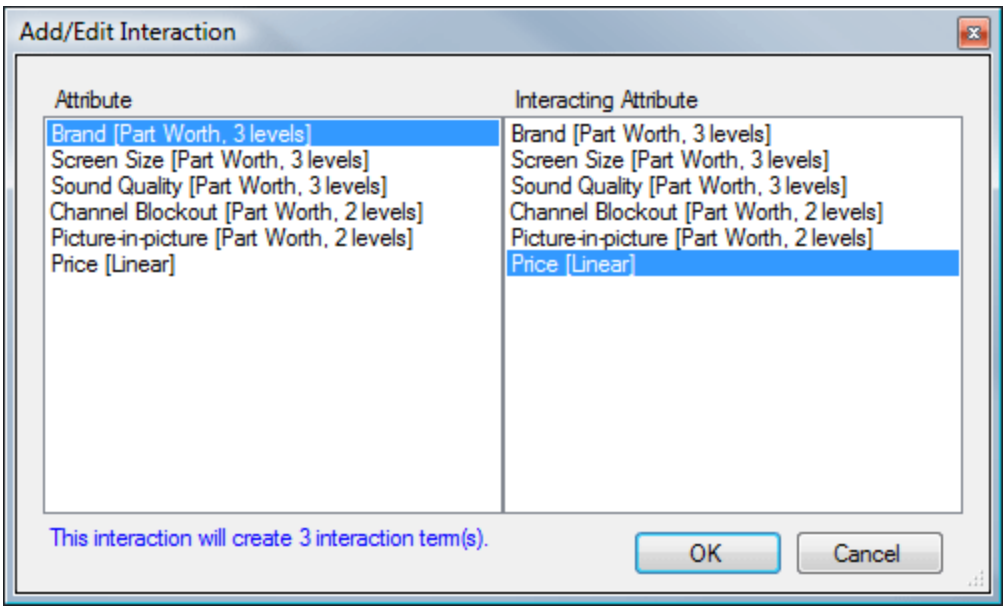
User-specified coding is also used for estimating parameters for .CHO datasets produced by our MaxDiff software. It is easiest to set all attributes to user-specified (in one step) using the ***Other Tasks*** drop-down control.

Excluded

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

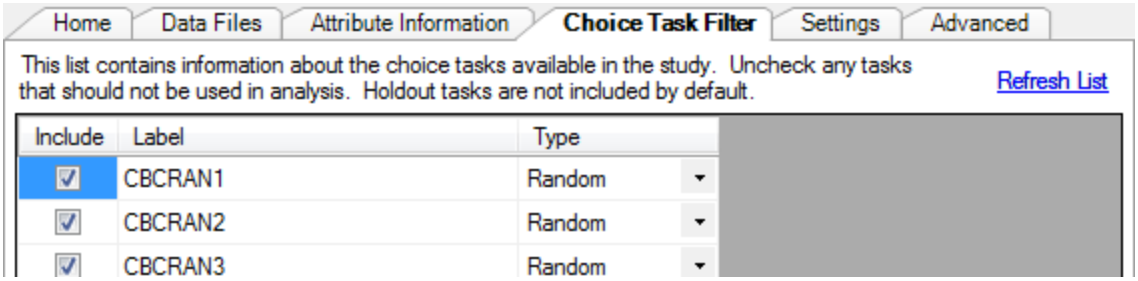
Specifying Interactions

CBC/HB can automatically include first-order interactions (interactions between two attributes). To add interaction terms to the model, click the **Add...** button within the *Interactions* panel.



Choose the two attributes that are to interact. For part-worth coded attributes, interactions add (J-1)(K-1) levels to the model, where J is the number of levels in the first attribute and K is the number of levels in the second attribute. However, after expanding the array of part worth utilities to include the "omitted" parameters, there are a total of JK (expanded) utility values representing interaction terms written to the output file.

4.8 Choice Task Filter

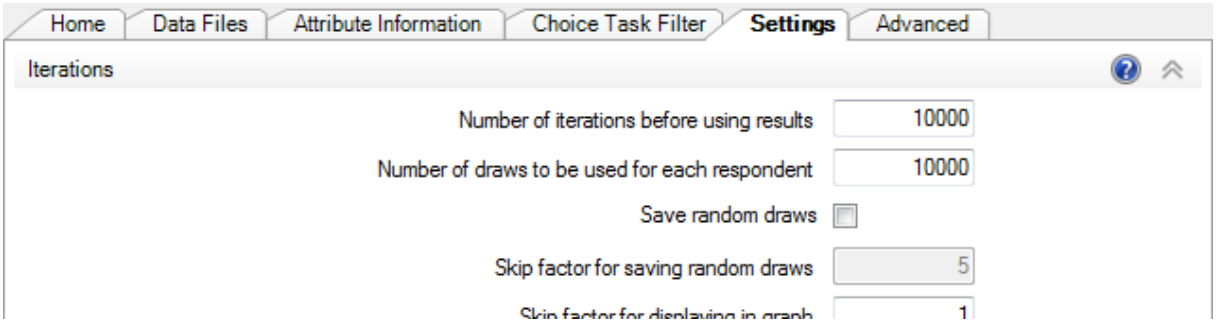


The *Choice Task Filter* tab displays a list of all available tasks in the data set. The list is automatically generated when the project is created. If you have changed the data file used by your project, this list may need updating. In that case, click the **Refresh List** link in the corner.

With Sawtooth Software's CBC data collection system, we often distinguish between "random" tasks and "fixed" tasks. Random tasks generally refer to those that are experimentally designed to be used in the estimation of attribute utilities. Fixed tasks are those (such as holdout tasks) that are held constant across all respondents and are excluded from analysis in HB. Rather, they are used for testing the internal validity of the resulting simulation model. The type of each task can be changed by selecting the appropriate option from the drop down. The task type has no effect on estimation -- it is available for your information only.

You can exclude any task by unchecking the corresponding box. Besides excluding fixed tasks, some researchers also prefer to omit the first few choice tasks from estimation, considering them as "warm-up" tasks.

4.9 Estimation Settings

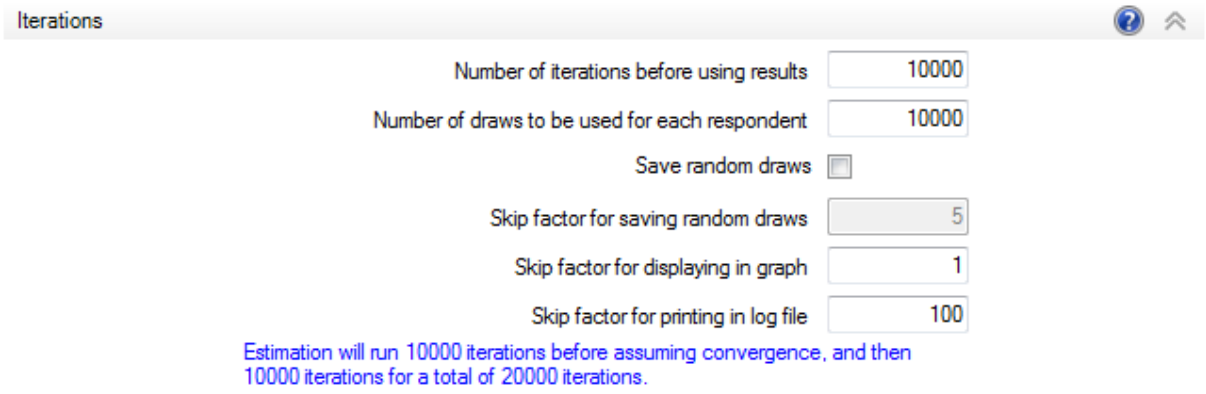


This tab displays the parameter values that govern the estimation. The settings are divided into various categories:

1. Iteration settings
2. Data coding settings
3. Respondent filters
4. Constraints
5. Miscellaneous

4.9.1 Iterations

These settings determine how much information should be generated during estimation.



Number of iterations before using results

This determines 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.

Number of draws to be used for each respondent

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 final point estimates of respondents' betas are computed by averaging each respondent's draws after iterations have finished. The default is ***not*** to save random draws (have the means and standard deviations for each respondent's draws accumulated as iteration progresses). If not saving draws 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.

However, if you choose to save draws to disk for further analysis, there is a trade-off between the benefit of statistical precision and the time required for estimation and potential difficulty of dealing with very large files. Consider the case of saving draws to disk. Suppose you were estimating 25 part worths for each of 500 respondents, a "medium-sized" problem. Each iteration would require about 50,000 bytes of hard disk storage. Saving the results for 10,000 iterations would require about 500 megabytes. Approximately the same amount of additional storage would be required for interim results, so the entire storage requirement for even a medium-sized problem could be greater than one gigabyte.

Skip factor for saving random draws (if saving draws)

This 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 available in the .HBU file, 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

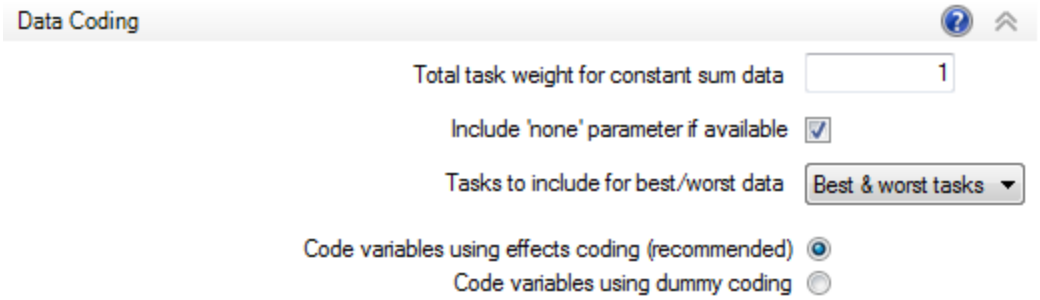
This controls the amount of detail that is saved in the graphical display of the history of the iterations. If using a large number of iterations (such as >50,000), graphing the iterations can require significant time and storage space. It is recommended in this case to increase the number to keep estimation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

Skip factor for printing in log file

This 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.

4.9.2 Data Coding

These settings describe how to treat the data during estimation.



Total task weight for constant sum data

This option is only applicable if you are using allocation-based responses rather than discrete choices in the data file. If you believe that respondents allocated ten chips independently, you should use a value of ten. If you believe that the allocation of chips within a task are entirely dependent on one another (such as if every respondent awards all chips to the same alternative) you should use a value of one. Probably the truth lies somewhere in between, and for that reason we suggest 5 as a starting value. A data file using discrete choices will always use a total task weight of 1.

Include 'none' parameter if available

We generally recommend always estimating the none parameter (but perhaps ignoring it during later simulation work). However, you can omit the "none" parameter by unchecking this box. In that case, any tasks where None has been answered are skipped. The None parameter (column) and None alternative are omitted from the design matrix.

Tasks to include for best/worst data

If you have data with Best-Worst responses, you can select which tasks to include in utility estimation: Best & worst tasks, Best tasks, or Worst tasks only. If not using Best-Worst data, this field is ignored.

Code variables using effects/dummy coding

With effects coding, the last level within each attribute is "omitted" to avoid linear dependency, and is estimated as the negative sum of the other levels within the attribute. With dummy coding, the last level is also "omitted," but is assumed to be 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.

4.9.3 Respondent Filters

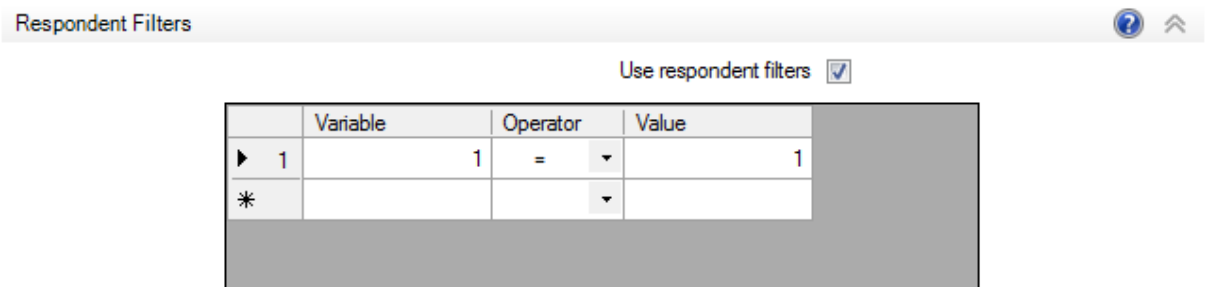
Sometimes it is desirable to filter (select) respondents for inclusion in the HB run. You can provide a demographics.csv file that includes filtering variables, including labels on the first row. The comma-separated values (.csv) file containing demographics is specified using the Choice Data File Tab. If a demographics.csv file is selected on that dialog, any demographic variables in a .cho/.chs file will be ignored.

Sawtooth choice data files (.cho or .chs) also can have demographic variables on the second line of a respondent record, though no labels are supplied. Below, we show a section from a .cho file, for respondent #8960. The second number of the header (displayed in red) states how many demographics are located on the second line (if zero, the second line is omitted. The segmentation variables that may be used for filtering respondents are on the second line, displayed in red.

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

The filtering abilities of CBC/HB are not meant to be comprehensive, but rather provide a basic method for filtering. The SMRT and Lighthouse Studio (formerly known as SSI Web) software have more sophisticated methods of filtering when exporting the data.

Next we describe the settings to create respondent filters.



Use respondent filters

This will turn respondent filtering on or off. If turned off, all respondents are included in estimation. If turned on, only respondents meeting **all** the specified criteria will be included in estimation.

Respondent Filter Window

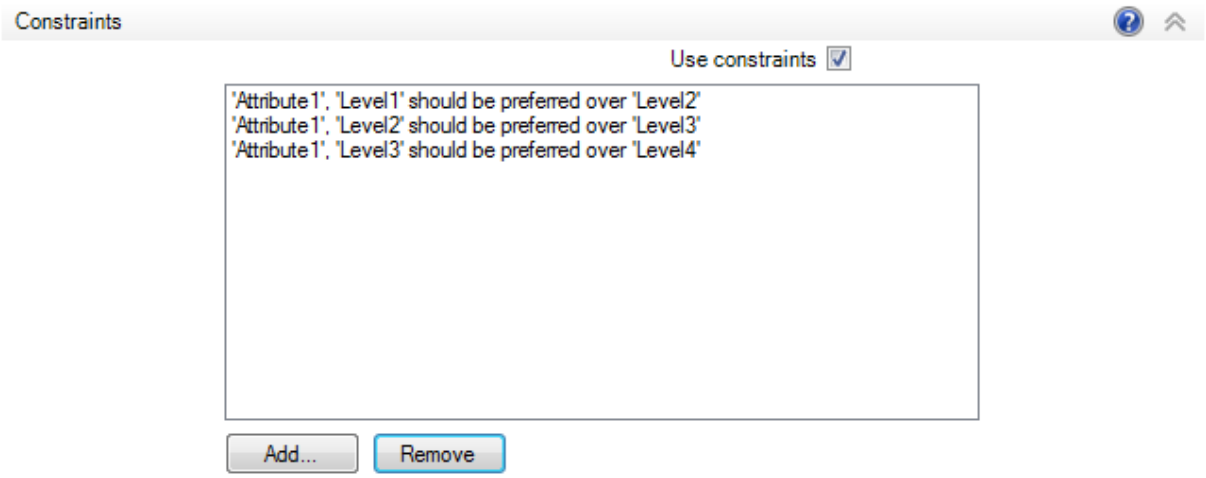
Respondent filters may be specified in the grid, one per line. In the first column, the number of the variable is specified (numbering starts at 1). The operator options are Equal, Not Equal, Greater Than, Less Than, Greater Than or Equal To, and Less Than or Equal To. The last column contains the value used to qualify each respondent. For example, to specify that all

respondents having a "4" for the first variable (as the above example does) should be included, we specify "1 = 4".

4.9.4 Constraints

Constraining utility estimates involves forcing part-worths or beta coefficients to have certain orders or signs. For example, we might want to force the utility of high prices to be lower than the utility for lower prices. More details regarding the methodology for constraints, as well as recommendations and warnings, are given in the next section of this online Help and documentation.

The Constraints area provides settings that describe how to constrain parameters during estimation.

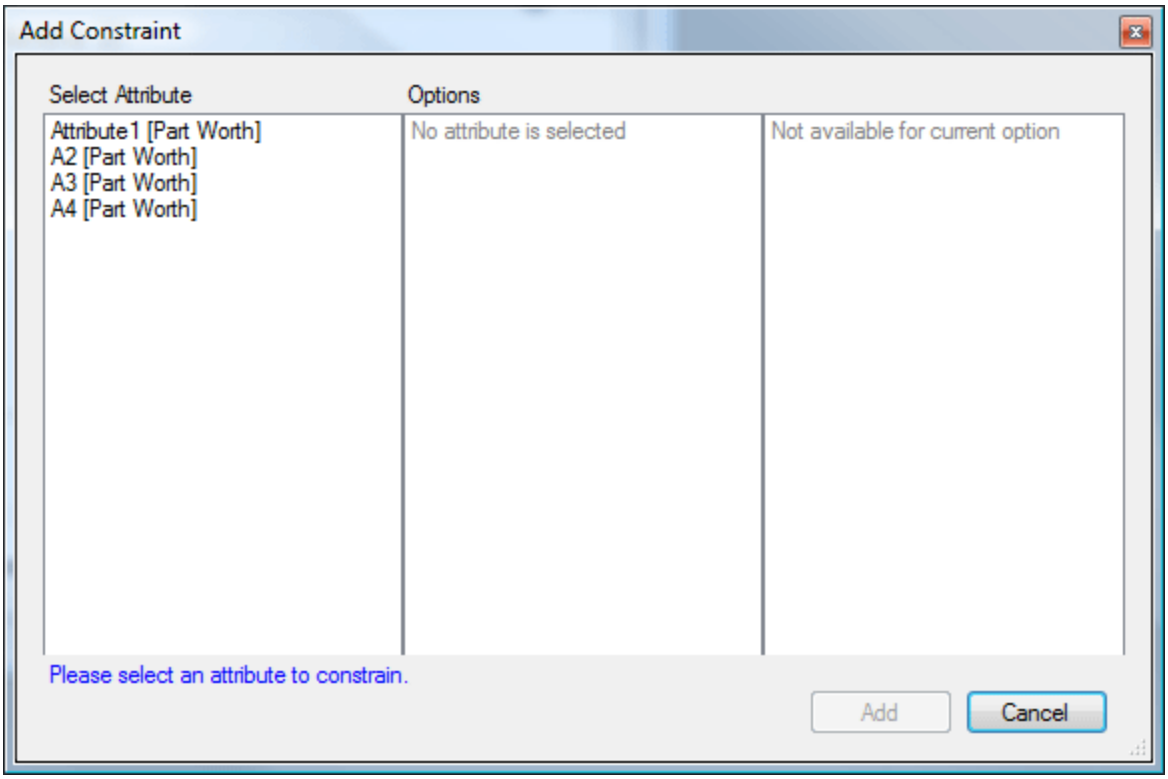


Use constraints

This will turn constraints on or off. If turned on, parameters will be constrained according to the specified constraints.

Add...

When clicked, the following dialog will appear:



Constraints may be specified by selecting an attribute to constrain, and then the desired constraint. Part worth attributes may constrain one level to be preferred over another, or the whole attribute can be specified to be constrained best-to-worst or worst-to-best (these options create multiple constraints for all levels).

Remove

When one or more constraints are selected, the remove button is enabled. When clicked, the constraints are removed permanently.

4.9.5 Utility 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, on signs of linear coefficients, and between coefficients from user-specified coding. 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 sometimes 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 Web site (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 (Tie Draws)." 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:

β_i = unconstrained part worths for the ith individual
α = means of the distribution of unconstrained part worths
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 β_i that violate the specified order constraints, and the probability of the ith individual choosing the kth alternative in a particular task is

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

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.

Tying after Estimation (Tie Draws)

To use this method, select ***Tools | Tie Draws...*** *after* you have obtained an unconstrained HB solution *and saved random draws*. The program requires the presence of **studyname.hbu**, and **studyname.dra** files. It creates an output file named **studyname_tiedraws.hbu**, which is formatted like **studyname.hbu**, and also creates a **studyname.csv** file, readable by Excel.

The TIEDRAWS utility does recursive tying of each of the random draws of part worths for each respondent and then averages them to get an estimate of that respondent's constrained part-worths. The program also provides an on-screen display indicating the percentage of unconstrained random draws for which each constraint was violated.

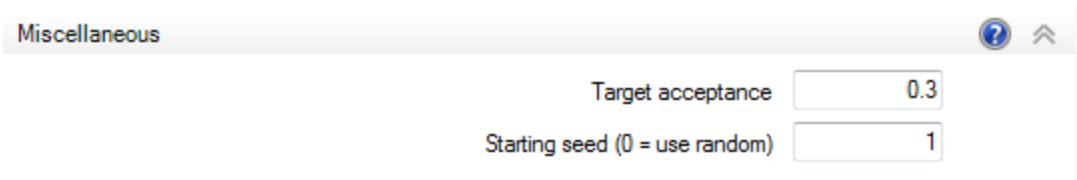
Simultaneous Tying often works a bit better than Tying After Estimation. This is to be expected, since the estimation process is informed of the constraints in Simultaneous Tying but not in Tying After Estimation. Simultaneous Tying also has the advantage that you can use it without having to save large files of random draws. However it also has a disadvantage: it requires that you specify the constraints before estimation begins, but there may be attributes for which you don't know whether to employ constraints or not. Doing separate HB estimations for many combinations of constraints could take a long time.

One way around this problem, assuming you have included "fixed" holdout choice sets in the interview, is as follows (by "fixed," we mean every respondent sees the same alternatives):

- Do unconstrained estimation, saving random draws.
- Run ***Tie Draws…*** to experiment with different constraint sets, seeing how well each predicts holdout choices and shares of choice.
- Choose the best constraint set, and re-run the estimation using Simultaneous Tying, without saving random draws.

4.9.6 Miscellaneous

These settings relate to miscellaneous aspects of estimation.



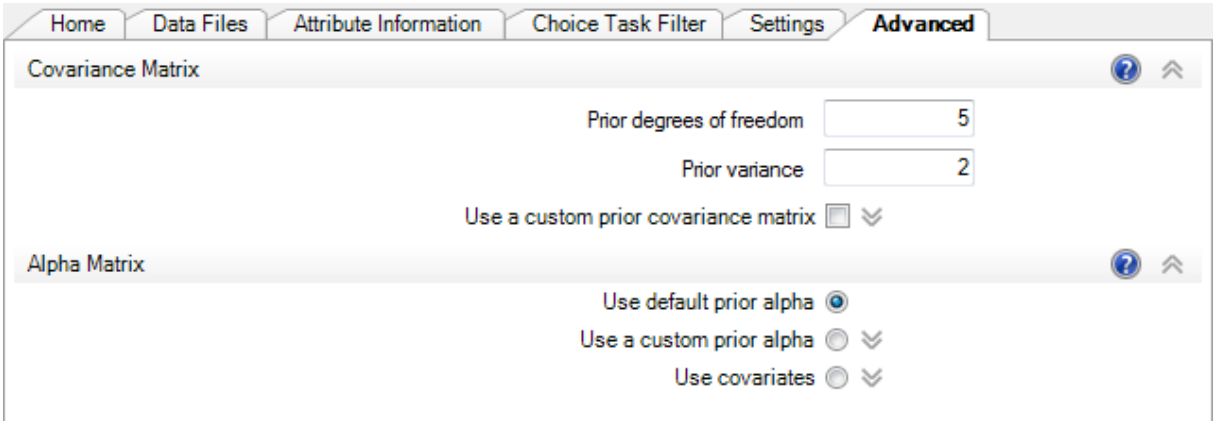
Target acceptance

This is used to set the target rate at which new draws of beta are accepted (the jump size is dynamically adjusted to achieve the target rate of acceptance). The default value of 0.3 indicates that on average 30% of new draws should be accepted. The target acceptance has a range between 0.01 and 0.99. Reports in the literature suggest that convergence will occur more rapidly if the acceptance rate is around 30%.

Starting seed

The starting seed is a value used to seed the random number generator used to draw multivariate normals during estimation. If a non-zero seed is specified, the results are repeatable for that seed. If the seed is zero, the system will use the computer clock to randomly choose a seed between 1 and 10000. The chosen seed will be written to the estimation log. When using different random seeds, the posterior estimates will vary, but insignificantly, assuming convergence has been reached and many draws have been used.

4.10 Advanced Settings

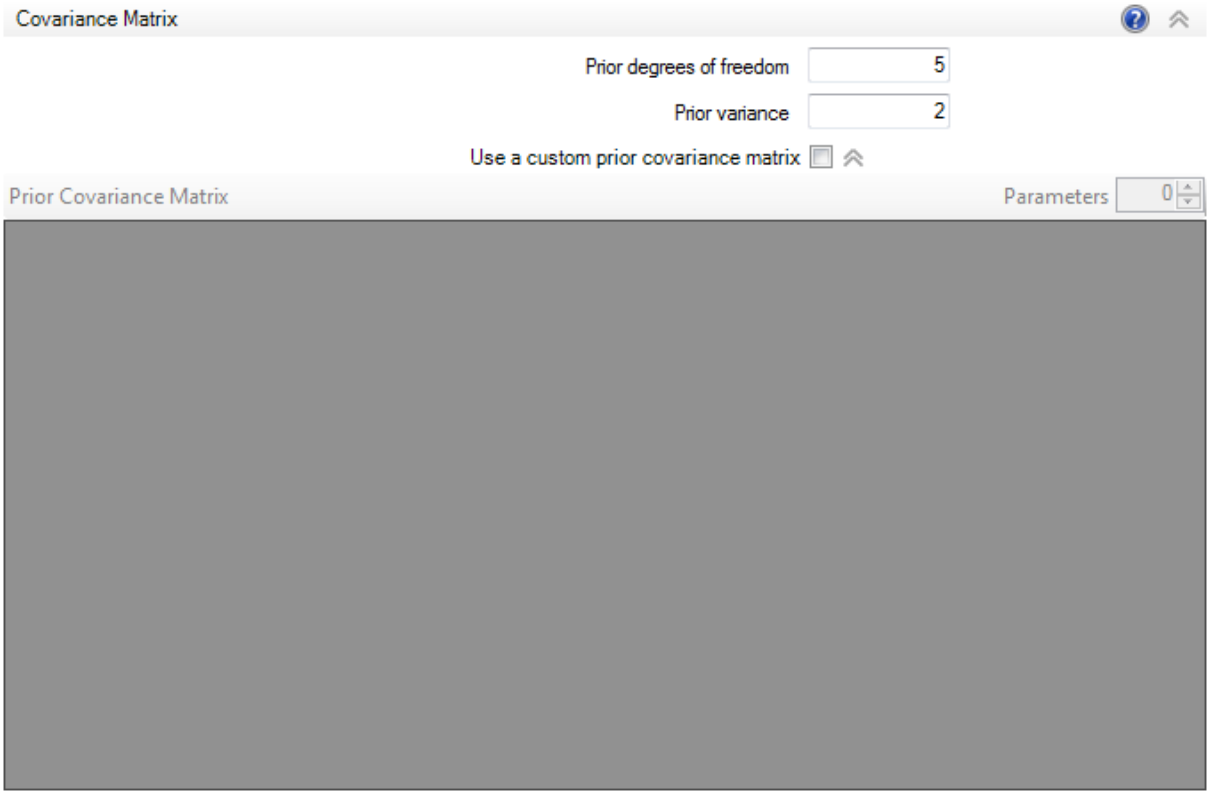


Most users will probably never need to change the advanced settings. However, some additional settings are available to provide more flexibility to deal with extreme types of data sets and to give advanced users greater control over estimation. The advanced settings are divided into two categories:

1. Covariance matrix settings
2. Alpha matrix settings

4.10.1 Covariance Matrix

This topic covers the covariance matrix settings in the software. See Appendix G for more information about the covariance matrix.



Prior degrees of freedom

This value is the additional degrees of freedom for the prior covariance matrix (not including the number of parameters to be estimated), and can be set from 2 to 100000. 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 0.05 to 100. 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.

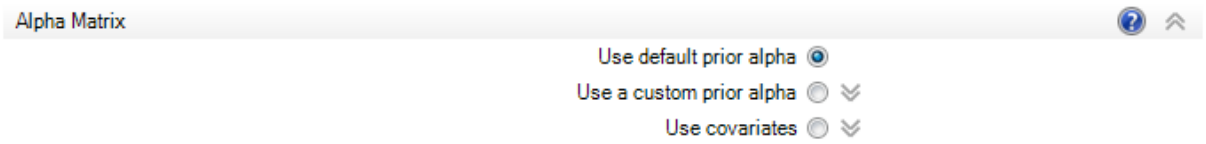
Use custom prior covariance matrix

CBC/HB uses a prior covariance matrix that works well for standard CBC studies. 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  icon to make the prior covariance matrix visible. The number of parameters can be adjusted by using the up and down arrows on the Parameters field, or you may type a number in the field. The number of parameters needs to be the same as the number of parameters to be estimated. Values for the matrix may be typed in or pasted from another application such as Excel. The user-specified prior covariance matrix overrides the default prior covariance matrix (see Appendix G) as well as the prior variance setting.

4.10.2 Alpha Matrix

Most users will not change the default alpha matrix. Advanced users may specify new values for alpha using this dialog.

Covariates are a new feature with v5 of CBC/HB. 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.



Use default prior alpha

Selecting this option will use a default alpha matrix with prior means of zero and prior variances of 100. No demographic variables will be used as covariates.

Use a custom prior alpha

Users can specify their own prior means and variances to be used in the alpha matrix. The means and variances are expanded by clicking the icon.

The number of parameters for the means and variances can be adjusted by using the up and down arrows of the Parameters field, or you may type a number in the field. The number of parameters needs to be the same as the number of parameters to be estimated (k-1 levels per attribute, prior to utility expansion). Values for the matrix may be typed or pasted from another application such as Excel.

Use Covariates

CBC/HB v5 allows demographic variables to be used as covariates during estimation. The available covariates can be expanded by clicking the icon.

Clicking *Refresh List* will scan the demographic file (see Data Files to specify a demographics file) and populate the list of available covariates. Individual variables can be selected for use by clicking the 'Include' checkbox. The labels provided are for the benefit of the user and not used in estimation. Each covariate can be either *Categorical* or *Continuous*.

Categorical covariates such as gender or region are denoted by distinct values (1, 2, etc.) in the demographic file. If a covariate is categorical, the number of categories is requested (i.e. the number of genders would be two: for male and female). The number of categories is necessary since they are expanded using dummy coding for estimation.

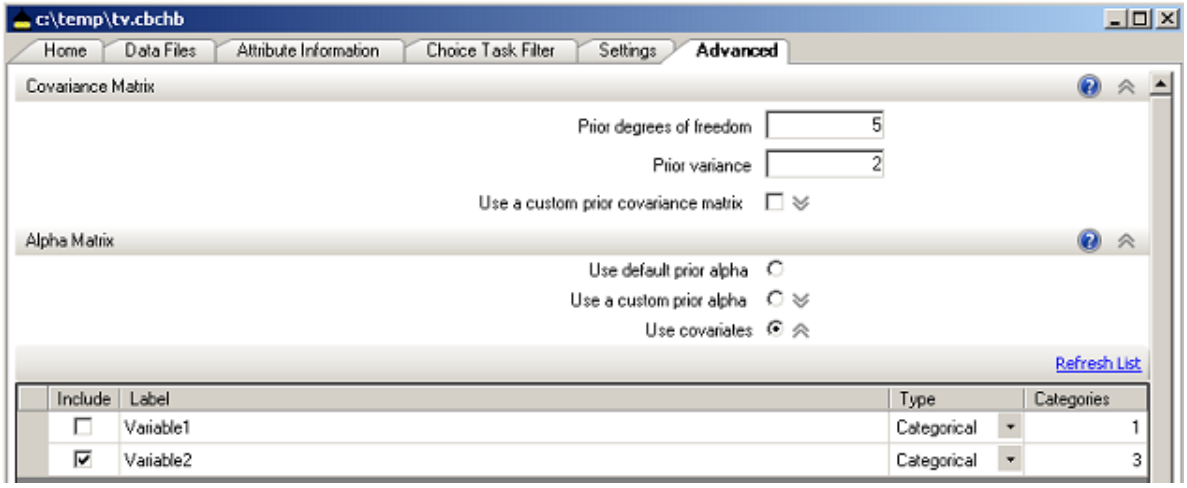
Continuous covariates are not expanded and used as-is during estimation. We recommend zero-centering continuous covariates for ease of interpreting the output.

4.10.3 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, 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). If they are categorical, they must be recorded as consecutive integers starting with 1. 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.

In the example below, there are two covariates in the demographics.csv file, and the second covariate (a categorical variable with three values: 1, 2, or 3) is being used in the run.



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.

Note: the weights associated with a covariate within the _alpha.csv file reflect the *change* in the base (intercept) part-worth value when respondents take on the characteristic as described by that covariate. They cannot be interpreted as the part-worth utility estimate alone. Since the application of covariates follows a regression model, the population estimate for respondents taking on specific covariate realizations is equal to the part-worth utility associated with the intercept plus the adjustments (weights) of the covariates associated with the respondents.

If you make changes to the demographics.csv file, you may need to click the ***Refresh List*** link to ask CBC/HB to reread the demographics.csv file for use in the run.

4.11 Using the Results

At the end of the computation, several files are available containing the results:

Files named **studyname.hbu** and **studyname_utilities.csv** contain final part worth estimates for each respondent. These are the averages of hundreds or thousands of draws either saved during the final stage of the iterations or accumulated on the fly, if those were not saved. The **studyname_utilities.csv** file is a comma-separated text-only file that may be directly opened with Excel™. The format of the **studyname.hbu** file is described in Appendix A. Either of these files, perhaps after minor modification, may be used in a conjoint simulator. If using Sawtooth Software's market simulator, the **studyname.hbu** file may directly be imported into your simulation study. If using SMRT, you select ***Analysis | Run Manager | Import*** within SMRT's menu system.

If you have not saved random draws to disk, a file named **studyname_stddev.csv** contains within-respondent standard deviations for each respondent's part worths. Its format is similar to that of **studyname.hbu**.

A file named **studyname_summary.txt** contains summaries of the data. Its contents are slightly different, depending on whether or not you have used constraints. If there were no constraints, then this file contains the estimated average part worths for the population of respondents, as well as the variance-covariance matrix estimated for the population distribution. If there were constraints, then it includes those same values for the unconstrained part worths, as well as average values for the constrained part worths. (See the section on Constraints for further clarification.) Like the final part worth estimates, this matrix is obtained by averaging results saved during the final stage of the iterations.

A file named **studyname_alpha.csv** contains successive random draws of the mean of the population distribution. The point estimate of the average part worths for the population of respondents as reported in the **studyname_summary.txt** file is obtained by averaging these draws (for the range of used draws). One way to determine when convergence occurs is to inspect this file to see whether there are systematic trends in any values.

If constraints were used, a file named **studyname_meanbeta.csv** contains successive estimates of the mean of the constrained betas. Like the **studyname_alpha.csv** file, it can be inspected to determine when convergence occurs.

A file named **studyname_covariances.csv** contains successive random draws of the variances and covariances of the population distribution. The final point estimate of the variances and covariances is obtained by averaging the draws during the final stage of the iterations. Only the elements on and above the diagonal of the covariance matrix are saved in this file.

Finally, if you have saved random draws, a **studyname_draws.csv** file is available with estimates of each respondent's part worths for all the iterations from which those values were saved. This may be a very large file, since it contains potentially thousands of estimates of part worths for each respondent. The data are arranged in order by respondent. This file can provide the raw data for analyses you may undertake using statistical software packages.

5 How Good Are the Results?

5.1 Background

Early 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. Fortunately, since then, computers have become faster, and it is now possible to do HB estimation within a reasonable amount of time for even relatively large data sets.

5.2 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 a data set examined by Huber *et al*. (1998) 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 SMRT 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 other. 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 10th 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 10th 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.

6 References

6.1 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.

Cohen, Steve (2003), "Maximum Difference Scaling: Improved Measures of Importance and Preference for Segmentation," 2003 Sawtooth Software Conference Proceedings, 61-74.

Feller, William, "An Introduction to Probability Theory and Its Applications," Vol. 1, Second edition, Wiley 1957, page 104.

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.

Orme, Bryan (2005), "Accuracy of HB Estimation in MaxDiff Experiments," Technical paper available at http://www.sawtoothsoftware.com.

Pinnell, Jon (1999), "Should Choice Researchers Always Use 'Pick One' Respondent Tasks?"

Sawtooth Software Conference Proceedings, Sawtooth Software, Sequim, WA.

Sa Lucas, Luis (2004), "Scale Development with MaxDiffs: A Case Study," 2004 Sawtooth Software Conference Proceedings, 69-82.

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.

7 Appendices

7.1 Appendix A: File Formats

Input Files:

The **studyname.cho** file contains information about choices made by each respondent, as well as the description of each choice task.

The **studyname.chs** file contains information about allocations (constant sum, or chip allocation) made by each respondent, as well as the description of each choice task.

Output Files:

The **studyname_alpha.csv** file contains the estimated population mean for part worths. There is one row for each recorded iteration. The average part worths are "expanded" to include the final levels of each categorical attribute which were temporarily deleted during estimation.

The **studyname_meanbeta.csv** file is only created if you have specified constraints. There is one row for each recorded iteration. The average part worths are "expanded" to include the final levels of each categorical attribute which were temporarily deleted during estimation.

The **studyname_covariances.csv** file contains the estimated variance-covariance matrix for the distribution of part worths across respondents, for each saved iteration. Only the elements on-or-above the diagonal are saved.

The **studyname_utilities.csv** file contains point-estimates of part worths or other parameters for each respondent, along with variable labels on the first row.

The **studyname_draws.csv** file contains estimated parameters for each individual saved from each iteration, if you specified that it should be created. Values in the **studyname.hbu** file are obtained by averaging the draws found in the **studyname_draws.csv** file. This file is formatted like the **studyname_utilities.csv** file except that it also includes a column for the draw. It can be very large because it contains not just one record per respondent, but as many as you decided to save — perhaps thousands.

The **studyname.hbu** file contains point-estimates of part worths or other parameters for each respondent. (Detailed file format shown further below.)

The **studyname_priorcovariances.csv** file contains the prior covariance matrix used in the computation.

The **studyname_stddev.csv** file is only created if you elect not to save random draws. In that case, it contains the within-respondent standard deviations among random draws. There is one record for each respondent, consisting of respondent number, followed by the standard deviation for each parameter estimated for that respondent.

The **studyname_summary.txt** file contains final estimates of the unconstrained population means, the constrained sample means if constraints were used, and the unconstrained variance-covariance matrix for the distribution of part worths across respondents. Those sections of the file are labeled alphabetically, and are formatted so as to be read by humans rather than computers. The estimated

population means and constrained sample means are both expanded to include the final levels for each categorical attribute which were temporarily deleted during estimation. The variance-covariance matrix is also expanded to include those temporarily deleted rows and columns.

Studyname.HBU File Format

The **studyname.hbu** file contains the main result of a CBC/HB calculation: estimates of part worths or other parameters for each respondent. Since this file always has the same name, it is important that you rename it before doing further analyses with the same study name to avoid over-writing it. The file contains a header section that describes which parameters have been estimated, followed by a record for each respondent.

Following is an example of such a header:

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

The first line contains the number of attributes, whether "None" is included (1 if yes, 0 if no), the total number of parameters estimated for each individual, and the numbers 1 and 1. (These last two values are just to conform with the file format for the LCLASS module, which uses the same header format.)

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

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

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

The record for each respondent starts with a line that contains:

- Respondent number
- An average value of RLH (on a 0-1000 scale for compatibility with other modules), obtained by averaging the root-likelihood values for each random draw of his/her parameter estimates
- A value of zero (for compatibility with other modules)

- The total number of parameter estimates per respondent
- A negative one if a "None" option is included, or a zero if not (for compatibility with other modules)

It is followed by the parameter values for that respondent, in the same order as the labels in the header.

7.2 Appendix B: Computational Procedures

Introduction

We previously attempted to provide an intuitive understanding of the HB estimation process, and to avoid complexity we omitted some details that we shall provide here.

With each iteration we re-estimate α, 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. We previously described the estimation of the betas in some detail. Here we provide details for the estimation of α 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 α 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**, $1/n \sum_n u\,u'$ approaches the identity, $1/n \sum_n vv'$ approaches **D** as shown below:

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

where the symbol => means "approaches."

Thus, to draw a vector from a multivariate distribution with mean α 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 $\alpha + T\,u$ is normally distributed with mean α and covariance matrix **D**.

Estimation of Alpha:

If there are **n** individuals who are distributed with covariance matrix **D**, then their mean, α, is distributed with covariance matrix $1/n\ D$. 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 $1/n\ D$.

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 α and $ß_i$

$$H = pI + \sum_n (\alpha - ß_i)\,(\alpha - ß_i)'$$

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

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

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:

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

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

7.3 Appendix C: .CHO and .CHS Formats

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

Note: Many users will find the .CSV format data files easier to work with than the file formats described below. You can quickly convert a .CHO file to .CSV using the ***Tools + Convert .cho to .csv*** option.

.CHO File Layout

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

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

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

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

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

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

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

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

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

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

Line 2:

4	**2**
Interview Duration	Segmentation Variable

These "extra" variables are largely a carryover from very early versions of CBC software, and are of no use in the CBC/HB system. If you specify that there are no "extra" variables on line 1, you can omit this line of data.

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

Line 3:

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

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

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

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

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

Level Displayed for Each Attribute in First Concept

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

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

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

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

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

1 3 lbs
2 5 lbs

3 7 lbs

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

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

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

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

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

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

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

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

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

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

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

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

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

.CHS File Layout

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

.CHS data file:

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

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

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

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

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

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

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

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

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

These "extra" variables are largely a carryover from very early versions of CBC software, and are of no use in the CBC/HB system. If you specify that there are no "extra" variables on line 1, you can omit this line of data.

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

Line 3:	**3**
	Number of Concepts first task

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

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

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

Line 4: 2 1 2 3 2 3 7
Level Displayed for Each Attribute in First Concept, plus point allocation

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

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

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

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

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

1 3 lbs
2 5 lbs
3 7 lbs

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

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

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

Line 5: 3 3 3 1 3 1 3
Level Displayed for Each Attribute in Second Concept, plus point allocation

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

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

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

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

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

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

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

7.4

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

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

When you specify that some attributes use "User-specified" coding, you are telling CBC/HB that certain or all values found in the .CHO or .CHS files should be used as-is within the design matrix (these may also be specified in the alternate .CSV data file formats, if you prefer to work with these). In the example below, we'll let CBC/HB code automatically all but one of the parameters to be estimated. ***Even though we'll only show an example for a .CHO file, the same procedure is followed within the relevant format for the .CHS file.***

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

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

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

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

We should point out that CBC/HB operates better if the variances of the estimated parameters are not too different from the prior assumptions of unity, and the absolute magnitude of the parameters are not vastly different. Convergence occurs much more quickly and properly if this is the case. If there is enough information at the individual level, the priors should have little effect on the outcome. But, with CBC projects, the amount of information for each respondent is often sparse, and the scaling of columns within the design matrix and resulting variance/scale of the estimated parameters may therefore be important.

We recognize that the effects-coding procedure used for the categorical attributes results in columns in the independent variable matrix coded as -1, 0, or +1. These codes are zero-centered, and have a range of 2 (+1 minus -1). Effects coding has worked well in practice with HB. We also suggest that you zero-center your variables and that they have a range not too different from effects coding.

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

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

The zero-centered prices are therefore -17.5, -7.5, 2.5, 22.5. The next step is to convert the values to a range near 2. We know that the current range is 22.5 minus -17.5 = 40. We would like the range to be 2. The target range (2) divided by the actual range (40) is equal to 0.05. Multiplying the zero-coded values by 0.05 results in:

```
-0.875
-0.375
+0.125
+1.125
```

Now that we have zero-coded the values for Price and given them a range that should result in proper convergence for the estimated coefficient, we are ready to inform CBC/HB regarding this coding procedure and place the values within the studyname.CHO file.

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

1. From the *Attribute Information* tab, right-click the Price attribute label, and select ***Change Coding Method | User-specified***. This tells CBC/HB to use the values (after zero-centering) currently found in the .CHO or .CHS file for this attribute within the design matrix.

2. Next, the user-defined coded values for Price need to be placed within the studyname.CHO file. Recall that the default coding for the studyname.CHO file for respondent 2001 looked like:

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

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

```
2001 7 6 18  0
6  1  2  3  4  5  6
5  1
2  1  2  1  2   0.125
1  1  3  1  1  -0.375
1  3  3  2  2  -0.875
2  3  2  2  1   1.125
3  2  1  1  2   0.125
3  99
```

Note that all the values must be space-delimited within the studyname.CHO file. Make sure there is

at least one space between all values. The values may include up to six decimal places of precision.

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

7.5

Appendix E: Analyzing Alternative-Specific and Partial-Profile Designs

Introduction

The CBC/HB System analyzes data from the studyname.CHO or studyname.CHS files, which are in text-only format. The studyname.CHO file is automatically generated by the CBC or CBC/Web systems, but you can create your own studyname.CHO or studyname.CHS files and analyze results from surveys that were designed and conducted in other ways.

Alternative-Specific Designs

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

Consider the following design:

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

There are actually six different attributes in this design:

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

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

In the studyname.CHO and studyname.CHS files (described in detail in Appendix C), there is a row of

coded values describing each product concept displayed in each task. Consider a two-alternative task with the following options:

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

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

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

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

Analyzing Partial-Profile Designs

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

Analyzing More Than One Constant Alternative

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

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

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

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

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

7.6

Appendix F: 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:

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

and

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

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:

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

And, an equivalent expression is:

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

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.

7.7

Appendix G: How CBC/HB Computes the Prior Covariance Matrix

In early 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.

Users now have the ability to specify prior variances rather than having to assume they are equal to unity, as well as the prior degrees of freedom. Advanced users may specify their own prior covariance matrix. See the Advanced settings for more information.

If you are curious regarding the prior covariance matrix that has been used for your most recent run, please refer to the *studyname_priorcovariances.csv* file, which is one of the default output files. This is a comma-separated values 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, ... a_I) and b with J levels (b_1, b_2, ... 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 (an identify matrix with the prior variance along the diagonal). 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.

7.8 Appendix H: Generating a .CHS File

The .CHS file format is used when respondents have used constant sum (chip allocation) for answering CBC questions. Its format is provided in Appendix C. Until allocation-based CBC questionnaires are directly supported within the SMRT or Lighthouse Studio (formerly known as SSI Web) platforms, some users may find a tool provided in CBC/HB convenient for generating .CHS files.

Clicking ***Tools | Convert CHO to CHS…*** accesses a tool that takes data provided in a .CHO format and converts it to a .CHS format. Optionally, the user can supply a separate text-only file of respondent answers (delimited by spaces, tabs, or hard returns). CBC/HB combines the allocation information found in the file of respondent answers with the information provided in the .CHO file, creating a .CHS file. Any choice responses found in the .CHO file are overwritten by allocation responses found in the optional file of respondent answers.

For example, the first two lines (representing the first two respondents) in the text-only file containing respondent answers may look like:

```
1001   50   0  50  80  20   0 100   0   0 . . . (more data follow)
1002  100   0   0  90   0  10  80  20   0 . . . (more data follow)
```

This file must begin with the respondent number. The respondent records do not need to be in the same order as in the .CHO file and the two files do not need to have the same number of cases. If a respondent's allocation data are not provided but that respondent exists in the .CHO file, original answers in the .CHO file for this respondent are written to the .CHS file.

In this example, three concepts were shown per choice task. Therefore, the first three values (representing task #1) sum to 100, the next three values (representing task #2) sum to 100, etc. It is not necessary that the values within a task sum to 100 (but the total points allocated within a task should not exceed 100). If respondents skipped a task, you should allocate 0 points to all concepts within that task.

Example:

1. With CBC for Windows or CBC/Web, the user creates a paper-and-pencil questionnaire and fields the study.
2. Using a text-only file containing respondent numbers, questionnaire versions, and "dummy" choice responses, the user utilizes the automatic ***Accumulate Paper & Pencil Data*** function and the ***File | Export*** functionality to export the data to a .CHO file.
3. The user provides a text-only file of respondent answers to the allocation questions (as described above).
4. Using the ***Tools | Convert CHO to CHS…*** functionality in CBC/HB, the user merges the information from the .CHO file and the file of respondent answers to create a final .CHS file.
5. CBC/HB estimates part worths using the final .CHS file.

Future versions of our CBC software may automatically collect allocation-based responses and generate a .CHS file, which will eliminate the extra steps involved.

7.9

Appendix I: 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.

7.10 Appendix J: Estimation for Dual-Response "None"

Introduction

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

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

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

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

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

Modeling Dual-Response None in CBC/HB

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

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

Data Setup in the CBC/HB File

We already introduced the .CHO file layout in Appendix C. When using "Dual-Response None" questionnaires, make the following modifications:

1. Each respondent records begins with a header that has five values. Set the fifth value equal to "2."

2. In the standard .CHO layout, the coding of each task is completed by a line with two values: "Choice" and "Task Duration." With the "Dual-Response None" format, each choice task is completed by a line with four values, such as:

3	**27**	**1**	**4**
1st stage Choice	Task Duration	Buy=1 No Buy=2	Task Duration

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

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

7.11 Appendix K: Estimation for MaxDiff Experiments

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

Please consider dining experiences in fast food restaurants. Among these attributes, which is the most and the least important to you?		
Which is Most Important?		Which is Least Important?
❍	Good tasting food	❍
❍	Offers healthy selections	❍
❍	Friendly service	❍
❍	Fun atmosphere	❍

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

Coding the .CHO File for MaxDiff Experiments

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

Consider a MaxDiff study with the following specifications:

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

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

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

```
1001   0    7    20    0
```

```
4   1
 0    0    0    0    0    0    1
 0    0    0    0    0    0    0
 0    0    1    0    0    0    0
 0    1    0    0    0    0    0
1   99
4   1
 0    0    0    0    0    0   - 1
 0    0    0    0    0    0    0
 0    0   - 1   0    0    0    0
 0   - 1   0    0    0    0    0
3   99
(etc. for 9 more sets)
```

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

Line 1:

1001	**0**	**7**	**20**	**0**
Respondent number 1001	No segmentation variables	7 attributes	20 total sets	No "None"

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

Line 2:

4	**1**
Next follows a set with 4 items	One selection from this set

Line 3:

0 0 0 0 0 0 1

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

Line 4:

0 0 0 0 0 0 0

Dummy codes representing item 8. In our study, if the 8^{th} item is present, all seven values are at 0.

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

Line 7:

1	**99**
The item in row 1 of this set is best	A filler value (time) of 99 to be compatible with .CHO format

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

Line 8:

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

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

Line 13:

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

---

Estimating the Model using CBC/HB

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

We should note that when using HB to estimate parameters for many items under dummy coding, the estimates of the parameters (relative to the reference "0" level) can sometimes be distorted downward, often quite severely when there are many items in the study and when the number of questions asked of any one respondent is relatively few. (This distortion of course makes it appear as if the "omitted" item's estimate is "too high" relative to the others.) To see if this difficulty is appearing for your data set, you might try coding a different level as the "omitted" item and compare the results. Or, you could take the step described below to avoid the problem.

To avoid potential problems when dealing with dummy coding under HB and very sparse data conditions, you should specify a more appropriate "custom prior covariance matrix" from the *Advanced Estimation Settings* tab. Check the *custom prior covariance matrix* box and click ***Edit...***. Specify a (k-1) x (k-1) matrix of values, where k is equal to the total items in your MaxDiff study, a k-1 is equal to the total number of parameters to be estimated (also equal to the number of attributes in the *Attribute Information* tab). If you wish to specify a prior variance of 1.0 (a typical default), the matrix is composed of "2"s across the main diagonal, and "1"s in the off-diagonal positions, such as:

```
2 1 1 1 . . .
1 2 1 1 . . .
1 1 2 1 . . .
1 1 1 2 . . .
. . . . . . .
. . . . . . .
. . . . . . .
```

To specify a different prior variance, multiply all values in the prior covariance matrix above by the desired variance constant.

When you have finished these steps, click ***Estimate Parameters Now...*** from the *Home* tab. Utility values are written to the .HBU and .CSV files. Remember, the utility of the omitted value for each respondent is 0, and the other items are measured with respect to that omitted item.

A Suggested Rescaling Procedure

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

1. Insert the score for the omitted item for each respondent (score of 0).

2. Zero-center the weights for each respondent by subtracting the average item weight from each weight.

3. To convert the zero-centered raw weights to the 0-100 point scale, perform the following transformation for each item score for each respondent:

$$e^{Ui}/(e^{Ui} + a - 1)$$

Where:

U_i = zero-centered raw logit weight for item i

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

a = Number of items shown per set

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

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

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

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

7.12 Appendix L: Hardware Recommendations

The CBC/HB System generally requires a fast computer with a generous amount of memory and storage space. Most computers available for purchase today are adequate to run CBC/HB for most problems.

Many people equate processor speed with overall speed. This is not the only factor. Overall speed is determined by the combination of processor speed, storage space speed/availability, and memory availability. The fastest processors on the market will not make the system significantly faster if the storage space and memory are too low. This is especially true for CBC/HB. While the algorithm is computationally intensive, it is often 'I/O bound' which means the running time is more dependent on factors such as storage space and memory. The data files in use by CBC/HB are either located on the hard drive or in memory, and are referenced quite often. If these are slow, or their capacities are insufficient, the fastest processor on the market will still appear to be slow.

In general, hard drive performance is optimal when the total used space is less than 50% of the capacity of the drive. Keep this in mind when selecting a hard drive. Other factors that affect performance are: transfer rate, seek time, and the drive's RPM (revolutions per minute) speed. As Solid State Drives (SSD) become cheaper, they present an ideal option for computers running CBC/HB.

Memory is often overlooked as a speed-increasing factor. When operating systems such as Windows begin to run out of physical memory, they swap data in memory to and from the hard drive (this state is called thrashing). This can bring a system to a standstill. Again, the rule of thumb is to have about twice as much memory as typically used. Various utilities can diagnose how much memory is commonly used. If the system takes a long time to start up, or if you notice that the hard drive activity light is on almost constantly, these are signs of insufficient memory.

Other common performance questions:

Is there a 64-bit version of CBC/HB? Would I benefit from a 64-bit system?

CBC/HB v5 runs in 64-bit mode when using 64-bit versions of Windows, and in 32-bit mode on 32-bit Windows. Our tests of 64-bit performance can be found in the Winter 2008 Sawtooth Solutions ("X64 HB: Making Fast Even Faster"), available at Sawtooth Software's website. In those tests, we found performance increases typically between 10%-25%.

Can CBC/HB use multicore processors? Would such a system be faster?

Today new machines contain multicore processors, which allow multiple tasks to be performed simultaneously. For a program to take advantage of multiple processors it must be "multithreaded," which means that the software must run independent tasks on separate "threads." The key is that each thread must be independent (i.e. rendering each pixel of an image can be done separately). While there are portions of the CBC/HB algorithm that are computed independently, overall the algorithm is highly dependent and thus does not lend itself well to multi-threading. We continue to do research in this area.

However, CBC/HB does benefit indirectly from having multiple cores/processors. When faced with a process doing an intense computation, the operating system will attempt to give the process its own processor, and shift other processes to the other processors. So, if CBC/HB can run on one processor while everything else runs on others, it will run faster than when it must share only one processor.

How do I configure my system to use a Solid State Drive?

If the only drive in the computer is a SSD, then no further configuration is necessary. If the SSD is not

the primary drive in the system, then Windows should be configured to use the SSD for temporary files. CBC/HB project files should also be stored on the SSD for estimation.

Can I run CBC/HB estimation on a project located on a network drive?

CBC/HB does its best to use the local temporary files location, but in practice it is best to run estimation from a local drive. Performance from a network drive is typically poor.

7.13

Appendix M: Calibrating Part-Worths for Purchase Likelihood

If you have asked additional Calibration Concept questions and formatted the calibration data in an appropriate format (see layout below), you can calibrate the part worth utilities for use with Sawtooth Software's purchase likelihood simulation model within the market simulator. To calibrate the data, select ***Tools | Calibrate Utilities...***. When you calibrate utilities, a new file is written named **studyname_calib.hbu**.

Background

In the mid-1990s, before computers were fast enough to make HB estimation feasible in market research applications for medium to large CBC datasets, Sawtooth Software developed a fast procedure for individual-level part worth estimation called ICE (Individual Choice Estimation). As part of the ICE procedure, we provided a way for users to display a Calibration Concept section (similar to that offered by the earlier ACA system), and to use results from that section to calibrate (rescale) the utilities for use within Sawtooth Software's Purchase Likelihood Simulation Model.

Calibration Concept sections involve showing respondents multiple product concepts (in full-profile) one-at-a-time and asking them to rate their purchase likelihood (typically on a 100-point scale) for each concept.

Calibrated utilities are scaled such that when the sums of product utilities are submitted to the following equation, they produce a least-squares fit to respondents' stated purchase likelihood on a 100-point scale:

$$\text{Purchase Likelihood} = 100 * [e^{Ui} / (1 + e^{Ui})]$$

where Ui is the total utility for the product concept and e is the exponential constant.

The Regression Equation

To calibrate the part worth utilities to fit purchase likelihood, we fit an ordinary least squares regression (for each respondent) relating part worth utilities to purchase likelihood scores.

The respondent answers (the dependent variable in the regression) are transformed to logits (log odds). Because the logit transform is very sensitive at high and low probabilities, any probabilities more extreme than 0.05 or 0.95 are first truncated to those bounds (so, if a respondent gives a product concept a score of 100 on a 100-point scale, the response is trimmed to 0.95 when converting responses to probabilities). We transform the probability-scaled responses (the dependent variable) to logits according to the formula:

$$\ln[p/(1-p)]$$

The independent variable in the regression is simply the sum of the raw utilities (taken from the .hbu file) for the each product concept.

Two parameters are estimated via ordinary least-square regression: a slope and intercept. The part

worths are calibrated and written to the **studyname_calib.hbu** file by multiplying them by the slope, and then adding the intercept divided by the number of attributes in your study to all part worths. Each respondent's fit statistic in the part worth utility file (.hbu file) is replaced with the R-squared (times 1000) from the regression.

There are two exceptions. For respondents with regression coefficients less than zero, the regression coefficient is set at 0.01 and the r-square is reported to be zero. For respondents with regression coefficients greater than or equal to 0, but less than 0.01, the regression coefficient is set at 0.01, and the r-square is not changed. In either case, the intercept is solved to best fit calibration responses.

We recommend that at least five or six product concepts be shown to respondents (in full profile), so that the number of observations is more than double the number of parameters to estimate in the regression.

It is best if the product concepts shown to respondents have large differences in expected utilities. Larger expected differences in the dependent variable (the Y's in the regression) will lead to greater stability of the betas (the scaling parameters for the calibration). We'd recommend showing respondents a very poor product concept, followed by a very good concept, and then concepts in between.

.CAL File Format

The calibration data must be formatted in an appropriate text-only (blank-delimited) file with the following format:

```
1001   5   3
 3  1  1  1
12  3  3  3
16  1  2  3
76  2  2  2
15  3  2  1
```

Line 1:

field 1, respondent number
field 2, number of calibration concepts
field 3, number of attributes

Line 2: respondent answer, followed by attribute level codes for the concept displayed in calibration concept #1, in the same order as specified in the *studyname*.ATT file. There are as many attribute codes as attributes in the study.

Line 3 through 6: Same specifications as Line 2 for calibration concepts 2 through 5.

(Note: although we have shown a data file with hard returns dividing the different sections of the respondent record, each respondent record can be formatted on a single line.)

7.14 Appendix N: Scripting in CBC/HB

CBC/HB v5 includes the ability for advanced researchers to write scripts to automate common tasks. This includes the creation of projects, the manipulation of settings, and estimation. This allows researchers doing repetitive tasks to automate these common projects with relatively little effort. Using CBC/HB's scripting features does not require any knowledge of programming.

Using Scripting in CBC/HB

Scripting in CBC/HB is done using the CBC/HB Command Interpreter, which is similar to a DOS-style command prompt. The interpreter is launched by running the *CBCHBCon.exe* program, located in the CBC/HB installation folder. This can be run by clicking **Sawtooth Software | Tools** from Windows' Start menu, or alternatively from the Windows Run Command window or a command prompt by typing *CBCHBCon*.

When run, a **CBCHB>** prompt appears in a console window. Commands may be entered after a prompt, similar to a Windows command prompt. Commands can be followed by optional arguments, as shown below:

```
CBC/HB Command Interpreter v5.0.0

CBCHB> CreateProject "D:\Studies\TV.cho"
OK

CBCHB> Exit
```

Creating script files

A script of commands can be saved to a text file which is then passed to the command interpreter. The interpreter will run the commands in the file as if they were typed directly by the user. For example, a text file may contain the following commands:

```
CreateProject "D:\Studies\TV_US.cho"
SetAttributeCodingMethod 6 Linear
SetAttributeLevelValue 6 1 300
SetAttributeLevelValue 6 2 350
SetAttributeLevelValue 6 3 400
SetAttributeLevelValue 6 4 450
SaveAs "D:\Studies\TV_US.cbchb"
CreateProject "D:\Studies\TV_UK.cho"
SetAttributeCodingMethod 6 Linear
SetAttributeLevelValue 6 1 300
SetAttributeLevelValue 6 2 350
SetAttributeLevelValue 6 3 400
SetAttributeLevelValue 6 4 450
SaveAs "D:\Studies\TV_UK.cbchb"
Exit
```

To run a script file, type *CBCHBCon /in:filename* (where filename is the full path to the file, e.g. "D:\Studies\myscript.txt". If the path contains spaces, surround the path with quotes. NOTE: Unless the 'Exit' command is in the script, the interpreter will remain open for input from the keyboard after the script

is finished.

Saving output

A log of the commands and results can be saved by specifying an output file. To specify an output file, type *CBCHBCon /out:filename* (where filename is the full path to the file, e.g. "D:\Studies\output.txt". If the path contains spaces, surround the path with quotes. The /in: and /out: parameters may be used simultaneously.

Using scripts within another program

Users may wish to run CBC/HB within the context of another program such as Excel or SPSS. From within these programs script files can be generated, which can then be executed in the interpreter. Sawtooth Software can support problems with the interpreter, but does not give support on generating or executing scripts from other software. Please refer to the software's documentation on how to create files or run external programs.

Script errors

If there is a problem or a mistake in the script, an error is displayed on the line following the command. For example:

```
CBCHB> airspeed "unladen swallow"
Error: Unknown command
```

Script Command Reference

Run strScriptFile

Runs a script file. The parameter *strScriptFile* should ocntain the full path to the script file, e.g. "D:\Studies\TV.script" with quotes included. Usage:

Run "D:\Studies\TV.script"

Rem

Specified a comment in the script. Anything following the rem command until the return key is pressed is ignored. Usage:

rem Creating a new project
CreateProject "D:\Studies\TV.cho"

CreateProject strDataFile

Creates a new project using an existing .cho or .chs file. These are the only types of input files supported at this time. The parameter *strDataFile* should contain the full path to the data file, e.g. "D:\Studies\TV.cho" with quotes included. The project is created with default settings. If a .att file is available, it will automatically be imported as well. Usage:

CreateProject "D:\Studies\TV.cho"

EstimateProject strProject strImage

Estimates an existing CBC/HB v5 project file. The parameter *strProject* should contain the full path to the project file, e.g. "D:\Studies\TV.cbchb". The parameter *strImage* is optional and specifies the name of a file to save the estimation graph to (in .png format). Usage:

```
EstimateProject "D:\Studies\TV.cbchb"
EstimateProject "D:\Studies\TV.cbchb" "D:\Studies\TV.png"
```

SaveAs strProject

Saves a project previously created using *CreateProject*. The parameter *strProject* should contain the full path to the project file, e.g. "D:\Studies\TV.cbchb". If the folder does not exist, the command will fail. This command only needs to be called once the project settings have been modified to their desired values, but must be called prior to estimation. Usage:

```
CreateProject "D:\Studies\TV.cho"
SaveAs "D:\Studies\TV.cbchb"
```

SetDemographicFile strDemoFile

Specifies a comma-separated values file containing demographic variables. The parameter *strDemoFile* should contain the full path to the file, e.g. "D:\Studies\TV.csv'. Usage:

```
SetDemographicFile "D:\Studies\TV.csv'
```

SetAttributeLabel iAtt strLabel

Changes the label of an attribute (by default attributes are labeled "Attribute 1", "Attribute 2", etc.). The parameter *iAtt* is the attribute index (ranging from 1 to N, where N is the number of attributes). The parameter *strLabel* is the new attribute label. Usage:

```
SetAttributeLabel 1 "Brand"
```

SetAttributeCodingMethod iAtt coding

Changes the attribute coding of an attribute. The parameter *iAtt* is the attribute index (ranging from 1 to N, where N is the number of attributes). The parameter *coding* is the new attribute coding using one of the following values: *PartWorth*, *Linear*, *UserSpecified* and *Excluded*. By default all attributes are *PartWorth*. Usage:

```
SetAttributeCodingMethod 6 Linear
```

SetAttributeLevelLabel iAtt iLev strLabel

Changes the label of a level (by default levels are labeled "Level 1", "Level 2", etc.). The parameter *iAtt* is the attribute index (ranging from 1 to N, where N is the number of attributes). The parameter *iLev* is the level index within attribute iAtt (ranging from 1 to M, where M is the number of levels in the attribute). The parameter *strLabel* is the new level label. Usage:

```
SetAttributeLevelLabel 1 1 "Brand X Burgers"
```

SetAttributeLevelValue iAtt iLeve dValue

Changes the value of a level (by default levels have values of 1, 2, 3, etc). These values are only used if the attribute has *Linear* coding. The parameter *iAtt* is the attribute index (ranging from 1 to N, where N is the number of attributes). The parameter *iLev* is the level index within attribute iAtt (ranging from 1 to M, where M is the number of levels in the attribute). The parameter *dValue* is the new value of the level. Usage:

SetAttributeLevelValue 6 1 300

AddInteraction iAtt1 iAtt2

Specifies a two-way interaction between attributes. This command is called for each desired interaction. The parameters *iAtt1* and *iAtt2* are the attribute indices of the involved attributes (ranging from 1 to N, where N is the number of attributes). Usage:

AddInteraction 1 2
AddInteraction 2 3

SetBurnInIterations iValue

Sets the number of iterations performed before results are saved. The default is 10000 iterations. The parameter *iValue* is the new number of iterations. Usage:

SetBurnInIterations 20000

SetSavedDraws iValue

Sets the number of iterations saved after convergence is assumed. The default is 10000 draws. The parameter *iValue* is the new number of draws. Usage:

SetSavedDraws 20000

SetSaveRandomDraws bValue

Sets whether individual draws are saved or not. By default draws are not saved. The parameter *bValue* should be set to *true* to save draws, or *false* otherwise. Usage:

SetSaveRandomDraws true

SetSkipDraws iValue

Sets the skip factor for saving individual draws when being saved. This value is only used if random draws are saved. The default value is 10. The parameter *iValue* is the new skip factor. Usage:

SetSkipDraws 5

SetSkipGraph iValue

Sets the skip factor for displaying information to the graphical display. This value is only used if running using the graphical progress window. The default value is 10. The parameter *iValue* is the new skip factor. Usage:

SetSkipGraph 100

SetSkipLog iValue

Sets the skip factor for writing information to the log file. The default value is 100. The parameter *iValue* is the new skip factor. Usage:

SetSkipLog 1000

SetTaskWeight dValue

Sets the total task weight used in estimation. If tasks are discrete choice (one response per task), this value has no effect. The default is 5.0. The parameter *dValue* is the new task weight. Usage:

SetTaskWeight 2.0

SetEstimateNone bValue

Sets whether the none option is estimated if present. By default the none is estimated. The parameter *bValue* should be set to *true* to estimate the none option, or *false* otherwise. Usage:

SetEstimateNone false

SetVariableCoding varcoding

Sets the variable coding of levels during the build process. The parameter *varcoding* is the new coding, which may be either *Effects* or *Dummy*. The default is to use effects coding. Usage:

SetVariableCoding Dummy

SetRandomSeed iValue

Sets the value used to seed the random generator during estimation. The default is 0, which indicates to use a value from the system clock (and is reported in the estimation log file). The parameter *iValue* is the new random seed. Usage:

SetRandomSeed 1

SetUseConstraints bValue

Sets whether constraints should be imposed during estimation. Constraints are not used by default. The parameter *bValue* should be set to *true* to use constraints, or *false* otherwise. Usage:

SetUseConstraints true
AddLevelConstraint 1 1 2

AddLevelConstraint iAtt iLev1 iLev2

This command adds a constraint between two levels. The parameter *iAtt* is the attribute index, *iLev1* is the level index of the preferred level, and *iLev2* is the index of the less preferred level.

Usage:

SetUseConstraints true
AddLevelConstraint 1 1 2

AddLinearZeroConstraint iAtt comparator

This command adds a constraint on a linear attribute to be greater than or less than zero. The parameter *iAtt* is the attribute index, and *comparator* can be either *GreaterThan* or *LessThan*. Usage:

AddLinearZeroConstraint 1 GreaterThan

AddTwoLinearsConstraint iAtt1 comparator iAtt2

This command adds a constraint that one linear attribute must be greater or less than another. The parameter *iAtt1* is the index of the first attribute, *comparator* can be either *GreaterThan* or *LessThan*, and *iAtt2* is the index of the second attribute. Usage:

AddTwoLinearsConstraint 1 GreaterThan 2

SetUseRespondentFilter bValue

Sets whether respondent filters will be applied. The default is to not filter respondents. The parameter *bValue* should be set to *true* to filter respondents, or *false* otherwise. Usage:

SetUseRespondentFilter true
AddRespondentFilter 1 Equals 2

AddRespondentFilter iVariable comparator dCompareValue

Adds a respondent filter. This command is called for each desired filter. Filters require that the data file contain demographic variables or a separate demographics file has been specified using *SetDemographicFile*. The parameter *iVariable* is the index (from 1 to N, where N is the number of variables) of the demographic. The parameter *comparator* is one of the following: *Equal*, *NotEqual*, *GreaterThan*, *LessThan*, *GreaterThanOrEqual*, or *LessThanOrEqual*. The parameter *dCompareValue* is the value to compare against. Respondents are included in analysis if they meet the criteria of all filters. Usage:

SetUseRespondentFilter true
AddRespondentFilter 1 Equal 2

SetPriorDegreesOfFreedom iValue

Sets the prior degrees of freedom. The default is 5. The parameter *iValue* is the new degrees of freedom. Usage:

SetPriorDegreesOfFreedom 10

SetPriorVariance dValue

Sets the prior variance. The default is 1.0. The parameter *dValue* is the new prior variance. This value is ignored if using a custom prior covariance matrix. Usage:

SetPriorVariance 1.0

SetUsePriorCovarianceMatrix bValue

Sets whether a custom prior covariance matrix is to be used. A custom matrix is not used by default. The parameter *bValue* should be set to *true* to use a custom prior covariance matrix, or *false* otherwise. Usage:

SetUsePriorCovarianceMatrix true

SetPriorCovarianceMatrix matrix

Sets the custom prior covariance matrix. The parameter *matrix* should be expressed as a string (in quotes), with row values comma-delimited and contained with brackets, and rows comma-delimited. Usage:

SetPriorCovarianceMatrix "[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]"

SetAlphaMatrixType enumType

This command changes how the alpha matrix is computed. The parameter *enumType* may be *Default*, *CustomPrior* or *Covariates*. Usage:

SetAlphaMatrixType Covariates

SetCovariate iIdx bInclude strLabel enumType iNumCategories

Sets particular settings regarding covariates if the alpha matrix type is set to Covariates. The parameter *iIdx* is the index of the covariate in the demographic file (ranging from 1 to N where N is the number of variables). The parameter *bInclude* is either *true* or *false*, to indicate whether this variable should be used as a covariate in estimation. The parameter *strLabel* is the label of the variable. The parameter *enumType* can be either *Categorical* or *Continuous*, depending on the type of variable used. The parameter *iNumCategories* is the number of categories (if the type is *Categorical*). Usage:

SetCovariate 1 true "Gender" Categorical 2
SetCovariate 2 true "Age" Continuous

SetCustomPriorAlphaMeans means

Sets the prior alpha means if the alpha matrix type is set to *CustomPrior*. The parameter *means* should be expressed as a string (in quotes), with values comma-delimited and contained with brackets. Usage:

SetCustomPriorAlphaMeans "[1, 1, 0, 0, 1]"

SetCustomPriorAlphaVariances variances

Sets the prior alpha variances if the alpha matrix type is set to *CustomPrior*. The parameter *matrix* should be expressed as a string (in quotes), with row values comma-delimited and contained with brackets, and rows comma-delimited. Usage:

SetCustomPriorAlphaVariances "[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]"

Index

- 6 -

64-bit processing 6

- A -

Acceptance rate 14
Allenby (Greg) 3, 56, 60
Alpha Matrix 51
Alpha.csv file 55, 62
Alternative-specific attributes 76
ATT file 17, 31, 67
Attribute information tab 31
Avg Variance 25

- B -

Batch mode 17
Bayes theorem 9
Best/worst scaling 31, 48, 90
BET file 44
Burn-in iterations 25

- C -

Calibrating part-worths 6, 96
Capacity limitations 5
CBC software 17
CBC/Web software 17
Chip allocation 78
CHO file 17, 23, 29, 31, 62, 73, 84
CHO file format 67
Choice task filter 34
CHS file 17, 23, 29, 31, 62, 73, 84
CHS file format 67
Conditional probability 9
Constant-sum data 78
Constraints 35, 44
Convergence 13, 14, 25, 35
Covariance matrix 49, 55
Covariances.csv file 55, 62
Covariates 6, 53
CSV file 55, 62
CSV Input Format 17, 19
Custom Prior Covariance Matrix 49

- D -

Degrees of freedom 49
Densities 14
DRA file 55, 62
Draws 13, 25, 35, 44, 57
Dual-response "None" 88
Dummy coding 31, 38, 48, 81

- E -

Effects coding 23, 31, 48, 81
Exclude attribute 31

- G -

Gibbs sampling 13, 65

- H -

Hardware recommendations 5, 94
HBU file 55, 62
HBU file format 62
Hit rates 44
Holdout tasks 34, 57
Home tab 23
Huber (Joel) 1, 57, 60

- I -

ICE 3, 56, 76
Interactions 31

- J -

Jump size 14
Jumping distribution 14

- L -

Latent class 56, 76
Lenk (Peter) 3, 56, 60
Likelihood of the data 14, 25, 76
Linear coding 31, 44
LOG file 25, 35
Log likelihood 25
Logit model 12, 14, 25

- M -

Main effects 23
MaxDiff scaling 31, 48, 90

Meanbeta.csv file 55, 62
Metropolis Hastings algorithm 13, 14
Monotonicity constraints 35, 42, 44
Monte Carlo Markov Chain 13

- N -

None alternative 23, 57, 62, 67, 88

- O -

Opening a project 17
Output files 55

- P -

Parameter RMS 25, 44
Part worth coding 31
Partial-profile designs 76
Percent certainty 25
Point estimates 35
Posterior probability 9, 14
Prior covariance matrix 48, 49, 81
Prior degrees of freedom 48, 81
Prior variance 48, 49, 81
Priorcovariances.csv file 62
Priors 14, 48, 81
Purchase Likelihood 96

- Q -

Quantitative attributes 31
Quick start instructions 3

- R -

Random starting seed 48
Respondent filter 35
Respondent filters 40
Restarting 28
RLH 25, 44
Root likelihood 25, 44

- S -

Scripting in CBC/HB 98
Simultaneous tying 44
Skip factor 35
Speed 94
Starting seed 47
Stddev.csv file 55, 62
SUM file 44
Summary.txt file 55, 62

- T -

Task weight 35, 38, 78
Tie Draws program 44

- U -

User-specified coding 31, 73
Utility constraints 35, 44, 85

- V -

Variance 25, 44

- W -

Wishart distribution 13