Sawtooth Software

RESEARCH PAPER SERIES

Has My HB Model Converged?
How many iterations should I run?

Bryan Orme
Sawtooth Software, Inc.

3210 N. Canyon Rd., Provo, Utah
+1 801 477 4700
www.sawtoothsoftware.com

Has My HB Model Converged?
How many iterations should I run?

Bryan Orme, Sawtooth Software

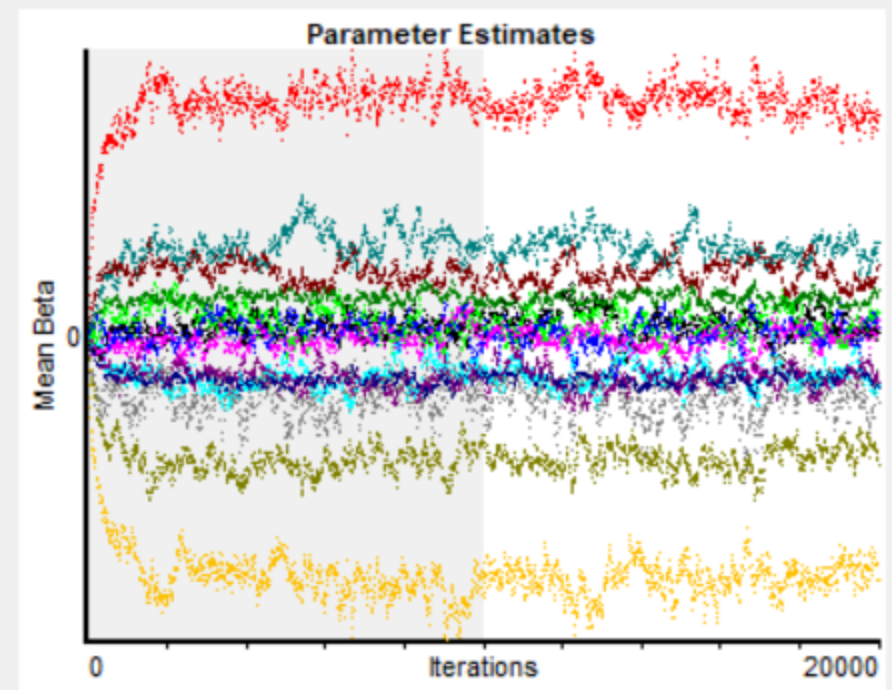
June 2025

HB Estimation Overview

Hierarchical Bayes MNL takes 1000s of iterations to converge on useful utility estimates at the individual level. And the researcher needs to decide how many to run. The default in Sawtooth's program for CBC estimation is to use 20,000 iterations to get to the point at which the next several thousand proposals (draws) will be useful. HB initializes all utilities at an uninformative 0 starting point. During the first burn-in phase iterations, the utilities are noisy and relatively small, but over several hundred or thousand iterations, they begin to expand from 0 and fit the data better. The burn-in phase is depicted as the gray region in the history of the iterations chart below, from Sawtooth's real time output.

For the next many thousands of iterations (the white region), we use the draws (one draw per iteration) for each respondent to estimate the final utilities we use in market simulators. Although it's more proper to use the full information from the used draws (often thinned by a skip factor 10 or more), for ease of use most practitioners average across them to obtain a single set of utilities (point estimates) for each respondent.

Below is the history of average part-worth utilities for the population for each of 13 levels in a small CBC study. For such simple studies, the common Gibbs sampling (via Metropolis-Hastings) approach that Sawtooth uses converges quickly and reliably (typically in a couple minutes or fewer). A widely used R package "bayesm" also uses the same algorithm.



History of HB Utility Estimates, by Iteration

Reliability, Predictive Validity, and Convergence

Practitioners want their HB estimations to be reliable so that repeating the estimation from different random starting seeds would give essentially identical results (that way we protect against getting substandard results due to an unlucky starting seed). They also want them to have high predictive validity: predictions from the market simulator should fairly closely match target out-of-sample holdout tasks, or better yet and under ideal conditions, real market shares.

Convergence happens when each of the part-worth utilities wiggles (oscillates) around an unbiased estimate with no remaining trend. The wiggle from iteration to iteration is actually a good thing, as this variability characterizes the uncertainty in our model due to limited sample size, limitations of our experimental design plan, and the imperfect and therefore realistic nature of humans' responses to CBC tasks. In short, convergence means that if we repeat the HB run from a different starting seed, the utilities will settle on very close to the same means and variances from one run to another.

There's a formal statistic for measuring convergence in HB called R-hat (AKA the Gelman-Rubin statistic). It's calculated by running multiple HB runs from different seeds. At least four is suggested, and I found that fewer chains (sequences of estimates from different random starting seeds) than that tend to give more false positives regarding failure to converge. The R-hat compares between-chain variance to within-chain variance across the used draws (the draws after assumed convergence). See the Appendix for calculation details. We can estimate an R-hat value for each part-worth utility in a CBC (or MaxDiff) study and the accepted criterion for formal convergence is that none of the R-hat values should exceed 1.1.

Most CBC Studies Following Best Practices Converge within 20K/10K

At the 2024 Sawtooth Research Conference, Kurz and Rausch examined many simulated CBC datasets, finding that those with about 30 estimated parameters or fewer converge within about 20K burn-in iterations. They recommended sampling another 10K iterations during the "used" iterations phase. (Though, I'd recommend sampling across 30K or more if you can afford the time.)

I recently looked at 9 real CBC datasets and found Kurz and Rausch's findings to be spot on. Sawtooth's defaults work well in most choice modeling conditions.

Kurz and Rausch suggested that data sets involving more than about 30 parameters should do even more than 20K burn-in iterations before assuming convergence (recommending 50K or even 100K). At the 2025 Sawtooth Turbo Choice Modeling Event, Kevin Lattery from SKIM warned that the traditional Gibbs Sampling HB MNL algorithm often has a hard time converging on reliable results for very large CBC problems, such as FMCG data sets involving 100 or more SKUs plus price. In those cases, Kevin has found that Stan with its HMC (Hamiltonian Monte Carlo) no U-turn sampler obtains better convergence and predictive validity at the individual level (though he found that aggregate simulations such as from the typical simulators researchers build still worked well using traditional HB MNL).

In my examination of 9 real CBC datasets, I found two anomalies that didn't converge using the default iterations. One involved some harmful prohibitions between two attributes that Sawtooth's recommended design testing procedure prior to fielding would have flagged. The other involved a complex list of attributes and levels that did not converge well within the default iterations (largest R-hat value was 1.32). I purposefully included this study in my analysis because it was so demanding: a full-profile CBC study with a total of 55 levels including a None (n=1201). The 13 attributes had these number of levels, respectively: 6, 5, 2, 4, 4, 2, 5, 5, 4, 2, 4, 5, 6, with no attribute prohibitions.

Given that this 55-level CBC data set did not converge using the default settings in Sawtooth, I decided to experiment with the number of burn-in and used iterations to see if I could obtain acceptable convergence per R-hat—which leads me to pose an interesting thought experiment for you:

Ignoring time to compute (only 90 minutes for the longest run below), which of the following HB runs would you prefer for a 55-level CBC study on 13 attributes?

# Burn-in Draws	# Used Draws
10K	10K
20K	10K
30K	10K
40K	10K
10K	20K
20K	20K
30K	20K
20K	30K
10K	40K

If you thought that doing as many burn-in draws as you could was the most valuable thing, you'd choose the 40K/10K option. If sampling over as long a chain as you could was more important to you (you worried about long oscillation patterns), you'd choose the 10K/40K option. However, if it takes more than 10K iterations to arrive at a reasonably stable set of estimates, then something between those two extremes may work better.

Now for the answer, based on running 4 chains (HB runs) from different starting points:

# Burn-in Draws	# Used Draws	Maximum R-hat value for any of the 55 levels (target is for none to exceed 1.1)
10K	10K	1.28
20K	10K	1.32
30K	10K	1.31
40K	10K	1.34
10K	20K	1.10
20K	20K	1.13
30K	20K	1.17
20K	30K	1.10
10K	40K	1.08

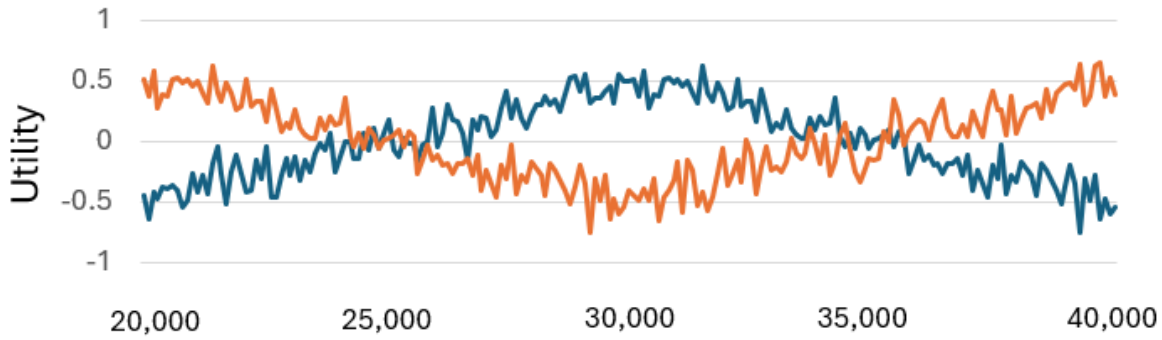
R-hat Values (Max over 55 levels) for a Complex CBC Data Set

For this particular dataset, sampling over at least 30K used draws is the way to acceptable convergence. It doesn't seem to matter for this complex dataset whether we use 10K or 20K burn-in iterations, so long as we are sampling across at least 30K used draws. (Other complex datasets may benefit from an initial 20K, 40K or more burn-in iterations.)

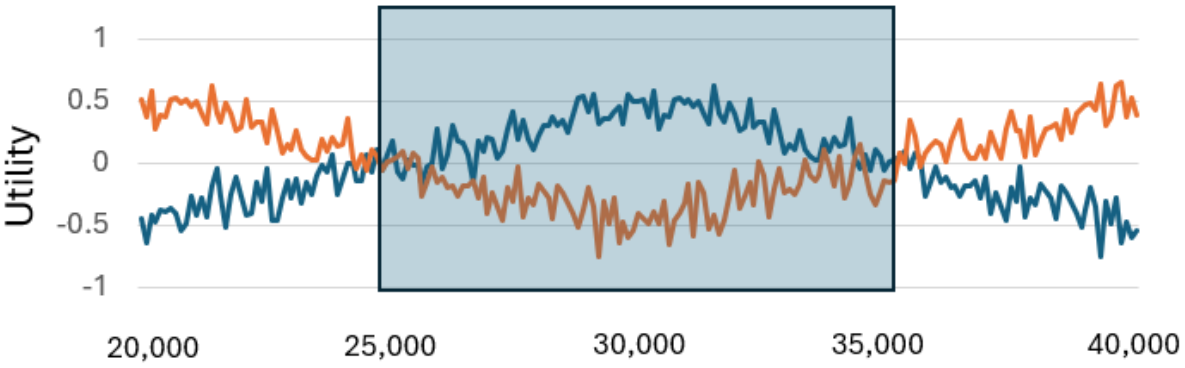
Why is it that sampling across a longer chain of used draws can improve convergence? One potential reason is that there may be long oscillation patterns that require a longer sampling interval to converge upon an unbiased mean posterior estimate for the sample.

Why It Can Be Safer to Sample across More Used Draws

For illustration, let's assume the case of long oscillation patterns. Imagine that we've run two independent HB chains from different random starting points (represented as the orange and blue utility proposals for a given attribute level). The resulting draws for iterations 20,000 through 40,000 are as follows:

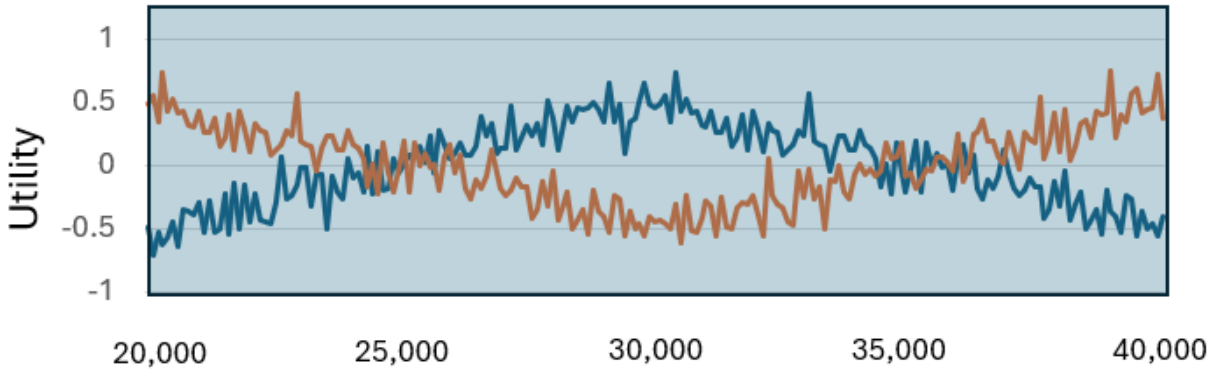


The unbiased mean population estimate for this attribute level is 0.00. But, there are long oscillation patterns that are out of sync for the two chains. Let's assume we made an unlucky decision: after the first 25K burn-in draws we sample over the next 10,000 draws. This interval:



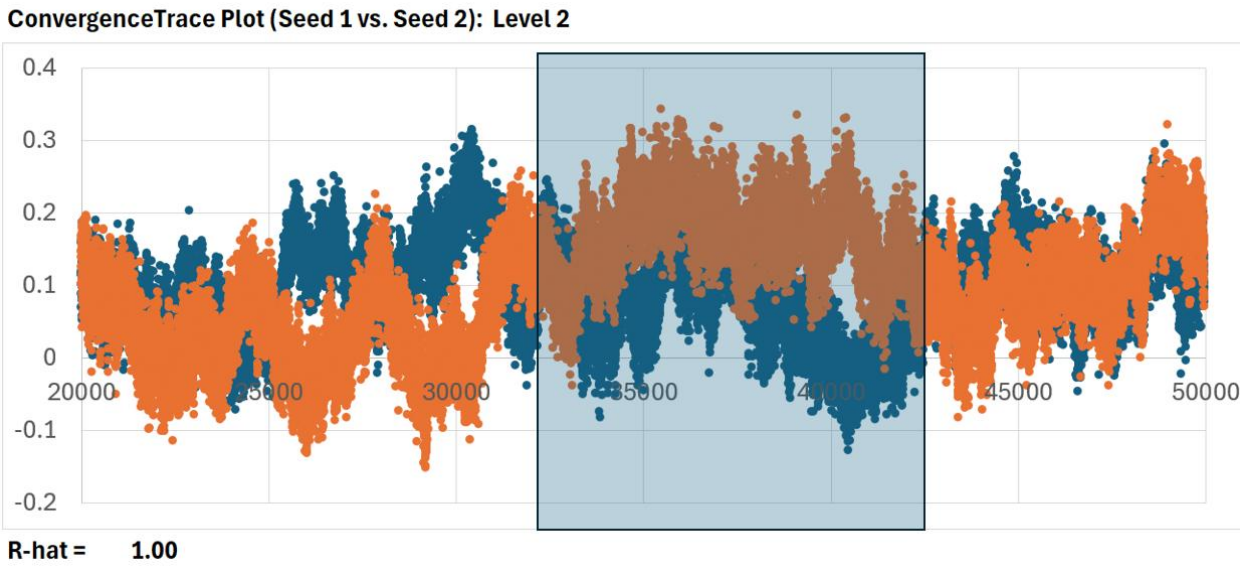
This would be an unfortunate choice for used draws, indeed. The mean of these draws for the blue chain is 0.28 and for the orange chain it's -0.26. The R-hat is 2.29. (Much larger than the 1.1 threshold.)

Alternatively, let's imagine we had decided that after the first 20,000 burn-in iterations we would start using draws from iteration 20,001 through 40,000. This interval:



This safer choice, to sample across 20,000 draws instead of 10,000, would lead to mean sampled draws of -0.01 for the blue chain and 0.03 for the orange chain (very near the true mean of 0.0), for an R-hat of 1.00, indicating convergence. (Well below the threshold of 1.1.)

A contrived and unrealistic example? Not entirely! Searching through the complex real CBC dataset with 55 levels I've been discussing, I was able to find long oscillations for some of the parameters, including the parameter I've graphed below. Below is a sequence of draws 20,001 through 50,000, exhibiting long oscillation patterns. The R-hat across these 30K used draws is 1.00, but if we had been unlucky to select 10,000 used draws from draw 32501 to 42500 the R-hat is 1.15, indicating lack of convergence.



These examples illustrate that we as analysts can get unlucky if there are long oscillation patterns in the draws and we choose 10K instead of 20K+ used iterations. Challenging data sets can exhibit longer oscillation patterns. It's safer to sample across more draws.

Simple View of Convergence

A simple and more intuitive way to assess convergence (in a cursory way) is by running correlations across the population mean utility estimates (collapsed across the used draws) for your four HB runs (your 4 chains). I've done so for the 10K/40K HB settings described above. I placed my population utility estimates in Excel as four columns and 55 rows. My correlations are all extremely high due to both the good convergence of the model and the fairly large sample size (n=1201).

	Chain 1	*Chain 2*	*Chain 3*	*Chain 4*
Chain 1	1			
Chain 2	0.999417	1		
Chain 3	0.999228	0.999207	1	
Chain 4	0.999129	0.999384	0.999329	1

Any one of these four chains of used draws works well for delivering my final model and market simulator.

I show this correlation table not because I recommend this practice for assessing convergence (using R-hat by separate level is better), but to show you how similar the mean results are across chains using different random seeds when you've obtained good convergence and with relatively large sample size. If you created a similar correlation table and your correlations weren't all near 1.0, you could XY scatter-plot the data and investigate further, level by level. If results are still concerning, run HB again with more burn-in and used iterations to try to improve convergence.

Is Lack of Convergence a Problem?

We have seen many data sets that fail to achieve formal convergence and yet consistently predict holdout choice tasks very well[1]. If the goal is predictive accuracy, then it wouldn't be malpractice to use non-converged HB runs *that predict well* in practice.

If the goal, however, is inference and statistical testing, not achieving convergence is a bad thing indeed. We should note that some complex data sets, such as with 100+ parameters to estimate, may both fail to converge *and* fail to predict well at the individual level (as reported by Kevin Lattery in his Turbo 2025 presentation). For such data sets, if individual-level prediction matters Kevin recommends using Stan with its HMC sampler. (Our experience is that with HB-MNL models and their resulting market simulators most researchers are concerned with population-level or segment-level predictions rather than individual-level predictions.)

[1] For example, with zero-centered utility specification (which Sawtooth traditionally uses), some mean estimated utilities are below -2, usually resulting in predictions very close to zero share of preference in a market simulator. Whether the chain has converged on -2.5 or -3.5 doesn't matter much at all, even though the R-hat may exceed 1.1.

Unless you work in the FMCG space, it's unlikely that you'll design and field a CBC study with more than 50 parameters to estimate. Most researchers in the vast majority of modeling situations are doing very well to use Sawtooth's HB routine (assuming you follow other best-practice principles such as few attribute prohibitions and reasonable sample sizes).

Guidelines for Practice

All else equal, the more iterations you can do in HB, the better. Though, at some point you've done plenty and you're just wasting time to continue iterating. Some academics prefer to do 100K/100K just to be safe. But, practitioners in the trenches like guidelines to streamline their work and protect against bad results.

I'd recommend the following for CBC and MaxDiff studies:

	Burn-in Iterations	**Used Iterations**
<30 parameters to estimate	20K	10K
30-50 parameters to estimate	25K	30K
50-100 parameters to estimate	40K	60K
>100 parameters to estimate*	50K	75K

*Consider whether aggregate logit (or a low-dimensional latent class MNL solution) would be suitable for the modeling task, such as often is the case with sparse MaxDiff or Bandit MaxDiff. If needing formal convergence or individual-level prediction accuracy, consider using Stan with HMC.

Appendix

R-hat involves straightforward algebra involving calculating the means of the chains and their variances for each part-worth utility estimate. There are many sources online showing the equations and code to do so.

A Wikipedia article https://en.wikipedia.org/wiki/Gelman-Rubin_statistic shows the equations (accessed May 2025):

J Monte Carlo simulations (chains) are started with different initial values. The samples from the respective burn-in phases are discarded. From the samples $x_1^{(j)}, \ldots, x_L^{(j)}$ (of the j-th simulation), the variance between the chains and the variance in the chains is estimated:

$$\overline{x}_j = \frac{1}{L}\sum_{i=1}^{L} x_i^{(j)}$$ Mean value of chain j

$$\overline{x}_* = \frac{1}{J}\sum_{j=1}^{J} \overline{x}_j$$ Mean of the means of all chains

$$B = \frac{L}{J-1}\sum_{j=1}^{J}(\overline{x}_j - \overline{x}_*)^2$$ Variance of the means of the chains

$$W = \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{L-1}\sum_{i=1}^{L}(x_i^{(j)} - \overline{x}_j)^2\right)$$ Averaged variances of the individual chains across all chains

An estimate of the Gelman-Rubin statistic R then results as[1]

$$R = \frac{\frac{L-1}{L}W + \frac{1}{L}B}{W}.$$

You can also access code online to implement R-hat. One such example I accessed (May 2025) from https://medium.com/@niall oulton/understanding-the-r-hat-statistic-d83b3b5ca162 included the following Python code:

```
def calculate_r_hat(samples):
    # Assuming samples is a 2D array of shape (num_chains,
num_samples)
    num_chains, num_samples = samples.shape

    # Calculate the within-chain variance
    W = np.mean(np.var(samples, axis=1, ddof=1))

    # Calculate the between-chain variance
    chain_means = np.mean(samples, axis=1)
    B = num_samples * np.var(chain_means, ddof=1)

    # Estimate the marginal posterior variance
    var_plus = ((num_samples - 1) / num_samples) * W + (1 /
num_samples) * B

    # Calculate R-hat
    r_hat = np.sqrt(var_plus / W)
    return r_hat
```

(Technical endnote: I realize I'm ignoring the ESS issue, which is needed to more completely assess the quality of the posterior distribution.)