Bayesian inference with Biogeme

Michel Bierlaire

December 20, 2025

Report TRANSP-OR xxxxxx
Transport and Mobility Laboratory
School of Architecture, Civil and Environmental Engineering
Ecole Polytechnique Fédérale de Lausanne
transp-or.epfl.ch

Series on Biogeme

Contents

The package Biogeme (`biogeme.epfl.ch`) is designed to estimate the parameters of various models. It is particularly designed for discrete choice models. Originally designed to use maximum likelihood estimation, it is also possible to use Bayesian inference to estimate the parameters of the model. It is particularly useful for mixtures models, as it allows to avoid the calculation of complex Monte-Carlo integrals.

We assume that the reader is already familiar with discrete choice models, and has successfully installed Biogeme. This document has been written using Biogeme 3.3.2.

It is also highly recommended to review foundational concepts such as simulation methods, Bayesian inference, and Markov chain Monte Carlo. Although these topics are briefly introduced here, a solid understanding of them greatly helps in fully appreciating the power and flexibility of Bayesian estimation for discrete choice models.

1 Bayesian inference

Bayesian inference consists of fitting a probabilistic model to observed data and representing the outcome by a probability distribution over the model parameters.

Bayesian inference differs fundamentally from frequentist inference in the way uncertainty about model parameters is represented and quantified. In the frequentist framework, parameters are treated as fixed but unknown constants, and uncertainty arises solely from the randomness of the data-generating process. In contrast, the Bayesian approach treats parameters as random variables endowed with a prior distribution, which encodes the information available before observing the data. This modeling choice does not imply that parameters are intrinsically random, but rather reflects epistemic uncertainty: the distribution represents our state of knowledge about plausible parameter values given the information at hand. After observing data, Bayes' theorem updates this prior into a posterior distribution, which synthesizes both prior beliefs and empirical evidence. The posterior distribution is therefore the central object of Bayesian inference, providing coherent measures of uncertainty, enabling probabilistic predictions, and allowing for direct probability statements about parameters themselves.

Consider a discrete choice model characterized by a vector of parameters θ and a likelihood function $L(\mathcal{D} \mid \theta)$, where $\mathcal{D}$ denotes the observed data: for each individual in the sample, it contains the values of the explanatory variables as well as the observed choice. The likelihood function represents the probability that the model, with parameters θ, reproduces exactly all the observations in the sample.

In the frequentist framework, estimation consists of finding a point estimate $\hat{\theta}$ that maximizes the likelihood or the log-likelihood. In the Bayesian framework, however, the parameters are treated as unknown quantities described by a prior density $p(\theta)$, reflecting the information available before observing the data.

Once the data are observed, inference is performed through Bayes' theorem, which combines the prior with the likelihood to obtain the posterior distribution of the parameters:

$$p(\theta \mid \mathcal{D}) = \frac{L(\mathcal{D} \mid \theta)\, p(\theta)}{\int L(\mathcal{D} \mid \theta')\, p(\theta')\, d\theta'}. \tag{1}$$

The denominator ensures that the posterior integrates to one.

The distribution (1) is too complex to be computed in closed form for realistic choice models. Consequently, we resort to simulation-based approximations. We briefly introduce the concept of simulation, first for simple distributions (Section 2) and then using Markov Chain Monte Carlo (Section 3).

2 Simulation and Monte Carlo approximation

The arithmetic of random variables can quickly become intricate. Even in the simple case of two independent random variables X and Y, with respective probability density functions (pdf) f_X and f_Y, the distribution of their sum is not straightforward. If we define $Z = X + Y$, the pdf of Z is obtained through a transformation known as *convolution*:

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(x)\, f_Y(z - x)\, dx.$$

For instance, assume that both X and Y follow a uniform distribution:

$$X \sim U(0,1), \qquad Y \sim U(0,1).$$

Then, the calculation of the convolution shows that Z follows a triangular distribution:

$$f_Z(z) = \begin{cases} 0, & z < 0, \\ z, & 0 \leqslant z \leqslant 1, \\ 2 - z, & 1 < z \leqslant 2, \\ 0, & z > 2. \end{cases}$$

However, in practice, the convolution integral rarely has a closed form, making it difficult to handle. The idea of simulation consists in generating concrete numerical values produced according to the probability law of the random variables of interest. Regular arithmetic can then be applied on those values, without the need to use complex transformations such as convolutions.

Let X be a random variable with probability density function (pdf) f_X. A *draw from* X is a numerical value obtained from a random mechanism whose outcomes follow exactly the distribution of X.

Formally, consider a sequence of independent draws $X_1, X_2, \ldots, X_R$ from X. For any fixed R, the empirical distribution of these draws can be represented by a histogram. As R becomes large, the histogram provides an increasingly accurate approximation of the true pdf f_X.

More precisely, for any interval $[a, b]$,

$$\frac{1}{R} \sum_{i=1}^{R} \mathbf{1}\{X_i \in [a, b]\} \xrightarrow[R \to \infty]{\text{a.s.}} \int_a^b f_X(x)\, dx, \tag{2}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function, and "a.s." stands for almost surely, meaning that the convergence holds with probability 1. This property demonstrates that the draws reproduce the probability structure of X: the relative frequency with which the draws fall in any region converges to the probability mass assigned to that region by the pdf f_X.

In this sense, a draw from X is not merely a number, but a realization generated according to f_X, and repeated draws allow us to recover the shape of the density through their empirical distribution.

This is illustrated in Figure 1, which displays histograms of 100'000 independent draws from two uniform random variables $X \sim U(0,1)$ and $Y \sim U(0,1)$, together with the histogram of their sum $Z = X + Y$. The first two panels show that the empirical distributions of X and Y closely match the flat density of the uniform distribution. The third panel presents the resulting distribution of Z, whose histogram approaches the theoretical triangular density obtained by the convolution of the two uniforms. This confirms that, as the number of draws increases, the simulated empirical distributions converge to their corresponding probability density functions. Here, no integral was involved in the calculation. Only simple additions of draws.

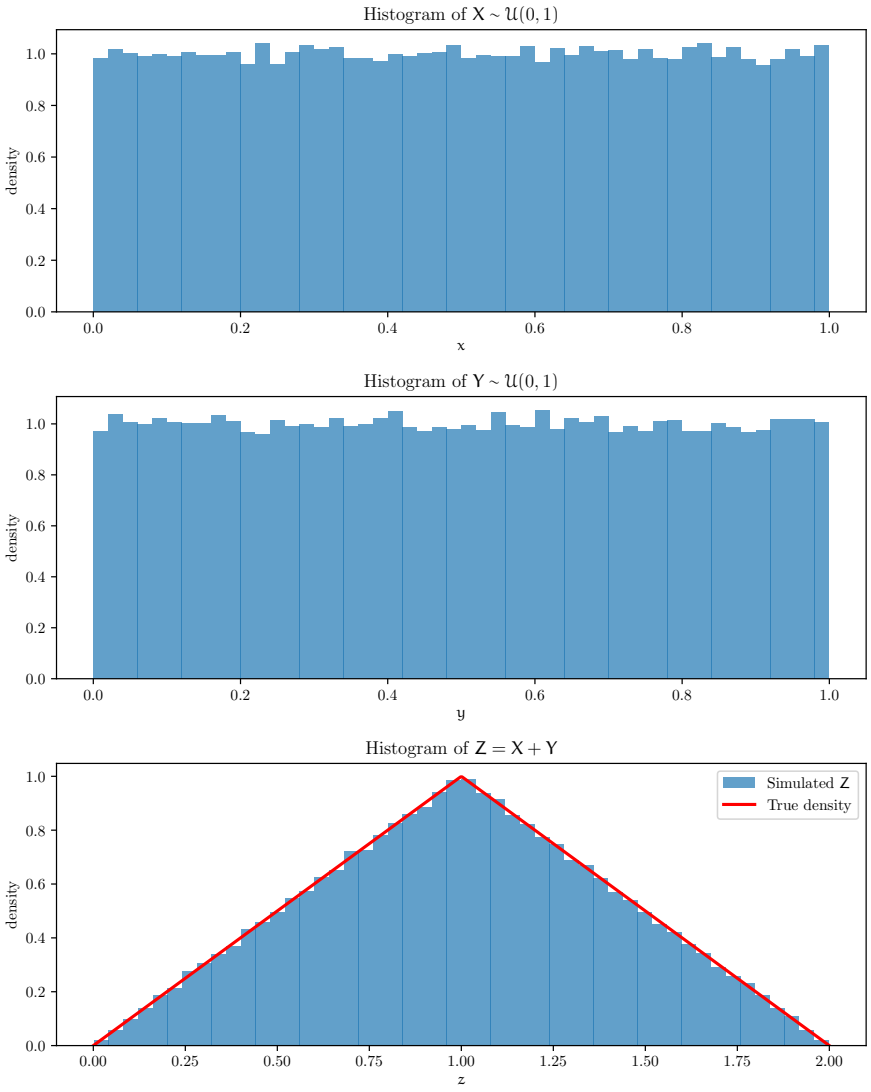

Figure 1: Histograms of X, Y, and $\mathsf{Z} = \mathsf{X} + \mathsf{Y}$ with theoretical density of Z.

In the Bayesian context, many quantities of interest are expectations under the posterior, such as $\mathbb{E}[\theta \mid \mathcal{D}]$ or predictive probabilities. Simulation plays the same role: Monte Carlo averages of draws from the posterior approximate these expectations without requiring analytical integration.

Most programming languages and numerical libraries provide a built-in function for generating draws from the uniform distribution $\mathrm{U}(0, 1)$. Although the sequence returned by such a function is deterministic (a pseudo-random number generator), it nonetheless exhibits all the statistical properties of a truly random sequence — in particular the convergence property (2). Once we have uniform draws, a variety of simple algorithms exist for transforming them into draws from other distributions (normal, extreme value, gamma, etc.) For a comprehensive treatment of these methods, the reader is referred to the standard text by Ross (2012).

Unfortunately, sampling from the posterior distribution (1) of the parameters of a choice model cannot be achieved through simple transformations of uniform draws. Instead, it requires more advanced simulation techniques, known as Markov chain Monte–Carlo (MCMC) methods. As those methods are the core of Bayesian inference, we provide a brief introduction in the next section. We invite the interested reader to consult the literature for a more comprehensive description (Wang, 2022, Gelman et al., 2014).

3 Markov Chain Monte Carlo (MCMC)

Bayesian estimation of discrete choice models requires working with posterior distributions that rarely have a closed-form expression. Markov Chain Monte Carlo (MCMC) methods provide a practical way to approximate such posteriors numerically: they generate a sequence of parameter values whose empirical distribution mimics the posterior.

The term *Monte-Carlo* refers to the city of Monte-Carlo in the Principality of Monaco, famous for its casino. In mathematics and statistics, "Monte-Carlo" is used whenever randomness is employed as a computational tool, typically to approximate integrals, expectations, or probability distributions.

In this section, we emphasize the main ideas behind MCMC and its role in Bayesian estimation. A detailed description of the numerical output, diagnostics, and practical advice for model interpretation is provided in Section 5.

3.1 From posterior to Markov chain

For choice models, the posterior (1) cannot be written in a simple analytic form, nor can it be sampled from directly.

An MCMC algorithm addresses this by constructing a *Markov chain*

$$\theta^{(1)}, \theta^{(2)}, \theta^{(3)}, \dots$$

such that, after a sufficiently long run, the distribution of $\theta^{(t)}$ stabilizes and coincides with the posterior $p(\theta \mid y)$. This limiting distribution is called the *stationary distribution* of the chain.

For this to hold, the chain must satisfy three mathematical properties:

- **Irreducibility**: it can eventually reach any region of the parameter space with positive posterior probability.
- **Aperiodicity**: it does not get trapped in deterministic cycles.
- **Positive recurrence**: it returns often enough to important regions of the space, so that a stable long-run distribution exists.

Modern samplers such as the No-U-Turn Sampler (NUTS), used by PyMC and therefore by Biogeme, are designed so that these conditions hold for a broad class of models (see Appendix B).

3.2 Burn-in, adaptation, and warm-up

The early iterations of an MCMC chain are not representative of the posterior. The chain typically starts from arbitrary initial values, far from the region where the posterior density is high, and must first "find" this region. This is illustrated on a simple example in Appendix A. In addition, advanced samplers automatically tune internal parameters (step sizes, mass matrices, etc.) in the first phase of the run.

Conceptually, one can distinguish:

Burn-in the period during which the chain moves from its initial position to the typical region of the posterior;

Adaptation the period during which the algorithm adjusts its own tuning parameters.

Draws produced during these phases do not follow the stationary distribution and are discarded. In practice, these phases are merged into a single warm-up period that PyMC and Biogeme handle automatically. The user only needs to be aware that early iterations are meant for tuning, not for inference. Only the draws produced after warm-up are used for inference and diagnostics.

3.3 Why multiple chains?

Instead of running a single long chain, it is standard practice to run several independent chains (for instance, four), each starting from a different initial value. If all chains target the same posterior distribution and have run long enough:

- they should explore the same region of the parameter space,
- their trajectories should look similar (after burn-in),
- and their empirical distributions should agree.

Comparing chains is a powerful way to detect non-convergence or poor exploration. The formal diagnostics (such as $\widehat{\mathsf{R}}$ and effective sample sizes) and their interpretation are discussed in Section 5.

3.4 Autocorrelation and effective information

Unlike independent Monte-Carlo draws, MCMC samples are correlated: $\theta^{(t)}$ and $\theta^{(t+1)}$ tend to be similar. This *autocorrelation* means that N MCMC draws do not contain as much information as N independent draws would. Intuitively, if the chain moves slowly, many consecutive draws look alike and bring little new information.

To account for this, one often reports an *effective sample size* (ESS), which answers the question: "How many independent draws would contain the same information as the correlated draws produced by the Markov chain?" High autocorrelation implies a smaller ESS for a given number of iterations. The precise formulas and thresholds used in Biogeme are described later, in Section 5.

A larger effective sample size implies smaller Monte Carlo error for posterior summaries; for example, the Monte Carlo standard error of a posterior mean scales approximately as $1/\sqrt{\text{ESS}}$.

3.5 From chains to posterior summaries

Once the chains have converged and a sufficiently large number of post-burn-in draws is available, the posterior distribution can be approximated by the empirical distribution of these draws. Any posterior quantity of interest can then be estimated by Monte-Carlo averages:

- posterior means and modes of parameters (e.g., taste coefficients, scale parameters);
- posterior uncertainties (standard deviations, credible intervals, Highest Density Intervals);
- posterior predictive quantities (choice probabilities, elasticities, value of time, etc.);
- model comparison metrics such as WAIC and LOO.

The next section (*Interpreting the Bayesian Estimation Output in Biogeme*) explains which summaries are reported by Biogeme, how they are computed from the chains, and how they should be used in practice to assess convergence, goodness-of-fit, and the reliability of the estimated choice model.

4 Model specification in Biogeme

The specification of the choice models in Biogeme for Bayesian estimation is very similar as the specification of the models for maximum likelihood estimation, with some important differences.

4.1 Prior distributions

Biogeme allows the user to assign a Bayesian prior distribution to any parameter defined with the `Beta` class. A prior is specified by providing a user-defined *prior factory*, that is, a Python callable following a simple protocol: it receives the PyMC variable name of the parameter, the initial value supplied in the Biogeme model, and (when applicable) the lower and upper bounds imposed on the parameter. The function must return a valid PyMC distribution object, which Biogeme will use as the prior in the Bayesian estimation. Typical applications include enforcing sign constraints or expressing expert knowledge about plausible parameter magnitudes. If no user-defined prior is supplied, Biogeme automatically constructs a default prior: an unbounded Normal distribution centered at the initial value if no bounds are provided, or a truncated Normal distribution whenever lower or upper bounds are specified. Custom priors, such as heavy-tailed distributions, asymmetric shapes, or sign-restricted distributions, can easily be implemented by defining a suitable prior factory and passing it as an argument when declaring the parameter.

The example below defines a prior factory that constructs a Student-t distribution (allowing for heavier tails than a Gaussian prior) and truncates it so that the parameter can only take negative values. The function returns a PyMC distribution object that Biogeme will use when building the Bayesian model. Once defined, this prior can be supplied directly to a `Beta` parameter through its `prior` argument.

```
import pymc as pm
from pytensor.tensor import TensorVariable

def negative_student_prior(
    name: str,
    initial_value: float,
    lower_bound: float | None,
    upper_bound: float | None,
) -> TensorVariable:
    base = pm.StudentT.dist(mu=0.0, sigma=10.0, nu=5.0)
    # Lower bound ignored; enforce upper bound at 0.
    upper = 0.0 if upper_bound is None else min(0.0, upper_bound)
    return pm.Truncated(
        name=name,
        dist=base,
        upper=upper,
        initval=initial_value,
    )
```

The prior is then passed to a Biogeme parameter as follows:

```
b_cost = Beta(
    'b_cost',
    value=-1.0,
    lowerbound=None,
    upperbound=None,
    status=0,
    prior=negative_student_prior,
)
```

In most applications, the default priors used by Biogeme — a Normal distribution (when no bounds are specified) or a Truncated Normal distribution (when lower or upper bounds are provided) — are entirely adequate and require no user intervention. Custom priors should be introduced only when specific prior knowledge, such as heavy-tailed behavior, is substantively justified.

Note that it is generally not recommended to use uniform priors for parameters in models estimated with NUTS. Uniform distributions have flat density over their support and do not provide meaningful curvature for the Hamiltonian dynamics. As a result, they often lead to poor exploration of the posterior, divergent transitions, and slow or unstable convergence.

4.2 Random coefficients and Monte-Carlo integration

A major practical advantage of Bayesian estimation in Biogeme is that *mixture models do not require any numerical integration over random parameters*. This stands in sharp contrast with classical maximum likelihood estimation, where mixtures of logit models involve integrals which do not admit closed-form solutions, and require Monte–Carlo integration.

Under Bayesian estimation, the random parameters themselves are treated as *latent variables*. Instead of integrating them out, the MCMC sampler *simulates* them jointly with the structural parameters. Conditional on a specific draw of the random parameters, the contribution of each observation is simply the log-likelihood of the elementary model (typically a logit). The integral no longer appears explicitly. MCMC approximates it automatically by randomly sampling from the posterior distribution of the latent coefficients.

- It eliminates the need for high-dimensional numerical integration.
- It avoids Monte–Carlo noise in the likelihood, which improves numerical stability and often speeds up convergence.
- It allows complex hierarchical and latent-structure models to be estimated with essentially no additional analytical effort.

Because the sampler takes care of integrating over the random parameters, the analyst must provide only the *conditional* log-likelihood, that is, the log likelihood of the model *given* a specific realization of the random coefficients. In most discrete choice settings, this is simply the log of a logit probability. There is no need to write a mixed logit formula or to manually integrate over the mixing distribution.

This simplification is one of the most important practical benefits of Bayesian estimation of mixture models: *the algorithm handles all the integration implicitly, while the user works with only the elementary likelihood.*

Some more details about how it is actually done are available in Appendix C.

In many applications, it is useful to retain the simulated draws of random coefficients in the estimation output, for instance to analyze heterogeneity, compute individual-level effects, or perform post-estimation simulations.

The following instruction tells Biogeme to explicitly store the draws of a random coefficient as part of the estimation results:

```
b_time_rnd = DistributedParameter('b_time_rnd', b_time + b_time_s * b_time_eps)
```

Here, b_time_rnd represents the individual-specific realization of the time coefficient, constructed from its mean, scale, and random disturbance, and stored for later inspection.

4.3 Scale parameters and identification

When estimating a mixture of logit models by maximum likelihood, the scale of a normally distributed random coefficient is identified only up to its absolute value: because the normal distribution is symmetric, a positive or negative estimate of the scale parameter yields exactly the same likelihood. For this reason, analysts sometimes allow the scale parameter to be unconstrained during optimization, giving the numerical algorithm more freedom to move in the parameter space.

In Bayesian estimation, however, such unconstrained specification is highly problematic. If the prior does not enforce a positivity constraint on the scale parameter, the sampler will explore both the positive and negative regions of the parameter space, even though these correspond to the *same* underlying model. As a consequence, the posterior distribution becomes artificially bimodal, with symmetric modes around zero and a mean tending toward zero, even when the true scale is large. This phenomenon severely degrades mixing, inflates posterior uncertainty, and can render posterior summaries essentially meaningless. Therefore, it is *critical* to impose a positivity constraint when defining the prior of any scale parameter or truncated distribution). This ensures that the sampler explores only the meaningful region of the parameter space and avoids the spurious bimodality that would otherwise arise.

In practice, enforcing the required positivity constraint for scale parameters is straightforward in Biogeme. The `Beta` constructor includes optional `lowerbound` and `upperbound` arguments, which can be used to restrict the parameter domain. To ensure that a scale parameter remains strictly positive during Bayesian estimation, one simply specifies a small positive lower bound, for example `lowerbound = 1e-6`. This prevents the sampler from crossing zero, thereby eliminating the artificial sign ambiguity and ensuring that the posterior distribution remains unimodal and interpretable. Such a small bound has no practical impact on the model but is essential for stable and meaningful Bayesian inference.

4.4 Mixtures with panel data.

A major advantage of Bayesian estimation becomes particularly clear in the context of mixture models with panel data. Under maximum likelihood estimation, the likelihood contribution of an individual must account for the full sequence of choices in the panel. This requires computing the joint probability of the entire trajectory, conditional on the random coefficients, and then performing Monte-Carlo integration over the distribution of these coefficients.

In the Bayesian framework, the situation is much simpler. Biogeme treats the random parameters as explicit latent variables that are sampled by the MCMC algorithm, and therefore *no Monte-Carlo integration is required*. Just as importantly, when the data are organized in panels, Biogeme assumes that the random parameters are drawn *per individual*, not per observation. This means that all observations belonging to the same person share the same realization of the random coefficients.

From the user's perspective, this structure greatly simplifies the model definition. Conditional on the random parameters, the contribution of a *single* observation is simply the logit kernel:

```
log_probability_one_observation = loglogit(v, av, CHOICE)
```

Biogeme automatically aggregates these contributions over the observations belonging to the same panel and handles the sampling of the random coefficients internally. Thus, the analyst only needs to specify the per-observation likelihood, and Biogeme takes care of both the panel

structure and the integration over the random parameters. This is considerably simpler than maximum likelihood estimation, while providing a fully coherent Bayesian treatment of random heterogeneity.

Note that, although the specification itself becomes simpler, mixture models with panel data remain computationally demanding. Because the posterior distribution of random coefficients can be complex and highly correlated, these models typically require:

- substantially longer estimation times than simple logit models,
- more MCMC iterations to ensure good mixing,
- and careful monitoring of convergence diagnostics.

Users should therefore anticipate heavier computational requirements, especially when estimating models with many random parameters or long individual trajectories.

4.5 Simulation after Bayesian estimation

Once the posterior distribution of the model parameters has been obtained, Biogeme offers two complementary approaches for conducting simulations. These correspond to two different interpretations of uncertainty and allow the analyst either to work with a single representative parameter vector or to propagate full posterior uncertainty into the simulated quantities.

Simulation using posterior means. The first approach consists of replacing each parameter by its posterior mean and running a deterministic simulation of the model. In Biogeme, this is performed by providing the vector of posterior means to the simulator:

```
results = biosim.simulate(the_beta_values=betas)
```

This corresponds to evaluating the model at a single point estimate, analogous to using maximum likelihood estimates in classical estimation. It is simple, fast, and produces easily interpretable results. However, it does not reflect the uncertainty contained in the posterior distribution: all variability across draws is ignored. This method is appropriate when a single "typical" prediction is desired.

Bayesian simulation using posterior draws. The second approach incorporates full parameter uncertainty by repeatedly simulating the model for multiple posterior draws:

```
bayesian_results = biosim.simulate_bayesian(
    bayesian_estimation_results=estimation_results,
    percentage_of_draws_to_use=3
)
```

Here, Biogeme samples a subset of the posterior draws (e.g., 3% of them) and computes the simulation for each draw. The final output is therefore a distribution of simulated quantities, not a single value. This method fully reflects posterior uncertainty and is essential when one wishes to obtain prediction intervals, Bayesian credible regions, or a quantification of the robustness of policy scenarios.

In summary, simulation using posterior means yields a single-point prediction, while Bayesian simulation propagates the entire uncertainty about the parameters into the results. Both approaches are useful, and the choice between them depends on whether the analysis calls for deterministic predictions or uncertainty-aware decision support.

5 Interpreting the Bayesian Estimation Output in Biogeme

Biogeme performs Bayesian estimation by relying internally on the PyMC probabilistic programming library. The output produced by Biogeme therefore mirrors the standard PyMC diagnostics while presenting them in a unified report. This section provides a short explanation of the quantities that appear in Biogeme's Bayesian output, together with practical tips for interpretation. We refer the reader to the online documentation of PyMC and ArviZ, as well as Vehtari et al. (2016), Watanabe (2010) for more information.

5.1 General Information About the Estimation

Sample size. Number of observations used in the estimation.

Sampler. Biogeme automatically chooses an appropriate MCMC sampling method based on the computational resources available on the machine. When the JAX and NumPyro libraries are installed, Biogeme prefers the NumPyro implementation of the No-U-Turn Sampler (NUTS), which can exploit modern accelerators. If multiple computational devices (such as GPUs or TPUs) are detected, Biogeme uses a *parallel* sampling strategy, running one Markov chain on each device. If a single device is available, Biogeme instead uses a *vectorized* strategy in which all chains are processed simultaneously on the same device, enabling efficient batched computation.

If JAX or NumPyro is not available, Biogeme falls back to the standard PyMC implementation of NUTS, running on the CPU and typically parallelizing chains across the available CPU cores. This approach is robust and ensures reproducible results even in environments without specialized hardware. Although the automatic selection mechanism generally provides the most efficient configuration, users may also specify the sampling method manually when needed, for example to force CPU-based sampling or to explicitly select NumPyro's parallel or vectorized modes.

Number of chains. Biogeme runs multiple independent MCMC chains (usually 4). Agreement between these chains is essential for diagnosing convergence.

Number of draws per chain. Number of samples retained from each chain after any warm-up phase. The total number of draws equals "(draws per chain) × (number of chains)." More draws improve precision but do not fix convergence issues if chains do not mix properly.

Acceptance rate target. The NUTS sampler adapts to reach a target acceptance rate (typically between 0.8 and 0.95). If the adaptive procedure cannot achieve this value, it may indicate difficult posterior geometry (e.g., strong correlations, heavy tails).

Run time. Total wall-clock time used to obtain the posterior sample.

5.2 Posterior Log-Likelihood and Predictive Fit

Biogeme computes several quantities evaluating how well the estimated parameters explain the observed data.

Posterior predictive log-likelihood. An approximation of

$$\log \mathfrak{p}(\mathsf{Y} \mid \mathbb{E}[\theta]),$$

that is, the log-likelihood evaluated at the posterior mean. Higher (less negative) values indicate better fit. This is similar in spirit to evaluating the log-likelihood at MLEs in frequentist estimation.

Expected log-likelihood. Biogeme averages the log-likelihood across all posterior draws:

$$\mathbb{E}[\log L(Y \mid \theta)] .$$

This reflects the model's typical predictive performance across the entire posterior, not just at a single representative point.

Best-draw log-likelihood. The highest log-likelihood found among all draws. It provides an upper bound on predictive fit under the estimated posterior.

Practical advice: If the posterior predictive log-likelihood and the expected log-likelihood differ substantially, the posterior may be wide or skewed, or the model could be sensitive to specific parameter values.

5.3 Information Criteria for Model Comparison

Biogeme computes WAIC and LOO using PyMC's built-in functions.

WAIC (Widely Applicable Information Criterion). A fully Bayesian generalization of the Akaike Information Criterion (AIC). WAIC estimates expected predictive performance on new data. Lower values indicate a better model.

Effective number of parameters $p_{\mathbf{WAIC}}$. This quantity measures the degree of flexibility that the model displays when fitting the data, as inferred from the variability of the log-likelihood across the posterior distribution. If p_{WAIC} is much larger than the number of free parameters, the model may be overfitting.

LOO (Pareto-smoothed Leave-One-Out cross-validation). A Bayesian approximation of leave-one-out predictive performance using importance sampling. As with WAIC, lower values indicate better predictive accuracy.

Standard errors. Both WAIC and LOO come with standard errors, useful for comparing two models. If the difference in criteria is smaller than twice the standard error, the evidence does not favor one model over the other.

Practical advice: For Bayesian model selection in Biogeme, LOO is typically the most robust criterion, especially for models with latent variables or hierarchical structures.

5.4 Posterior Parameter Summaries

For each parameter θ, Biogeme reports statistics derived from the posterior draws.

Name. Identifier of the parameter.

Value (Posterior mean). Expected value under the posterior distribution. This is often used as the "Bayesian point estimate."

Median. Posterior median, that is, the value such that half of the posterior mass lies below and half above. The median is a robust measure of central tendency and can differ substantially from the mean when the posterior distribution is skewed.

Mode. Posterior mode, computed from kernel density estimation. Useful when the posterior is skewed or multimodal.

Std err. Posterior standard deviation, measuring uncertainty around the mean.

z-value. Mean divided by standard deviation. Large absolute values signal that the posterior mass is far from zero.

p-value. Two-sided posterior probability that the parameter has the opposite sign from its mean. Extremely small values (e.g. < 0.01) indicate high confidence in the sign of the effect.

HDI (Highest Density Interval). Biogeme provides the lower and upper bounds of the highest-density interval (typically 95%). This interval contains the most probable values of the parameter.

Practical interpretation: If the HDI includes zero, the effect is not credibly different from zero. Unlike frequentist confidence intervals, HDIs directly correspond to probability statements about the parameter.

5.5 Identification diagnostics

This section provides numerical diagnostics intended to detect non-identification or weak identification of model parameters. Intuitively, identification problems arise when some linear combinations of parameters can vary substantially without affecting the likelihood, leading to a very flat (or nearly flat) posterior in certain directions. The diagnostics are based on the posterior draws and, when available, on the prior draws.

Posterior covariance diagnostics. The posterior covariance matrix summarizes the dispersion of the posterior distribution in parameter space. Its eigenstructure is particularly informative.

- *Minimum and maximum eigenvalues.* The eigenvalues measure posterior variance along orthogonal directions. A very *large* eigenvalue corresponds to a very wide (nearly flat) direction of the posterior, indicating weak or non-identification along a linear combination of parameters. Conversely, a very small eigenvalue indicates a tightly concentrated, well-identified direction.
- *Condition number.* Defined as the ratio of the largest to the smallest eigenvalue, it measures the anisotropy of the posterior covariance. Large values indicate near-dependencies among parameters. As a rule of thumb, values around 10^3 deserve attention, while values of 10^5 or more are a strong warning sign.
- *Effective rank.* This quantity can be interpreted as the effective dimensionality of the posterior variability (between 0 and the number of parameters). If it is much smaller than the total number of parameters, the posterior mass concentrates in a lower-dimensional subspace, which is consistent with (near) linear dependencies among parameters.

Prior covariance diagnostics. The same diagnostics are reported for the prior distribution. They provide a reference scale: if the prior covariance is well behaved (full rank, moderate condition number) but the posterior covariance becomes ill-conditioned, the identification issue is typically due to the likelihood or the model specification rather than the prior.

Identified by the prior. When prior draws are available, Biogeme compares prior and posterior dispersion at the parameter level. For each parameter, the ratio of the posterior standard deviation to the prior standard deviation is reported.

- A ratio close to 1 indicates that the data have not substantially reduced uncertainty, suggesting that the likelihood is weakly informative for that parameter.

- A ratio well below 1 (for example 0.1 or 0.01) indicates that the data are informative and that the parameter is well identified by the likelihood.

5.6 Convergence Diagnostics

Biogeme reports standard PyMC diagnostics:

$\widehat{\mathsf{R}}$ **(Gelman–Rubin statistic).** Values close to 1 indicate that chains are mixing well. Threshold: $\widehat{\mathsf{R}} \leqslant 1.01$ is generally considered good.

ESS (bulk). Effective sample size for the main mass of the posterior. Values above 400 indicate reliable estimation of means and variances.

ESS (tail). Effective sample size for the tails. Values above 100 are necessary for stable estimation of extreme quantiles.

Practical advice. If $\widehat{\mathsf{R}} > 1.01$ or ESS values are low, the sampler did not explore the posterior well. Consider:

- reparameterizing the model,
- increasing the number of tuning steps,
- checking for strong correlations or identifiability issues.

5.7 Simulated quantities

This section lists all quantities stored in the simulation output file (`.nc`), together with their dimensions and shapes. These objects correspond to the contents of the `InferenceData` structure produced by PyMC and used internally by Biogeme for diagnostics and reporting.

Each quantity is identified by a *group*, a *variable name*, its *dimensions*, and its *shape*. The most important groups are described below.

constant_data. Observed data passed to the model and treated as fixed. Each variable is indexed by the observation dimension (`Dimension.OBS`) and has length equal to the sample size. These variables are not random and are included for completeness and traceability of the estimation.

posterior. Posterior draws of model parameters and derived quantities. For scalar parameters (e.g. alternative-specific constants or taste coefficients), the dimensions are `(chain, draw)`. Observation-level quantities, such as the pointwise log-likelihood, add the observation dimension `Dimension.OBS`. These draws are the basis for posterior summaries, credible intervals, and identification diagnostics.

prior. Prior draws for the same parameters and derived quantities. These are generated independently of the data and are stored when prior sampling is enabled. Comparing prior and posterior dispersion helps assess whether parameters are identified by the data or mainly constrained by the prior.

log_likelihood. Pointwise log-likelihood contributions by observation, chain, and draw. These quantities are used for model comparison and diagnostics (e.g. WAIC, LOO), and allow inspection of how individual observations contribute to the fit.

sample_stats. Sampler diagnostics produced by the MCMC algorithm. They include acceptance rates, step sizes, number of leapfrog steps, tree depth, energy, and divergence indicators. These statistics are essential to assess the numerical reliability of the simulation and to detect pathologies such as poor adaptation or divergent transitions.

5.8 Graphical Diagnostics

Biogeme displays several plots produced by PyMC:

Trace plots. Show per-chain evolution of draws. *Good:* overlapping chains with no visible drift. *Bad:* chains stuck at different levels or showing trends.

Rank plots. Rank-normalized distributions across chains. Close agreement across chains indicates good mixing.

Energy plots (NUTS). Show the Hamiltonian energy distribution. Separated energy distributions or low BFMI signal poor exploration.

Autocorrelation plots. Show lag dependence. Fast decay indicates good mixing; slow decay suggests high parameter correlation.

Practical advice: Even when numerical diagnostics look good, graphical verification is essential. Graphs often reveal subtle issues (e.g., multimodality, slow transitions) that summary statistics cannot detect.

A Example of the simulation of a Markov chain

We illustrate the notion of stationary and time-reversible Markov chains with a simple example involving customer engagement on an online service (e.g., a subscription-based platform).

We consider a single user observed once per day. On any given day, the user is in exactly one of the following three engagement states:

- State 1: low engagement (rarely logs in, uses very few features),
- State 2: medium engagement (uses the service somewhat regularly),
- State 3: high engagement (uses the service intensively and frequently).

We assume that the evolution of the user's engagement from day to day can be modeled as a homogeneous Markov chain $(X_t)_{t\geqslant 0}$ taking values in $\{1, 2, 3\}$, with the following transition matrix:

$$P = \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.5 & 0.3 \\ 0.1 & 0.3 & 0.6 \end{pmatrix}.$$

Each entry P_{ij} denotes the probability that the user moves from state i on day t to state j on day $t+1$. The entries of P can be read as follows:

- From low engagement (state 1):
  - the user stays low-engagement the next day with probability 0.7,
  - moves up to medium engagement with probability 0.2,
  - jumps directly to high engagement with probability 0.1.
- From medium engagement (state 2):
  - the user drops to low engagement with probability 0.2,
  - remains at medium engagement with probability 0.5,
  - increases to high engagement with probability 0.3.
- From high engagement (state 3):
  - the user cools down to medium engagement with probability 0.3,
  - remains highly engaged with probability 0.6,
  - drops directly to low engagement with probability 0.1.

It is easy to verify that the Markov chain admits the uniform stationary distribution

$$\pi = (\pi_1, \pi_2, \pi_3) = \left(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right).$$

The Markov chain is also time-reversible with respect to π. This follows from the fact that $\pi_i = \pi_j = 1/3$ for all i, j, and that P is symmetric

The behavior of the Markov chain introduced above is illustrated in Figure 2. The figure displays the empirical frequency of each state as the simulation evolves over time. At the beginning of the run, these empirical frequencies fluctuate widely and do not yet reflect the target distribution. As the number of iterations increases, however, the proportions stabilize and gradually approach the theoretical stationary distribution $\pi = (1/3, 1/3, 1/3)$ derived earlier. A crucial practical implication is that the draws generated during the early iterations—before the chain has approached stationarity—should not be used as representative samples from the target distribution. Only after the chain has "settled" near equilibrium do the simulated states behave as valid draws from the desired stationary distribution. Typically, in this example, we

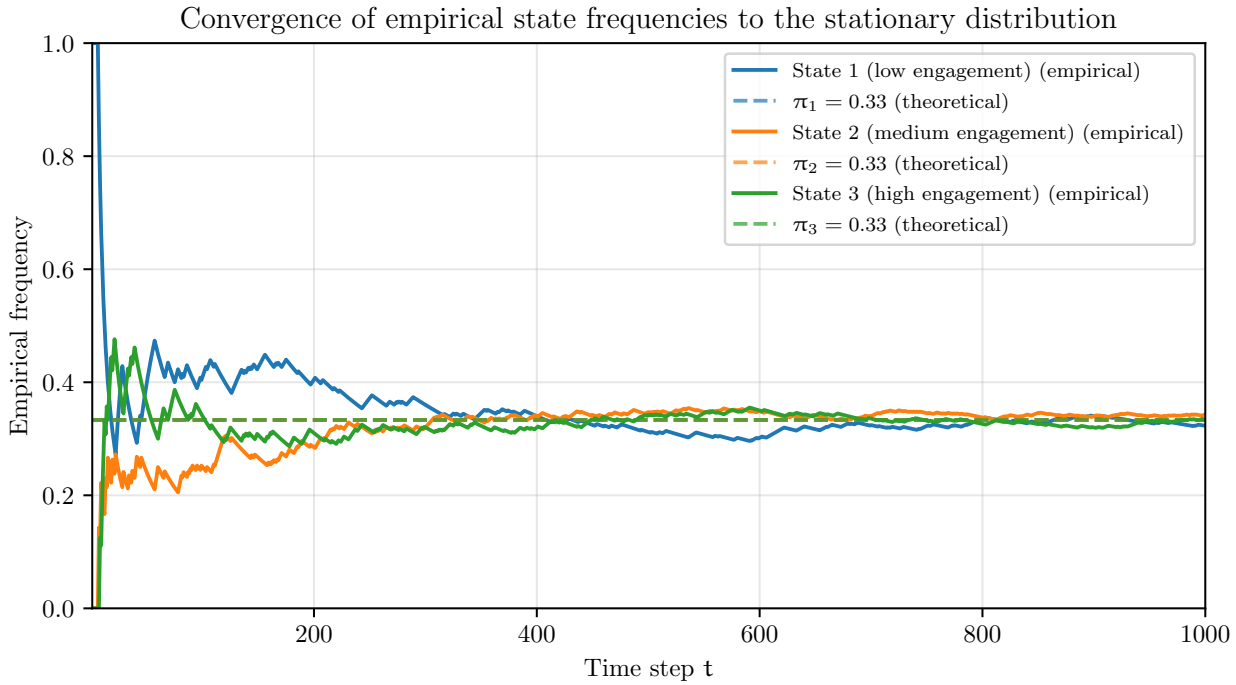

Figure 2: Simulation of the three–state Markov chain (t=0,. . . , 1000)

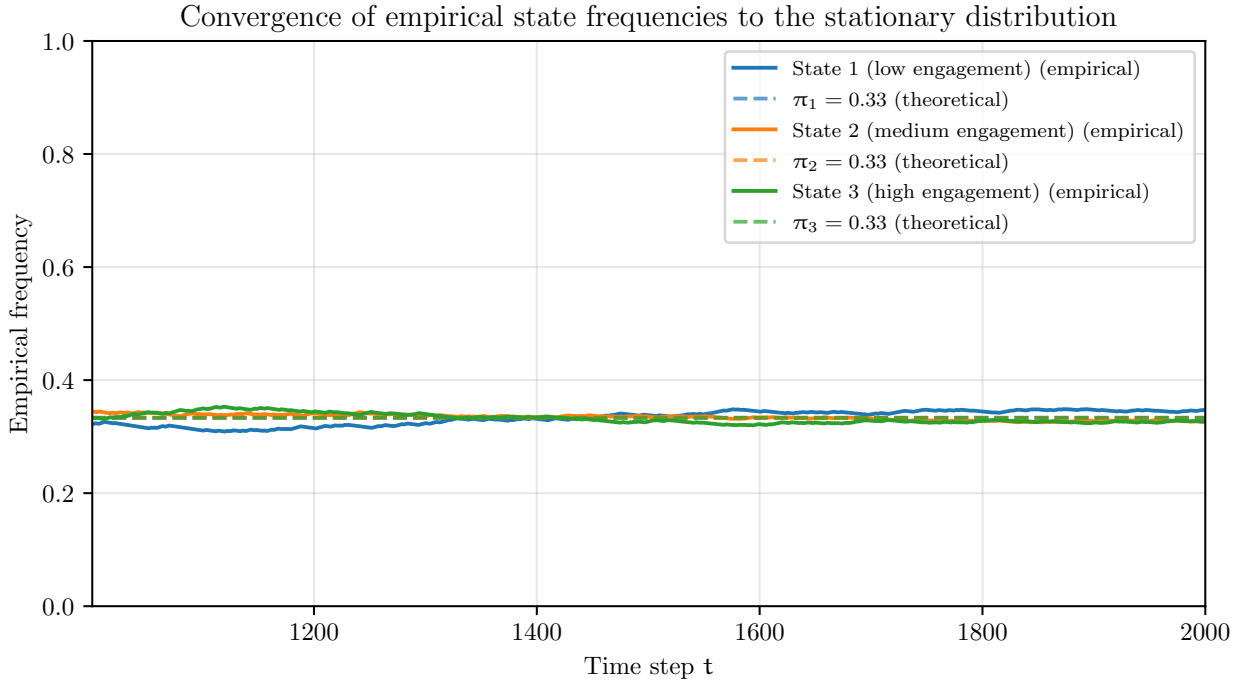

Figure 3: Simulation of the three–state Markov chain (t=1000,. . . , 2000)

would simply discard all the 1000 draws displayed in Figure 2, and start using the chain to generate more draws (Figure 3 illustrates the chain from step 1000 to step 2000).

Although this example is purely illustrative, it captures the key idea of MCMC: by simulating a Markov chain whose stationary distribution is the target distribution, the empirical frequencies of the chain approximate that distribution as the number of iterations increases.

B Hamiltonian Monte Carlo

PyMC relies on *Hamiltonian Monte Carlo* (HMC) and its adaptive variant, the *No-U-Turn Sampler* (NUTS), to generate draws from the posterior distribution of the model parameters. This section provides a conceptual description of the algorithm, assuming that the reader is already familiar with the Metropolis–Hastings (MH) algorithm and the idea of proposing candidate states through a Markov chain with an accept–reject mechanism.

B.1 Target distribution

Let $\theta \in \mathbb{R}^d$ denote the vector of model parameters, and let

$$f(\theta) \propto p(\theta \mid \mathcal{D})$$

be the posterior density given the observed data $\mathcal{D}$. As in standard MH, the goal is to construct a Markov chain whose stationary distribution is $f(\theta)$.

The main difficulty addressed by HMC is that, in high-dimensional models, simple random-walk proposals lead to very slow exploration of the parameter space. HMC overcomes this limitation by exploiting the *geometry* of the posterior distribution through its derivatives.

B.2 Augmenting the state space: momentum variables

The key idea of HMC is to introduce an auxiliary *momentum variable*

$$p \in \mathbb{R}^d,$$

and to define a joint density over (θ, p):

$$\pi(\theta, p) = f(\theta)\, \mathcal{N}(p \mid 0, M), \tag{3}$$

where M is a symmetric positive definite *mass matrix*.

The momentum variable has no direct statistical interpretation; it is an algorithmic device inspired by classical mechanics. Its role is to allow the sampler to generate coherent, directed moves across the parameter space, rather than diffusive random walks.

B.3 Hamiltonian formulation

Define the *potential energy*

$$U(\theta) = -\log f(\theta),$$

and the *kinetic energy*

$$K(p) = \frac{1}{2} p^\top M^{-1} p.$$

The sum

$$H(\theta, p) = U(\theta) + K(p) \tag{4}$$

is called the *Hamiltonian*. With this definition, the joint density (3) can be written as

$$\pi(\theta, p) \propto \exp\big(-H(\theta, p)\big).$$

Sampling from the posterior $f(\theta)$ can therefore be achieved by sampling from $\pi(\theta, p)$ and discarding the momentum variable.

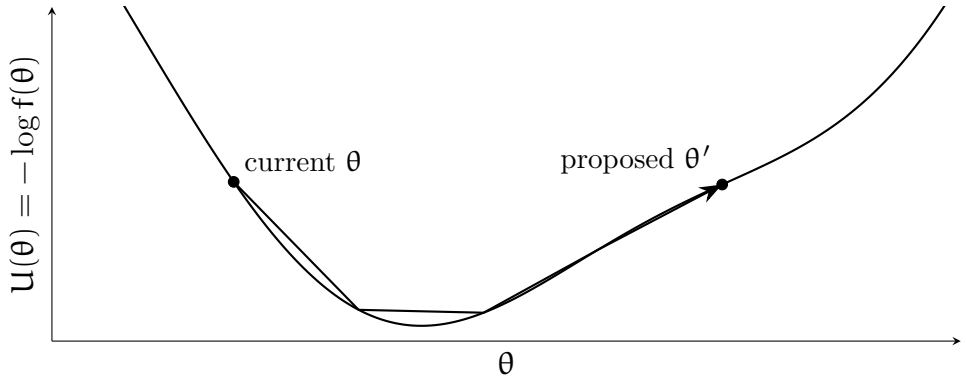

Figure 4: Illustration of the HMC intuition in one dimension. The posterior density $f(\theta)$ is converted into a potential energy $U(\theta) = -\log f(\theta)$. Hamiltonian dynamics use gradients of $U(\theta)$ (equivalently, of $\log f(\theta)$) to propose long, coherent moves from a current value θ to a distant candidate θ', avoiding the diffusive behavior of random-walk proposals.

B.4 Hamiltonian dynamics and the proposal mechanism

HMC generates proposals by simulating *Hamiltonian dynamics*, defined by the system of differential equations

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial p} = \mathbf{M}^{-1}\mathbf{p}, \tag{5}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial \theta} = -\nabla_\theta U(\theta) = \nabla_\theta \log f(\theta). \tag{6}$$

These equations show explicitly how the *gradient of the log posterior* is exploited. The dynamics push the parameter vector θ toward regions of high posterior density while preserving the total energy H.

Two fundamental properties of Hamiltonian dynamics are essential for MCMC:

- they are *volume preserving*, meaning that the transformation has unit Jacobian;
- they are *time reversible*.

These properties ensure that the resulting proposal can be embedded within a Metropolis–Hastings accept–reject step.

B.5 The leapfrog integrator

In practice, the differential equations above cannot be solved exactly. HMC therefore relies on a numerical scheme known as the *leapfrog integrator*, which approximates the dynamics using a step size ε.

Given a current state $(\theta, \mathbf{p})$, one leapfrog step consists of:

$$\begin{aligned}
\mathbf{p}\left(t + \tfrac{\varepsilon}{2}\right) &= \mathbf{p}(t) + \tfrac{\varepsilon}{2}\,\nabla_\theta \log f(\theta(t)), \\
\theta(t + \varepsilon) &= \theta(t) + \varepsilon\,\mathbf{M}^{-1}\mathbf{p}\left(t + \tfrac{\varepsilon}{2}\right), \\
\mathbf{p}(t + \varepsilon) &= \mathbf{p}\left(t + \tfrac{\varepsilon}{2}\right) + \tfrac{\varepsilon}{2}\,\nabla_\theta \log f(\theta(t + \varepsilon)).
\end{aligned}$$

The leapfrog scheme is carefully designed to preserve reversibility and volume, and to approximately conserve the Hamiltonian. As a result, large moves in parameter space can be proposed with high acceptance probability.

B.6 Metropolis correction

After L leapfrog steps, the algorithm produces a proposal $(\boldsymbol{\theta}', \mathbf{p}')$. Because numerical integration introduces small errors, the Hamiltonian is not exactly preserved. The proposal is therefore accepted with probability

$$\alpha = \min\bigl(1, \exp\bigl[-\mathsf{H}(\boldsymbol{\theta}', \mathbf{p}') + \mathsf{H}(\boldsymbol{\theta}, \mathbf{p})\bigr]\bigr)\,. \tag{7}$$

If the proposal is accepted, the new state of the chain is $\boldsymbol{\theta}'$; otherwise, the chain remains at $\boldsymbol{\theta}$. This correction step guarantees that the Markov chain has the desired stationary distribution.

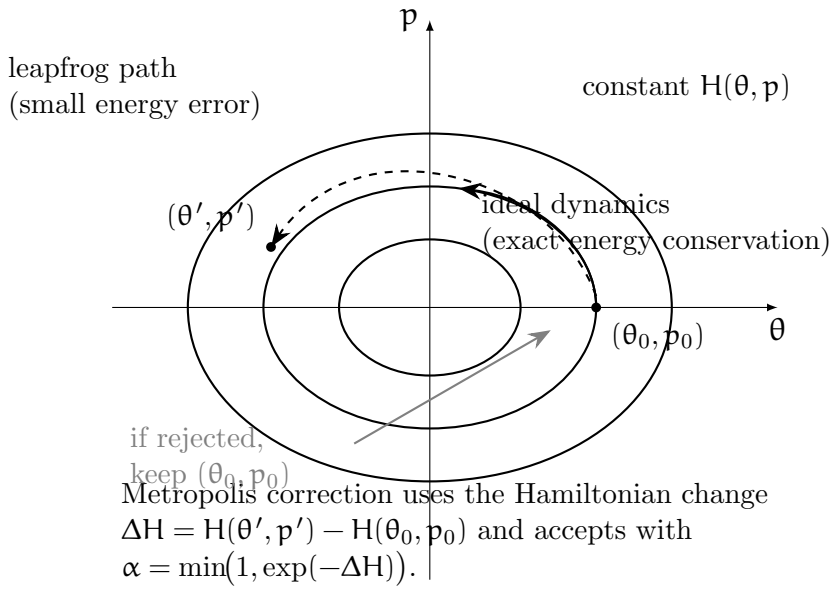

Figure 5: Phase-space view of HMC in one dimension. By introducing a momentum variable p, the sampler simulates (approximate) Hamiltonian dynamics in the augmented space (θ, p). Exact dynamics would conserve the Hamiltonian $\mathsf{H}(\theta, \mathsf{p}) = \mathsf{U}(\theta) + \mathsf{K}(\mathsf{p})$ and move along a constant-energy contour. The leapfrog integrator is reversible and volume preserving but introduces a small numerical energy error $\Delta\mathsf{H}$. The Metropolis accept/reject step corrects for this error, accepting the proposed state with probability $\alpha = \min(1, e^{-\Delta\mathsf{H}})$. As the step size decreases, $\Delta\mathsf{H}$ tends to be smaller and acceptance tends to increase.

B.7 Automatic tuning with NUTS

Choosing the number of leapfrog steps L is difficult in practice. PyMC therefore uses the *No-U-Turn Sampler* (NUTS), which dynamically extends the Hamiltonian trajectory forward and backward in time until it detects that the path is starting to reverse direction.

NUTS eliminates the need for manual tuning of L while preserving the same theoretical guarantees as HMC. From the user's perspective, only a target acceptance rate must be specified, and the sampler adapts internally to the geometry of the posterior distribution.

B.8 Implications for Bayesian estimation

Compared to standard Metropolis–Hastings algorithms, HMC and NUTS:

- exploit gradients of the log posterior to guide proposals,
- generate long-distance moves with high acceptance rates,
- handle strong posterior correlations efficiently,

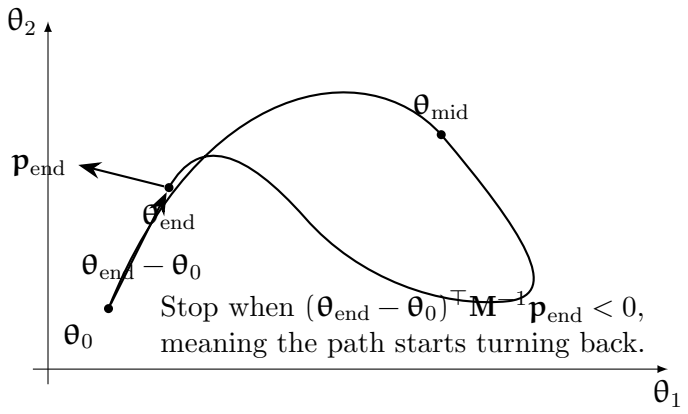

Figure 6: Geometric intuition for the NUTS stopping rule. NUTS grows a Hamiltonian trajectory and stops when the direction of travel (represented by the momentum) starts pointing back toward earlier states, indicating a "U-turn." This avoids wasting computation on trajectories that retrace their steps, and removes the need to manually choose the number of leapfrog steps.

- scale well to high-dimensional parameter spaces.

These properties explain why PyMC can reliably estimate complex Bayesian choice models, latent-variable models, and mixture models that would be computationally prohibitive with simpler MCMC methods.

Bibliographic pointers. The Metropolis–Hastings accept–reject framework dates to Metropolis et al. (1953) and Hastings (1970). The original "hybrid/Hamiltonian" Monte Carlo construction in statistics and lattice field theory is due to Duane et al. (1987). A widely cited modern reference that derives HMC from Hamiltonian mechanics, explains the role of momentum variables and the mass matrix, and connects energy conservation to high acceptance rates is Neal's handbook chapter (Neal, 2011). Practical and geometric intuition about HMC diagnostics (energy error, divergences, tuning, mass-matrix adaptation) is emphasized by Betancourt (2017). The No-U-Turn Sampler (NUTS), which removes the need to hand-tune the path length, is introduced by Hoffman and Gelman (2014). The leapfrog (Störmer–Verlet) integrator used in HMC is a standard symplectic, reversible, volume-preserving method; a canonical reference on symplectic integrators and geometric numerical integration is Hairer et al. (2006). For extensions that exploit local curvature via Riemannian geometry (conceptually related to mass-matrix design), see Girolami and Calderhead (2011).

C How PyMC implements implicit integration over random parameters.

The key reason why Bayesian estimation in Biogeme, via PyMC, does not require explicit Monte–Carlo integration lies in the way the probabilistic model is constructed and sampled. In PyMC, all unknown quantities are represented as random variables within a single joint probabilistic model. This includes both the structural parameters (such as mean taste coefficients) and the random parameters associated with individual heterogeneity. In other words, the random coefficients that would traditionally be integrated out in a mixture of logit likelihood are explicitly introduced as latent variables in the model.

Formally, let $\boldsymbol{\beta}$ denote the population-level parameters and let $\boldsymbol{\eta}_n$ denote the individual-specific random coefficients for individual n. Instead of defining a marginal likelihood of the form

$$p(y_n \mid \boldsymbol{\beta}) = \int p(y_n \mid \boldsymbol{\eta}_n)\, p(\boldsymbol{\eta}_n \mid \boldsymbol{\beta})\, d\boldsymbol{\eta}_n,$$

PyMC defines the joint posterior

$$p(\boldsymbol{\beta}, \boldsymbol{\eta}_{1:N} \mid \mathcal{D}) \propto \left[\prod_{n=1}^{N} p(y_n \mid \boldsymbol{\eta}_n)\, p(\boldsymbol{\eta}_n \mid \boldsymbol{\beta}) \right] p(\boldsymbol{\beta}),$$

where $\mathcal{D}$ denotes the observed choices. The individual-level random coefficients $\boldsymbol{\eta}_n$ are treated exactly like any other unknown parameter and are sampled jointly with $\boldsymbol{\beta}$ by the MCMC algorithm.

During sampling, PyMC repeatedly generates draws

$$\left(\boldsymbol{\beta}^{(s)}, \boldsymbol{\eta}_{1:N}^{(s)}\right), \qquad s = 1, \ldots, S,$$

from this joint posterior distribution. Conditional on a given draw $\boldsymbol{\eta}_n^{(s)}$, the likelihood contribution of observation n is simply the kernel of the elementary discrete choice model, typically a logit probability. The role that numerical integration plays in classical maximum likelihood estimation is thus replaced by averaging over posterior draws:

$$p(y_n \mid \boldsymbol{\beta}) \approx \frac{1}{S} \sum_{s=1}^{S} p\left(y_n \mid \boldsymbol{\eta}_n^{(s)}\right),$$

where the averaging is performed implicitly by the MCMC procedure itself.

From the user's perspective, this has an important practical implication. When specifying a mixture model in Biogeme, one does not write the integrated (mixed) likelihood. Instead, one provides only the conditional log-likelihood of the elementary model, given a realization of the random coefficients. PyMC handles the sampling of the latent random parameters and, through Monte–Carlo averaging over the posterior draws, automatically approximates the required integrals. This approach is both theoretically exact in the limit of an infinite number of draws and numerically stable, as it avoids introducing simulation noise directly into the likelihood evaluation.

References

Betancourt, M. (2017). A conceptual introduction to hamiltonian monte carlo. arXiv:1701.02434.

Duane, S., Kennedy, A. D., Pendleton, B. J. and Roweth, D. (1987). Hybrid monte carlo, *Physics Letters B* **195**(2): 216–222. DOI: 10.1016/0370-2693(87)91197-X.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A. and Rubin, D. B. (2014). *Bayesian Data Analysis*, Texts in Statistical Science Series, third edition edn, Chapman & Hall/CRC.

Girolami, M. and Calderhead, B. (2011). Riemann manifold langevin and hamiltonian monte carlo methods, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73**(2): 123–214. DOI: 10.1111/j.1467-9868.2010.00765.x.

Hairer, E., Lubich, C. and Wanner, G. (2006). *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, 2 edn, Springer. DOI: 10.1007/3-540-30666-8.
URL: *https://doi.org/10.1007/3-540-30666-8*

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications, *Biometrika* **57**(1): 97–109. DOI: 10.1093/biomet/57.1.97.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo, *Journal of Machine Learning Research* **15**(47): 1593–1623. Preprint: arXiv:1111.4246.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of state calculations by fast computing machines, *The Journal of Chemical Physics* **21**(6): 1087–1092. DOI: 10.1063/1.1699114.

Neal, R. M. (2011). Mcmc using hamiltonian dynamics, *in* S. Brooks, A. Gelman, G. L. Jones and X.-L. Meng (eds), *Handbook of Markov Chain Monte Carlo*, Chapman and Hall/CRC, pp. 113–162. Also available as arXiv:1206.1901.
URL: *https://arxiv.org/abs/1206.1901*

Ross, S. (2012). *Simulation*, fifth edition edn, Academic Press.

Vehtari, A., Gelman, A. and Gabry, J. (2016). Practical bayesian model evaluation using leave-one-out cross-validation and waic, *Statistics and Computing* **27**(5): 1413–1432. DOI: 10.1007/s11222-016-9696-4.

Wang, W. (2022). An introduction to the markov chain Monte Carlo method, *American Journal of Physics* **90**(12): 921–934. DOI: 10.1119/5.0122488.

Watanabe, S. (2010). Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. DOI: https://doi.org/10.48550/arXiv.1004.2316.