Journal of Statistical Software
January 2022, Volume 101, Issue 11. doi: 10.18637/jss.v101.i11

BSL: An R Package for Efficient Parameter Estimation for Simulation-Based Models via Bayesian Synthetic Likelihood

Ziwen An
Queensland University of Technology

Leah F. South
Lancaster University
Queensland University of Technology

Christopher Drovandi
Queensland University of Technology

Abstract

Bayesian synthetic likelihood (BSL; Price, Drovandi, Lee, and Nott 2018) is a popular method for estimating the parameter posterior distribution for complex statistical models and stochastic processes that possess a computationally intractable likelihood function. Instead of evaluating the likelihood, BSL approximates the likelihood of a judiciously chosen summary statistic of the data via model simulation and density estimation. Compared to alternative methods such as approximate Bayesian computation (ABC), BSL requires little tuning and requires less model simulations than ABC when the chosen summary statistic is high-dimensional. The original synthetic likelihood relies on a multivariate normal approximation of the intractable likelihood, where the mean and covariance are estimated by simulation. An extension of BSL considers replacing the sample covariance with a penalized covariance estimator to reduce the number of required model simulations. Further, a semi-parametric approach has been developed to relax the normality assumption. Finally, another extension of BSL aims to develop a more robust synthetic likelihood estimator while acknowledging there might be model misspecification. In this paper, we present the R package **BSL** that amalgamates the aforementioned methods and more into a single, easy-to-use and coherent piece of software. The package also includes several examples to illustrate use of the package and the utility of the methods.

Keywords: approximate Bayesian computation, covariance matrix estimation, Markov chain Monte Carlo, likelihood-free methods, pseudo-marginal MCMC, model misspecification, whitening transformation.

1. Introduction

In the Bayesian framework, inference on the parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^p$ of a statistical model is carried out using the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{y})$, where $\boldsymbol{y}$ is the observed data. Bayes' theorem shows that by incorporating information about $\boldsymbol{\theta}$ obtained from observed data $\boldsymbol{y}$ via the likelihood function $p(\boldsymbol{y}|\boldsymbol{\theta})$, the prior knowledge $p(\boldsymbol{\theta})$ can be updated to provide the posterior distribution,

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\Theta}} p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}} \propto p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

In most applications, the evidence $\int_{\boldsymbol{\Theta}} p(\boldsymbol{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}$ involves high dimensional integration and is intractable. Recovery of the posterior distribution often relies on sampling methods, such as Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC; Del Moral, Doucet, and Jasra 2006).

However, in complex models, the likelihood function can be intractable or very expensive to evaluate. The terminology "likelihood-free inference" typically refers to inference techniques that do not require direct evaluation of the likelihood function, but rely on model simulations to approximate the likelihood in some way. One successful method is approximate Bayesian computation (ABC; Sisson, Fan, and Beaumont 2018). ABC essentially estimates the intractable likelihood non-parametrically at $\boldsymbol{\theta}$ with a simulation $\boldsymbol{x} \sim p(\cdot|\boldsymbol{\theta})$. The raw dataset is usually reduced down to summaries with a carefully chosen summary statistic function $\boldsymbol{S}(\cdot) : \mathbb{R}^\delta \mapsto \mathbb{R}^d$, where δ and d are the dimension of the raw data and summary statistics, respectively. Denote the observed and simulated summary statistics as $\boldsymbol{s_y} = \boldsymbol{S}(\boldsymbol{y})$ and $\boldsymbol{s_x} = \boldsymbol{S}(\boldsymbol{x})$, respectively. The ABC likelihood function is given by

$$p_\epsilon(\boldsymbol{s_y}|\boldsymbol{\theta}) = \int_{\mathcal{Y}} K_\epsilon(\rho(\boldsymbol{s_y}, \boldsymbol{s_x}))p(\boldsymbol{x}|\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{x},$$

where $\rho(\cdot)$ is a discrepancy function, which measures the distance between the observed and simulated summary statistics under a certain metric, e.g., the Euclidean distance. Also, $K_\epsilon(\cdot)$ is a kernel weighting function, usually an indicator function $I(\cdot < \epsilon)$ for convenience, which links distance and tolerance ϵ. Here ϵ is used to trade-off between the bias and variance of the likelihood estimator, i.e., as ϵ approaches zero the bias reduces but the variance increases. In sampling methods like MCMC, a likelihood estimator with large variance can reduce efficiency by causing the Markov chain to become stuck (Doucet, Pitt, Deligiannidis, and Kohn 2015).

Due to the non-parametric nature of the ABC likelihood estimate, ABC can be very inefficient when the summary statistic is high dimensional (Blum 2010), which is often referred to as the curse of dimensionality. Furthermore, ABC requires the user to select ρ, K_ϵ and ϵ, and the results can be sensitive to these choices.

Wood (2010) proposes to use a multivariate normal distribution to approximate the likelihood function. Such an approximation is called the synthetic likelihood (SL). We later extend the term to not only the multivariate normal distribution but also other reasonable parametric approximations of the likelihood (Drovandi, Pettitt, and Lee 2015, provide a general framework for such methods). Price *et al.* (2018) analyze SL in the Bayesian framework and name it Bayesian synthetic likelihood (BSL). MCMC is used to explore the parameter space. The paper uses extensive empirical results to show that BSL can outperform ABC even when the model summary statistics show small departures from normality. BSL not only scales better with the summary statistic dimension, but also requires less tuning and can be accelerated with parallel computing. Recently, asymptotic properties of BSL have been derived under various assumptions. Frazier and Drovandi (2021) develop a result for posterior concentration and Frazier, Nott, Drovandi, and Kohn (2021) show that the BSL posterior mean is consistent and asymptotically normally distributed.

In the standard BSL approach, the most obvious computational drawback is the need to generate a large number of simulations n for estimating a high-dimensional covariance matrix in the SL. Several strategies have been devised to reduce the number of simulations n for estimating the synthetic likelihood. An, South, Nott, and Drovandi (2019) propose to use the graphical lasso to estimate a penalized covariance and provide an algorithm for selecting the penalty to ensure reasonable mixing when placed inside an MCMC algorithm. Ong, Nott, Tran, Sisson, and Drovandi (2018a) consider the shrinkage estimator of Warton (2008) in the context of variational Bayesian synthetic likelihood to reduce the number of simulations. To improve the efficacy of Warton's shrinkage estimator within BSL, Priddle, Sisson, and Drovandi (in press) employ a whitening transformation of the summary statistic. The whitening transformation can significantly de-correlate the summary statistic so that more shrinkage can be applied without losing much accuracy. Everitt (2017) considers a bootstrap approximation to lower the variance of the SL estimator. Furthermore, the normality assumption of SL has also been put under inspection by An, Nott, and Drovandi (2020), who consider a more flexible approximation of the synthetic likelihood using a semi-parametric estimator with a Gaussian copula. Frazier and Drovandi (2021) develop a robust BSL method by introducing a latent parameter to account for the proposed model to be potentially unable to re-produce all of the observed summary statistics.

There are existing packages in R (R Core Team 2021) for likelihood-free inference. For example, the R packages **abc** (Csillery, Francois, and Blum 2012) and **EasyABC** (Jabot, Faure, and Dumoulin 2013) exist for various ABC methods. The R package **synlik** (Fasiolo and Wood 2021) implements the classic SL method of Wood (2010). The package provides diagnostic tools for the normal synthetic likelihood and incorporates MCMC to find the approximate posterior distribution. However, **synlik** is limited to only the standard SL approach. Outside of R, the **ABCpy** (Dutta *et al.* 2021) package in Python (Van Rossum *et al.* 2011) includes most of the popular ABC algorithms and the standard SL approach. Further, the **ELFI** (engine for likelihood-free inference) package (Lintusaari *et al.* 2018) in Python currently has support for various ABC algorithms and also BOLFI (Bayesian optimization for likelihood-free inference; Gutmann and Corander 2016).

Given the wide applicability of BSL (e.g., Karabatsos 2018; Barbu, Sethuraman, Billig, and Levy 2018), it is important that BSL methods are directly accessible to practitioners. BSL has been used for statistical inference in forest models (Hartig, Dislich, Wiegand, and Huth 2014), cell biology (Drovandi, Grazian, Mengersen, and Robert 2018), cosmology (Leclercq 2018), stochastic differential equation mixed effects models (Picchini and Forman 2019) and time-censored aggregate data (Chen, Ye, and Zhai 2020). In this paper we introduce our **BSL** R package (An, South, and Drovandi 2022), which implements the Bayesian version of synthetic likelihood and many of the extensions listed earlier together with additional functionality detailed later in the paper and which is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=BSL`. The package is flexible in terms of the prior specification, the implementation of the model simulation function and the choice of the summary statistic function. Further, it includes several built-in examples to help the user learn how to use the package.

The rest of this paper is structured as follows. Section 2 provides a brief description of the statistical background for SL and several extensions based on unbiased estimators, semi-parametric approaches, robustness to misspecification and shrinkage approaches. In Sections 3 and 4, we introduce the main functionalities of **BSL** with an illustrative toy example and a real example, respectively. Section 5 concludes the paper with a summary and further discussion.

2. Bayesian synthetic likelihood

Following the notation in Section 1, we focus on reviewing three SL estimators to $p(\boldsymbol{s_y}|\boldsymbol{\theta})$ and two shrinkage covariance estimation methods in Section 2.1 to Section 2.5. We briefly introduce other implementation details in Section 2.6.

The SL estimator can be viewed as an auxiliary likelihood function $p_A(\boldsymbol{s_y}|\boldsymbol{\phi})$ where the subscript "A" denotes that we are using a parametric density approximation and $\boldsymbol{\phi}$ is the parameter of this parametric family of densities. To link this auxiliary likelihood to the actual likelihood, there is a functional relationship between $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, which may be denoted as $\boldsymbol{\phi}(\boldsymbol{\theta})$. However, we drop the dependence on $\boldsymbol{\theta}$ for notational convenience. The SL posterior is given by

$$p_A(\boldsymbol{\theta}|\boldsymbol{s_y}) \propto p_A(\boldsymbol{s_y}|\boldsymbol{\phi})p(\boldsymbol{\theta}).$$

Unfortunately, the mapping from $\boldsymbol{\theta}$ to $\boldsymbol{\phi}$ is typically unknown. However $\boldsymbol{\phi}$ can be estimated with simulations. The estimated synthetic likelihood is placed within an MCMC algorithm to sample from the corresponding approximate posterior of $\boldsymbol{\theta}$. Below we describe the synthetic likelihood estimators supported by our package, which amount to choosing the form of p_A and the type of estimator of $\boldsymbol{\phi}$.

2.1. The standard BSL likelihood estimator

Assume we have obtained a collection of n simulations from the model at a proposed parameter value of $\boldsymbol{\theta}$, i.e., $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \overset{\text{iid}}{\sim} p(\cdot|\boldsymbol{\theta})$. The corresponding summary statistics are denoted as $\boldsymbol{s}_i = \boldsymbol{S}(\boldsymbol{x}_i) \in \mathcal{S} \subseteq \mathbb{R}^d$ for $i = 1, \ldots, n$. Following Wood (2010), the SL $p(\boldsymbol{s_y}|\boldsymbol{\theta})$ is assumed to be roughly multivariate normal. The classic SL estimator can be written as

$$p_{sl}(\boldsymbol{s_y}|\boldsymbol{\phi}_{sl}) = \mathcal{N}(\boldsymbol{s_y}|\boldsymbol{\phi}_{sl}), \tag{1}$$

where $\boldsymbol{\phi}_{sl} = (\boldsymbol{\mu}(\boldsymbol{\theta}), \boldsymbol{\Sigma}(\boldsymbol{\theta}))$ is estimated with sample mean and covariance $\hat{\boldsymbol{\phi}}_{sl} = (\boldsymbol{\mu}_n(\boldsymbol{\theta}), \boldsymbol{\Sigma}_n(\boldsymbol{\theta}))$

$$\begin{aligned}\boldsymbol{\mu}_n(\boldsymbol{\theta}) &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{s}_i \\ \boldsymbol{\Sigma}_n(\boldsymbol{\theta}) &= \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{s}_i - \boldsymbol{\mu}_n(\boldsymbol{\theta}))(\boldsymbol{s}_i - \boldsymbol{\mu}_n(\boldsymbol{\theta}))^\top.\end{aligned} \tag{2}$$

Although $\hat{\boldsymbol{\phi}}_{sl}$ is an unbiased estimator of $\boldsymbol{\phi}_{sl}$, $\mathcal{N}(\boldsymbol{s_y}|\hat{\boldsymbol{\phi}}_{sl})$ is not an unbiased estimator of $\mathcal{N}(\boldsymbol{s_y}|\boldsymbol{\phi}_{sl})$. However, empirical results demonstrate that the approximate posterior exhibits very weak dependence on n (Price *et al.* 2018). Nevertheless, if n is prohibitively small, then there can be significant Monte Carlo error in the MCMC approximation of the BSL target.

2.2. An unbiased BSL likelihood estimator

Ghurye and Olkin (1969) provide an unbiased estimator for the multivariate normal density based on independent simulations from it. Price *et al.* (2018) show the viability of this estimator by using it in place of the standard SL estimator within an MCMC algorithm. The estimator, denoted as uBSL, requires $n > d + 3$ for unbiasedness to hold. The estimated auxiliary parameter $\hat{\boldsymbol{\phi}}_{go}$ can be written as $(\boldsymbol{\mu}_n(\boldsymbol{\theta}), \boldsymbol{M}_n(\boldsymbol{\theta}))$, where $\boldsymbol{M}_n(\boldsymbol{\theta}) = (n-1)\boldsymbol{\Sigma}_n(\boldsymbol{\theta})$. Definitions of $\boldsymbol{\mu}_n(\boldsymbol{\theta})$ and $\boldsymbol{\Sigma}_n(\boldsymbol{\theta})$ can be found in Equation 2. The unbiased likelihood estimator is given by

$$p_{go}(\boldsymbol{s_y}|\boldsymbol{\phi}_{go}) = (2\pi)^{-d/2}\frac{c(d,\, n-2)}{c(d,\, n-1)(1-1/n)^{d/2}}|\boldsymbol{M_n}(\boldsymbol{\theta})|^{-(n-d-2)/2} \Psi\Big(\boldsymbol{M_n}(\boldsymbol{\theta}) - \frac{(\boldsymbol{s_y}-\boldsymbol{\mu}_n(\boldsymbol{\theta}))(\boldsymbol{s_y}-\boldsymbol{\mu}_n(\boldsymbol{\theta}))^\top}{1-1/n}\Big)^{(n-d-3)/2}, \quad (3)$$

where

$$c(k, v) = \frac{2^{-kv/2}\pi^{-k(k-1)/4}}{\prod_{i=1}^{k}\Gamma\Big(\frac{v-i+1}{2}\Big)}$$

and

$$\Psi(\boldsymbol{A}) = \begin{cases} |\boldsymbol{A}| & \text{if } \boldsymbol{A} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

In spite of the fact that the normality assumption is rarely true in practice, the approximate posterior distributions obtained by uBSL show less dependence on n when compared to BSL (Price *et al.* 2018).

2.3. A semi-parametric BSL likelihood estimator

Although the estimators of Equations 1 and 3 are straightforward to compute, the normality assumption of the summary statistic could be restrictive in some cases. It is thus of interest to consider a more robust likelihood estimator that can handle non-normal summary statistics whilst not sacrificing much on computational efficiency.

Here we briefly describe the semi-parametric likelihood estimator of An *et al.* (2020), which is aliased as "semiBSL" by the authors. The paper shows examples where BSL fails to produce accurate posterior approximations whilst semiBSL can still produce reasonable posterior estimates. The semiBSL estimator harnesses kernel density estimation (KDE) to non-parametrically estimate the univariate distributions of the summary statistics and uses the Gaussian copula to model the dependence amongst summaries. This semi-parametric estimator provides additional robustness over the multivariate normal assumption and is quick to apply.

The estimator involves two aspects: modeling the marginals and modeling the dependence structure. We first describe KDE for estimating the marginals. Denoting s_i^j for $j = 1, \ldots, d$ as the j-th component of $\boldsymbol{s}_i$, if the summary statistic is continuous or approximately continuous (e.g., large counts), a KDE of the j-th marginal density is given by

$$\hat{g}_j(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - s_i^j), \quad (4)$$

where $K_h(\cdot)$ is a non-negative kernel function with bandwidth parameter h. See Izenman (1991) for a review and discussion of KDE. Similarly, the corresponding estimated cumulative distribution function $\hat{G}_j(\cdot)$ can be obtained. The choice of the kernel function could be various. An *et al.* (2020) select the Gaussian kernel for simplicity and for its unbounded support.

By using KDE, the semiBSL estimator can accommodate non-normal marginal summary statistics. For the second aspect, semiBSL uses a Gaussian copula to model dependence between summaries. The density function for the Gaussian copula is given by

$$c_{\boldsymbol{R}}(\boldsymbol{u}) = \frac{1}{|\boldsymbol{R}|} \exp\Big\{ -\frac{1}{2}\boldsymbol{\eta}^\top(\boldsymbol{R}^{-1} - \boldsymbol{I}_d)\boldsymbol{\eta}\Big\},$$

where $\boldsymbol{R}$ is the correlation matrix, $\boldsymbol{\eta} = (\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_d))^\top$, $\Phi^{-1}(\cdot)$ is the inverse CDF of a standard normal distribution, and $\boldsymbol{I}_d$ is a d-dimensional identity matrix. The main appeal of the Gaussian copula here is that it is fast to estimate the correlation matrix (see details later). Combining the advantages from both aspects, we are then able to have a tractable likelihood estimator written as $p_{ssl}(\boldsymbol{s_y}|\boldsymbol{\phi}_{ssl})$, where $\boldsymbol{\phi}_{ssl} = (\boldsymbol{R}, u_1, \dots, u_d, g_1(s_y^1), \dots, g_d(s_y^d))$ is estimated by $\hat{\boldsymbol{\phi}}_{ssl} = (\hat{\boldsymbol{R}}, \hat{u}_1, \dots, \hat{u}_d, \hat{g}_1(s_y^1), \dots, \hat{g}_d(s_y^d))$, $\hat{u}_j = \hat{G}_j(s_y^j)$ and $\hat{g}_j(s_y^j)$ is given by Equation 4. The full semiBSL density estimator is given by

$$p_{ssl}(\boldsymbol{s_y}|\hat{\boldsymbol{\phi}}_{ssl}) = \frac{1}{\sqrt{\hat{\boldsymbol{R}}}} \exp\Big\{ -\frac{1}{2}\hat{\boldsymbol{\eta}}_{\boldsymbol{s_y}}^\top(\hat{\boldsymbol{R}}^{-1} - \boldsymbol{I}_d)\hat{\boldsymbol{\eta}}_{\boldsymbol{s_y}}\Big\} \prod_{j=1}^{d} \hat{g}_j(s_y^j). \tag{5}$$

Finally we provide details on obtaining $\hat{\boldsymbol{R}}$. With a collection of simulations $\{\boldsymbol{\eta}_{\boldsymbol{s}_i}\}_{i=1}^n$, it can either be estimated with maximum likelihood estimation, or as An *et al.* (2020) advocate, one could use the Gaussian rank correlation (GRC; Boudt, Cornelissen, and Croux 2012). The GRC estimator is known to be more robust. The (i,j)-th entry of the GRC matrix is given by

$$\hat{\rho}_{i,j}^{\text{grc}} = \frac{\sum_{k=1}^n \Phi^{-1}\Big(\frac{r(s_k^i)}{n+1}\Big)\Phi^{-1}\Big(\frac{r(s_k^j)}{n+1}\Big)}{\sum_{k=1}^n \Phi^{-1}\Big(\frac{k}{n+1}\Big)^2},$$

where $r(\cdot) : \mathbb{R} \to \mathcal{A}$, and $\mathcal{A} \equiv \{1, \dots, n\}$, is the rank function. Note that here the GRC is used to estimated the correlation matrix in semiBSL. However, our package also permits the use of the GRC in standard BSL to provide more robustness. In doing so, a transformation to the covariance matrix is needed afterwards.

We point out that when the data generation process is complicated, most of the computational cost will be spent on model simulations. Thus the computational overhead introduced by the semi-parametric likelihood estimator may be negligible relative to the standard SL estimator.

2.4. A robust BSL likelihood estimator for model misspecification

Model misspecification could be particularly harmful to likelihood-free simulation-based approaches as the data generating process (DGP) is often complicated. Frazier, Robert, and Rousseau (2020) demonstrate how the pseudo-true parameter value is not recovered asymptotically by ABC algorithms under model misspecification. Further, Frazier and Drovandi (2021) demonstrate that BSL can produce unreliable inferences when the model is not compatible with the observed summary statistic (compatibility is described in more detail in

Definition 1 below). Let $P_{\boldsymbol{\theta}}$ and $\Pi_{\boldsymbol{\theta}}$ be the probability measure for the likelihood and prior beliefs for some $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^p$, respectively. Let $b_0 := \text{plim}\boldsymbol{S}(\boldsymbol{y})$ and $b(\boldsymbol{\theta}) := \text{plim}\boldsymbol{S}(\boldsymbol{x})$ be the probability limit for the observed and simulated summary statistics, respectively.

Definition 1. The model $P_{\boldsymbol{\theta}} \times \Pi_{\boldsymbol{\theta}}$ and summary statistic map $\boldsymbol{S}(\cdot)$ are compatible if

$$\inf_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \|b(\boldsymbol{\theta}) - b_0\| = 0.$$

The above definition of compatibility essentially means that asymptotically the observed summary statistic can be replicated with the assumed DGP and some parameter value with prior support. With the notion of compatibility defined, Frazier and Drovandi (2021) proposed two different approaches, mean adjustment and variance inflation, that help address incompatibility between model and summary statistics. We refer to these methods in this paper as robust BSL (rBSL). Both approaches define an augmented parameter $\boldsymbol{\zeta} = (\boldsymbol{\theta}^\top, \boldsymbol{\gamma}^\top)^\top \in \boldsymbol{\Theta} \times \boldsymbol{\mathcal{G}} \subseteq \mathbb{R}^{p+d}$, where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_d)^\top$ is the additional free parameter to account for mean adjustment or variance inflation depending on the approach. The prior distribution of $\boldsymbol{\zeta}$ can be written as $p(\boldsymbol{\theta}) \times p(\boldsymbol{\gamma})$, where $p(\boldsymbol{\gamma})$ is the prior distribution of $\boldsymbol{\gamma}$. We use the prior choice of Frazier and Drovandi (2021) in the **BSL** package, but refer the readers to the original paper for details.

Frazier and Drovandi (2021) demonstrate that when Definition 1 holds, the posterior for $\boldsymbol{\gamma}$ converges to its prior, and the inferences on $\boldsymbol{\theta}$ are similar to when standard BSL is used. When compatibility is not satisfied, the posterior for $\boldsymbol{\gamma}$ will concentrate away from zero and will be different from the prior for the corresponding statistics that the model cannot match. This allows the inferences on $\boldsymbol{\theta}$ to be less impacted by statistics that the model is not compatible with. A useful by-product of rBSL, in the presence of incompatibility, is that the variance of the robust synthetic likelihood estimator is much smaller than that of the standard synthetic likelihood estimator, since the mean adjustment and variance inflation help ensure that the observed statistic does not sit as far in the tails of the synthetic likelihood. Frazier and Drovandi (2021) demonstrate that the efficiency of MCMC can be greatly improved with rBSL when compatibility is not met.

The rBSL likelihood is still approximated by a Gaussian distribution,

$$p_{rsl}(\boldsymbol{s_y}|\boldsymbol{\phi}_{rsl}) = \mathcal{N}(\boldsymbol{s_y}|\boldsymbol{\phi}_{rsl}), \tag{6}$$

where $\boldsymbol{\phi}_{rsl}$ represents the auxiliary parameter of choice, i.e., $\boldsymbol{\phi}_{rsl\text{-}m}$ and $\boldsymbol{\phi}_{rsl\text{-}v}$ for mean adjustment and variance inflation approaches, respectively. For notational convenience, in Section 2.4, let $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ be the sample mean $\boldsymbol{\mu}_n(\boldsymbol{\theta})$ and sample covariance $\boldsymbol{\Sigma}_n(\boldsymbol{\theta})$ in Equation 2, respectively.

Mean adjustment

The mean adjustment approach (rBSL-M) aims to handle the incompatibility by adjusting the simulated means. Following the notation of sample mean and covariance in Equation 2, the adjusted mean parameter is defined as

$$\tilde{\boldsymbol{\mu}}_n(\boldsymbol{\zeta}) = \boldsymbol{\mu} + \text{diag}(\boldsymbol{\Sigma}^{1/2})\gamma,$$

where $\text{diag}(\boldsymbol{A})$ represents the diagonal matrix with diagonal elements the same as the diagonal elements of $\boldsymbol{A}$. This ensures that the components of $\boldsymbol{\gamma}$ have the same scale. The auxiliary parameter for rBSL-M can be written as $\boldsymbol{\phi}_{rsl\text{-}m} = (\tilde{\boldsymbol{\mu}}_n(\boldsymbol{\zeta}), \boldsymbol{\Sigma})$.

Variance inflation

Similarly, the variance inflation approach (rBSL-V) allows the variance of the synthetic likelihood to be inflated. The variance parameter for rBSL-V is given by

$$\tilde{\boldsymbol{\Sigma}}_n(\boldsymbol{\zeta}) = \boldsymbol{\Sigma} + \begin{pmatrix} \sigma_{11}\gamma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_{22}\gamma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{pp}\gamma_p^2 \end{pmatrix},$$

where σ_{ij} is the (i,j)-th element of $\boldsymbol{\Sigma}$. The auxiliary parameter for rBSL-V can be written as $\boldsymbol{\phi}_{rsl\text{-}v} = (\boldsymbol{\mu}, \tilde{\boldsymbol{\Sigma}}_n(\boldsymbol{\zeta}))$.

2.5. An accelerated likelihood estimator with shrinkage estimation

We can see from Equations 1 and 5 that both estimators involve estimation of a covariance or correlation matrix. The number of free parameters in the covariance matrix grows quadratically with the number of summary statistics. Thus, for a high-dimensional summary statistic, a large number n of simulations may be required to estimate the likelihood with reasonable precision.

Here we introduce two shrinkage estimators of the covariance matrix that have shown to be useful in empirical examinations (Ong *et al.* 2018a; An *et al.* 2019, 2020). The shrinkage estimators can be applied to BSL and semiBSL. Note that we skip uBSL because the unbiasedness property will be violated with shrinkage estimation. The selection of the penalty parameter will be discussed later in Section 2.6. For clarification, we change the notations in Section 2.5 temporarily and restore them back after. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be samples from a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The dimension of each sample is d. $\boldsymbol{S}$ is the sample covariance and $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix.

Graphical lasso

The graphical lasso (Friedman, Hastie, and Tibshirani 2008) aims to find a sparse precision matrix by maximizing the following l_1 regularized log-likelihood

$$\log p(x) = c + \log|\boldsymbol{\Theta}| - tr(\boldsymbol{\Theta}\boldsymbol{S}) - \lambda\|\boldsymbol{\Theta}\|_1,$$

where c is a constant, $\|\cdot\|_1$ is the l_1 norm, and λ is the penalty parameter. The problem is solved with convex optimization. The penalty parameter λ controls the sparsity in $\boldsymbol{\Theta}$, where large λ leads to high sparsity and vice versa. We refer the readers to Friedman *et al.* (2008) for more details about the graphical lasso. In our package we use the graphical lasso implementation in the R package **glasso** (Friedman, Hastie, and Tibshirani 2019).

Warton's estimator

Warton (2008) introduces a straightforward shrinkage estimator that is extremely fast to compute but still performs well. Define $\hat{\boldsymbol{D}}$ as the diagonal matrix of $\boldsymbol{S}$, then the sample correlation matrix can be computed with $\hat{\boldsymbol{C}} = \hat{\boldsymbol{D}}^{-1/2}\boldsymbol{S}\hat{\boldsymbol{D}}^{-1/2}$. The Warton's shrinkage for the correlation matrix is given by

$$\hat{\boldsymbol{C}}_\lambda = \lambda\hat{\boldsymbol{C}} + (1-\lambda)\boldsymbol{I}_d,$$

where $\lambda \in [0, 1]$ is the shrinkage parameter and $\boldsymbol{I}_d$ is the identity matrix. The correlation matrix reduces to the identity matrix as λ approaches 0. With a correlation to covariance conversion, we get the Warton's covariance estimator for $\boldsymbol{\Sigma}$

$$\hat{\boldsymbol{\Sigma}}_\lambda = \hat{\boldsymbol{D}}^{1/2} \hat{\boldsymbol{C}}_\lambda \hat{\boldsymbol{D}}^{1/2}. \tag{7}$$

Warton's estimation with whitening transformations

Motivated by further reducing the number of model simulations, Priddle *et al.* (in press) propose to use a de-correlation technique based on whitening transformations to encourage greater shrinkage of the covariance matrix. With the summary statistics significantly de-correlated, Priddle *et al.* (in press) propose to use Warton's estimator (Equation 7) with a significant amount of shrinkage in order to reduce the number of model simulations required. In some numerical examples, Priddle *et al.* (in press) find that combining the whitening transformations together with Warton's shrinkage estimator permits the use of a diagonal covariance matrix in the synthetic likelihood without much loss of accuracy compared to standard BSL. The whitening transformation (Kessy, Lewin, and Strimmer 2018) is a linear transformation that aims to de-correlate the random variables (or summary statistics in the context of BSL) such that the whitened covariance matrix could be close to an identity matrix.

Let $\boldsymbol{W}$ be a $d \times d$ matrix, which satisfies $\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{\Theta}$. Let $\boldsymbol{s}$ be a d-dimensional random variable with covariance matrix $\boldsymbol{\Sigma}$. A whitening transformation on $\boldsymbol{s}$ using $\boldsymbol{W}$ is $\tilde{\boldsymbol{s}} = \boldsymbol{W}\boldsymbol{s}$. In the BSL context, $\boldsymbol{s}$ represents the summary statistic. The covariance matrix of the whitened summary statistic is given by

$$\mathrm{Var}(\tilde{\boldsymbol{s}}) = \mathrm{Var}(\boldsymbol{W}\boldsymbol{s}) = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^\top = \boldsymbol{I}_d,$$

where $\boldsymbol{W}$ is the whitening matrix. Ideally, upon the above transformation, the covariance matrix of the synthetic likelihood would be close to an identity matrix. However, the restriction of $\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{\Theta}$ still does not identify a unique matrix $\boldsymbol{W}$. Kessy *et al.* (2018) outline five natural choices of the whitening matrix. Priddle *et al.* (in press) demonstrate that the principle component analysis (PCA) whitening method produces the most accurate posterior result when applied in a BSL algorithm. Thus, we take PCA as the default method to estimate the whitening matrix. The implementation of whitening transformations in **BSL** is through the R package **whitening** (Strimmer, Jendoubi, Kessy, and Lewin 2019).

Using eigendecomposition, the covariance matrix is decomposed as $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where $\boldsymbol{U}$ is the matrix of eigenvectors and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. The PCA whitening matrix is given by

$$\boldsymbol{W}_{\mathrm{PCA}} = \boldsymbol{\Lambda}^{-1/2} \boldsymbol{U}^\top.$$

The whitening matrix to use is determined by a relatively large number of offline simulations at a point estimate of the parameter, and is subsequently fixed during BSL. If the point estimate has reasonable posterior support, then the whitening transformation can be effective in de-correlating the statistic at model parameter values with non-negligible posterior support (Priddle *et al.* in press).

Algorithm 1: Procedure to select the penalty value λ for BSL or semiBSL with shrinkage estimation.

```
Input  : The BSL method to be used, either BSL or semiBSL, denoted method; the
         shrinkage estimation to be used, either graphical lasso or Warton, denoted
         shrinkage; parameter value with non-negligible posterior support $\boldsymbol{\theta}_0$; the
         number of simulations $n$; a number of potential penalty values, $\lambda_1, \ldots, \lambda_K$; the
         standard deviation of the log-likelihood estimator to aim for $\sigma$; and the
         number of repeats $M$.
Output: The selected penalty parameter $\lambda$.
for $m = 1$ to $M$ do
    Generate $n$ datasets with the simulation function $\{\boldsymbol{x}_i\}_{i=1}^n \overset{\text{iid}}{\sim} p(\cdot|\boldsymbol{\theta}_0)$.
    Compute the corresponding summary statistics $\{\boldsymbol{s}_i\}_{i=1}^n = \{\boldsymbol{S}(\boldsymbol{x}_i)\}_{i=1}^n$.
    for $k = 1$ to $K$ do
        if method is BSL then
            Compute the penalized covariance matrix $\boldsymbol{\Sigma}_{n,\lambda_k}$ using the shrinkage method
              of shrinkage with the samples $\{\boldsymbol{s}_i\}_{i=1}^n$.
            Compute the sample mean $\boldsymbol{\mu}_n(\boldsymbol{\theta})$.
            Set $\hat{\boldsymbol{\phi}}_{sl} = (\boldsymbol{\mu}_n(\boldsymbol{\theta}), \boldsymbol{\Sigma}_{n,\lambda_k})$.
            Estimate the log SL $l_{m,k} = \log p_{sl}(\boldsymbol{s_y}|\hat{\boldsymbol{\phi}}_{sl})$ with Equation 1.
        end
        if method is semiBSL then
            Compute and penalize the Gaussian rank correlation matrix using the
              shrinkage method of shrinkage with the samples $\{\boldsymbol{s}_i\}_{i=1}^n$, and save it to
              $\boldsymbol{R}_{n,\lambda_k}$.
            Compute $\hat{u}_1, \ldots, \hat{u}_d, \hat{g}_1(s_y^1), \ldots, \hat{g}_d(s_y^d)$.
            Set $\hat{\boldsymbol{\phi}}_{ssl} = (\boldsymbol{R}_{n,\lambda_k}, \hat{u}_1, \ldots, \hat{u}_d, \hat{g}_1(s_y^1), \ldots, \hat{g}_d(s_y^d))$.
            Estimate the log SL $l_{m,k} = \log p_{ssl}(\boldsymbol{s_y}|\hat{\boldsymbol{\phi}}_{ssl})$ with Equation 5.
        end
    end
end
Compute $\sigma_k$ as the standard deviation of $\{l_{m,k}\}_{m=1}^M$ for $k = 1, \ldots, K$.
Choose the $\lambda$ for which the empirical standard deviation $\sigma_k$ is closest to $\sigma$.
```

Penalty selection

An *et al.* (2019) introduce a penalty selection approach for BSLasso (BSL with graphical lasso). The general approach can also be used with Warton's shrinkage estimator. Our goal is to reduce the value of n required while still maintaining a similar level of noise in the SL approximation. Denote σ as the standard deviation of the penalized log-likelihood estimator that we are aiming for. The value of σ is typically around 1.5, as suggested below in Section 2.6. The algorithm for selecting the penalty parameter is given in Algorithm 1. Here we use λ for the penalty parameter for both the graphical lasso and Warton's estimator for notational convenience.

2.6. Other implementation details

Incorporating BSL with MCMC

Once we have selected a likelihood function estimator, we can use a Bayesian sampling algorithm to sample from the BSL approximation to the posterior $p(\boldsymbol{\theta}|\boldsymbol{s_y})$. This is currently achieved using MCMC in our package. Pseudo-code for the MCMC BSL algorithm (except rBSL) is provided in Algorithm 2. The rBSL estimator of Section 2.4 operates on the augmented parameter space $\boldsymbol{\Theta} \times \boldsymbol{\mathcal{G}}$. Frazier and Drovandi (2021) use a component-wise MCMC algorithm to update $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ alternately. The update of $\boldsymbol{\theta}$ is the same as Algorithm 2, and in each iteration $\boldsymbol{\gamma}$ is updated with a slice sampler (Neal 2003).

Price *et al.* (2018) demonstrate empirically that the BSL posterior depends weakly on the number of simulations n used to estimate the synthetic likelihood at each iteration of MCMC. Further, Frazier *et al.* (2021) show, under some conditions, that the BSL posterior can achieve correct frequentist uncertainty quantification asymptotically with n growing at any arbitrarily slow rate as the sample size increases. Therefore, as recommended by Price *et al.* (2018), we suggest to tune n based on maximizing computational efficiency in a similar way as suggested in the pseudo-marginal MCMC literature (Andrieu and Roberts 2009). In pseudo-marginal MCMC, an intractable likelihood is replaced with a non-negative unbiased estimator based on some form of Monte Carlo using N "particles"; for example, using a particle filter with N particles to obtain an unbiased estimator of the likelihood for state space models (Andrieu, Doucet, and Holenstein 2010). Increasing N reduces the variance of the likelihood estimator, which leads to a more statistically efficient MCMC algorithm. However, the cost per MCMC iteration increases. On the other hand, taking N too small leads to poor statistical efficiency of MCMC as the chain often gets stuck for long periods at a greatly overestimated estimate of the likelihood. Doucet *et al.* (2015) suggest to choose N that produces an estimated log-likelihood with a standard deviation of roughly 1 when evaluated at a parameter value with reasonable posterior support. In the context of standard BSL, Price *et al.* (2018) demonstrate empirically that aiming for a standard deviation of the synthetic log-likelihood estimator of around 1.5–2 at a representative parameter value provides the most computationally efficient results.

As shown in Price *et al.* (2018), there is another connection between BSL and pseudo-marginal MCMC, specifically related to the uBSL algorithm of Section 2.2. Unlike pseudo-marginal MCMC, the synthetic likelihood estimated from n simulations does not produce an unbiased estimate of the actual synthetic likelihood where the relationship between $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\boldsymbol{\theta}$ is known; or equivalently assuming that we could perform an infinite number of model simulations. Therefore, as already mentioned, BSL theoretically depends on n. The uBSL algorithm of Section 2.2 can be thought of as a pseudo-marginal MCMC method that targets $p(\boldsymbol{\theta}|\boldsymbol{s_y})$ when the summary statistics follow a multivariate Gaussian distribution since it uses an exactly unbiased estimator of the multivariate Gaussian density.

We recommend that the initial value of the Markov chain has non-negligible support under the BSL posterior. Our experience with MCMC BSL is that if the chain is initialized far in the tails of the BSL posterior, then MCMC BSL exhibits slow convergence. The initial value for the Markov chain may be sourced from experts, previous analyses in the literature or a short run of another likelihood-free algorithm. In low dimensions, using BSL with multiple randomized starts may be helpful in finding a region with non-negligible support.

Algorithm 2: MCMC BSL algorithm.

```
Input  : Summary statistic of the observed data $\boldsymbol{s_y}$; the prior distribution $p(\boldsymbol{\theta})$, the
         proposal distribution $q(\cdot|\cdot)$; the number of simulations used to estimate the
         synthetic likelihood $n$; the number of iterations $M$; and the initial value of the
         chain $\boldsymbol{\theta}^0$.
Output: MCMC samples $(\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^M)$ from the BSL posterior, $p_A(\boldsymbol{\theta}|\boldsymbol{s_y})$. Some
         samples can be discarded as burn-in if required.
Generate $n$ datasets with the simulation function $\{\boldsymbol{x}_i\}_{i=1}^n \overset{\text{iid}}{\sim} p(\cdot|\boldsymbol{\theta}^0)$.
Compute the corresponding summary statistics $\{\boldsymbol{s}_i\}_{i=1}^n = \{\boldsymbol{S}(\boldsymbol{x}_i)\}_{i=1}^n$.
Estimate the auxiliary parameter $\hat{\boldsymbol{\phi}}$ depending on the SL estimator.
Compute the estimated SL $\hat{p}_A^0 = \hat{p}_A(\boldsymbol{s_y}|\hat{\boldsymbol{\phi}})$ with Equation 1, 3 or 5.
for $i = 1$ to $M$ do
    Propose a parameter value with $\boldsymbol{\theta}^* \sim q(\cdot|\boldsymbol{\theta}^{i-1})$.
    Generate $n$ datasets with the simulation function $\{\boldsymbol{x}_i^*\}_{i=1}^n \overset{\text{iid}}{\sim} p(\cdot|\boldsymbol{\theta}^*)$.
    Compute the corresponding summary statistics $\{\boldsymbol{s}_i^*\}_{i=1}^n = \{\boldsymbol{S}(\boldsymbol{x}_i^*)\}_{i=1}^n$.
    Estimate the auxiliary parameter $\hat{\boldsymbol{\phi}}$ depending on the SL estimator.
    Compute the estimated SL $\hat{p}_A^* = \hat{p}_A(\boldsymbol{s_y}|\hat{\boldsymbol{\phi}})$ with Equation 1, 3 or 5.
    Compute $r = \min\left(1, \dfrac{\hat{p}_A^* \, p(\boldsymbol{\theta}^*) \, q(\boldsymbol{\theta}^{i-1}|\boldsymbol{\theta}^*)}{\hat{p}_A^{i-1} \, p(\boldsymbol{\theta}^{i-1}) \, q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{i-1})}\right)$.
    if $\mathcal{U}(0,1) < r$ then
        Set $\boldsymbol{\theta}^i = \boldsymbol{\theta}^*$ and $\hat{p}_A^i = \hat{p}_A^*$.
    else
        Set $\boldsymbol{\theta}^i = \boldsymbol{\theta}^{i-1}$ and $\hat{p}_A^i = \hat{p}_A^{i-1}$.
    end
end
```

3. Using the BSL package

In this section, we introduce the **BSL** package with an illustrative example available from the package. In the following section we consider a real example, which is also available in the package. Two main functionalities that are offered in the **BSL** package are (1) running an MCMC algorithm with a chosen SL estimator to sample from the approximate posterior distribution, and (2) selecting the penalty parameter with the given SL estimator and shrinkage method. Parallel computing is supported with the R package **foreach** (Kane, Emerson, and Weston 2013; Microsoft Corporation and Weston 2020b) so that the n independent model simulations can be run on a multi-core computer if desired.

3.1. Description of the MA(2) example

Here we consider the MA(2), moving average of order 2, time series model with parameter $\boldsymbol{\theta} = (\theta_1, \theta_2)$. The parameter space is constrained to $\{\boldsymbol{\theta} : -1 < \theta_2 < 1,\ \theta_1 + \theta_2 > -1,\ \theta_1 - \theta_2 < 1\}$ so that the process is stationary and invertible. The model has two favorable properties as an illustrative example: (1) the likelihood is tractable and hence sampling from the true posterior is feasible for comparisons, and (2) the data generation process is fast. The model can be described with the following equation

$$y_t = z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2}, \text{ for } t = 1, \dots, T,$$

where $z_t \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$ for $t = -1, 0, \dots, T$ is white noise. The prior is set to be uniform on the feasible region of the parameter space, and zero elsewhere. We set $T = 50$ and use the raw data as the summary statistic. Note that the likelihood function is exactly multivariate normal with mean zero, $\mathsf{VAR}(y_t) = 1 + \theta_1^2 + \theta_2^2$, $\mathsf{COV}(y_t, y_{t-1}) = \theta_1 + \theta_1\theta_2$, $\mathsf{COV}(y_t, y_{t-2}) = \theta_2$, and with all other covariances being 0. We load the observed data and simulation function with

```
R> data("ma2", package = "BSL")
```

3.2. The model object

Before running any BSL algorithm, we shall first define the model or DGP that will later be used in simulations. The S4 class 'MODEL' contains all the ingredients that constitute a model; namely a simulation function (fnSim), a summary statistic function (fnSum), a point estimate or initial value of the parameter (theta0) and other optional arguments (for example simArgs and fnLogPrior as given below). The first three must be provided for a valid 'MODEL' object. The validity check is automatic upon the creator function newModel. The following command creates a 'MODEL' object for the MA(2) example

```
R> ma2Model <- newModel(fnSim = ma2_sim, fnSum = function(x) x,
+    simArgs = list(T = 50), theta0 = c(0.6, 0.2), fnLogPrior = ma2_prior)

*** initialize "MODEL" ***
has simulation function: TRUE
has summary statistics function: TRUE
has initial guess / point estimate of the parameter: TRUE
running a short simulation test ... success
*** end initialize ***
```

Here we use the built-in simulation function ma2_sim. For simplicity, the summary statistic function returns the raw data without any processing. We set the initial guess of the parameter to be theta0 = c(0.6, 0.2), which will also be used as the starting value in MCMC. We can see in the console message that upon initialization, the newModel creator function automatically checks if the three requisites exist. By default, the creator function also checks if the functions can run properly by running 10 simulations as a short test. The short test can be disabled with flag test = FALSE if desired.

The simulation function

A fast simulation function is important to the efficiency of BSL methods. The package supports two types of simulation functions. One type of simulation function is where multiple simulations are produced with vectorized code. Ideally, if code vectorization can be implemented, it is preferable to speed up the simulation process. However, in most complex applications, it will usually not be possible to vectorize independent simulations. Thus the second type of simulation only generates a single realization of the model.

The simulation function must be provided by the user. We recommend optimizing the simulation function to avoid any inefficiency. If the DGP is complex and time-consuming, we also encourage users to write their own simulation function in C/C++ and use **Rcpp** (Eddelbuettel and François 2011) to call the function within R, as the running time of the algorithm is typically dominated by model simulations. For a reference text on writing functions in **Rcpp**, see Eddelbuettel (2013).

Non-vectorized simulation function. For a non-vectorized simulation function, the model parameter $\boldsymbol{\theta}$ must be supplied as the first argument of the function. Additional arguments can be put after $\boldsymbol{\theta}$ if needed. The output of a non-vectorized simulation function is not restricted to any R class, but must be appropriate to pass to the summary statistic function directly. For example, the function below takes two arguments and returns a single simulation result. The additional parameter T determines the length of the time series, and can be supplied as a list in the model creator (`simArgs = list(T = 50)`).

```
R> ma2_sim

function (theta, TT)
{
    rand <- rnorm(TT + 2)
    y <- rand[3:(TT + 2)] + theta[1] * rand[2:(TT + 1)] + theta[2] *
        rand[1:TT]
    return(y)
}
<bytecode: 0x55dc695279d8>
<environment: namespace:BSL>
```

Vectorized simulation function. The vectorized simulation function is slightly different to the non-vectorized one. It is specified with argument `fnSimVec` in the model creator `newModel`. If a vectorized simulation function is provided, it will override the usage of the non-vectorized function `fnSim`. The input of `fnSimVec` must follow the order of n, $\boldsymbol{\theta}$ and additional arguments. The output can either be a list of the n simulation results or a matrix where each row represents a simulation result. Note that vectorized simulation is incompatible with parallel computing described later. Below is a vectorized simulation function for the MA(2) model.

```
R> ma2_sim_vec

function (n, theta, TT)
{
    rand <- matrix(rnorm(n * (TT + 2)), n, TT + 2)
    y <- rand[, 3:(TT + 2)] + theta[1] * rand[, 2:(TT + 1)]
    +theta[2] * rand[, 1:TT]
    return(y)
}
<bytecode: 0x55dc695f2d58>
<environment: namespace:BSL>
```

The summary statistic function

The summary statistic function takes a simulation result as the first input. Similarly to the simulation function, if the summary statistic function requires additional arguments, they can be stored as a list and passed to `sumArgs` in the model creator function. The output of the summary statistic function must be a d-dimensional vector of summary statistics. For simplicity, we load the identity function that returns the same data as the summary statistic in the MA(2) example.

The prior function

`fnLogPrior` computes the log density of the prior distribution. If the prior function is not provided, an improper flat prior will be used by default. However, in practice, defining a proper prior distribution is always recommended. This function should only take $\boldsymbol{\theta}$ as its input argument, and the output must not be $+\infty$. Note that the normalization constant of the prior distribution can be ignored, as we do below for the MA(2) example. Here the prior is uniform on the defined triangular region of $\boldsymbol{\theta}$.

```
R> ma2_prior

function (theta)
{
    log(theta[2] < 1 & sum(theta) > -1 & diff(theta) > -1)
}
<bytecode: 0x00000203cf6184f8>
<environment: namespace:BSL>
```

3.3. The main function

The primary function of the package is `bsl`, which uses MCMC to draw samples from an approximation to the idealized BSL posterior distribution $p_A(\boldsymbol{\theta}|\boldsymbol{s_y})$. The type of likelihood estimator described in Section 2 (BSL, uBSL, semiBSL and rBSL) can be specified by the argument `method`. Shrinkage estimation on the covariance matrix in BSL and the correlation matrix in semiBSL can be specified with `shrinkage`. The minimal requirements of the `bsl` function include the observed data (`y`), the number of simulations for synthetic likelihood estimation (`n`), total number of MCMC iterations (`M`) and the covariance matrix of the normal random walk Metropolis algorithm (`covRandWalk`). For the MA(2) example, a minimalistic call to the main function that runs MCMC BSL (using the estimator of Equation 1) is given by the following command,

```
R> resultMa2BSL <- bsl(y = ma2$data, n = 500, M = 100000, model = ma2Model,
+    covRandWalk = ma2$cov, method = "BSL")
```

Similarly, we can run MCMC uBSL, semiBSL and rBSL by changing the `method` option,

```
R> resultMa2uBSL <- bsl(y = ma2$data, n = 500, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "uBSL")
R> resultMa2SemiBSL <- bsl(y = ma2$data, n = 500, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "semiBSL")
```

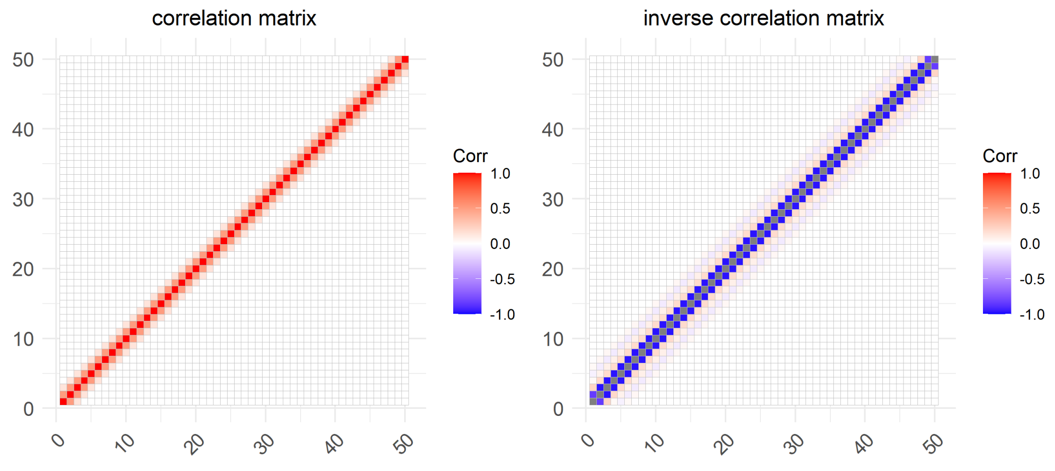

Figure 1: The correlation matrix (left) and inverse correlation matrix (right) of the true likelihood function for the MA(2) example.

```
R> resultMa2rBSLM <- bsl(y = ma2$data, n = 500, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "BSLmisspec",
+    misspecType = "mean", tau = 0.5)
R> resultMa2rBSLV <- bsl(y = ma2$data, n = 500, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "BSLmisspec",
+    misspecType = "variance", tau = 0.5)
```

The normal random walk covariance matrix, `covRandWalk`, should be defined based on the parameterization used in the MCMC sampling; if a parameter transformation is employed, the covariance matrix should be specified accordingly.

Shrinkage of the likelihood estimator

Implementation of shrinkage methods described in Section 2.5 can be specified with arguments `shrinkage` (`"glasso"` for the graphical lasso estimator and `"Warton"` for Warton's estimator) and `penalty` for the penalty value (λ in the graphical lasso or Warton's estimator). Note that the shrinkage takes place on the covariance matrix for method `"BSL"`, and on the correlation matrix (of the Gaussian copula) for method `"semiBSL"`.

Shrinkage should only be used when there is a relatively large number of entries near zero in the precision matrix for graphical lasso and covariance matrix for Warton's approach. This can be checked by inspecting the inverse correlation matrix or the correlation matrix at a representative parameter value. In the MA(2) example, we are able to calculate the true covariance matrix. For more complex models the covariance matrix can be estimated with a large number of model simulations at a reasonable parameter value. Here we use the `ggcorrplot` function in **ggcorrplot** (Kassambara 2019) to visualize the matrices. The visualization of the correlation matrix and inverse correlation matrix for the MA(2) example are shown in Figure 1.

Both of the figures suggest that the correlation and inverse correlation matrices are sparse. It is also recommended to check the level of sparsity by computing the proportion of partial correlations below a certain threshold. For example, 81% of the partial correlations between the summary statistics of the MA(2) example are below 0.01. Thus, for this example, shrinkage estimation is expected to reduce the number of model simulations required for estimating the synthetic likelihood whilst not sacrificing much on posterior accuracy. We will describe how to select the penalty parameter in our **BSL** package in Section 3.5. The following code runs MCMC BSL with the graphical lasso and Warton's shrinkage estimator.

```
R> resultMa2BSLasso <- bsl(y = ma2$data, n = 300, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "BSL",
+    shrinkage = "glasso", penalty = 0.027)
R> resultMa2BSLWarton <- bsl(y = ma2$data, n = 300, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "BSL",
+    shrinkage = "Warton", penalty = 0.75)
```

When there is significant correlation between summaries, the whitening transformation can be effective in allowing a significant amount of shrinkage when employing the Warton's estimator, while still maintaining a reasonable level of accuracy. Whitening is not currently supported for semiBSL. If a whitening transformation is desired, a whitening matrix needs to be estimated prior to running the `bsl` function. This can be done with the function `estimateWhiteningMatrix`. The example code below estimates a whitening matrix using the `"PCA"` method (as suggested in Section 2.5) with 20000 simulations at a parameter value of $\boldsymbol{\theta} = (0.6, 0.2)^\top$.

```
R> W <- estimateWhiteningMatrix(20000, ma2Model, method = "PCA",
+    thetaPoint = c(0.6, 0.2))
```

Supplying the estimated whitening matrix into the `bsl` function under argument `whitening` enables the whitening transformation. Alternatively, simply setting `whitening = TRUE` automatically estimates a whitening matrix with 1000 model simulations at the parameter value provided in `model`. The following code runs the standard BSL method with the Warton's shrinkage estimator and whitening transformation.

```
R> resultMa2BSLWhitening <- bsl(y = ma2$data, n = 300, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "BSL",
+    shrinkage = "Warton", penalty = 0.6, whitening = W)
```

Parameter transformation

If the prior distribution is bounded or the normal random walk cannot explore the parameter space efficiently, parameter transformation is recommended. Recall $p(\boldsymbol{\theta})$ is the prior distribution for $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^p$. Suppose the parameters are independent and bounded, the prior function can be decomposed as

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{p} p_i(\theta_i), \text{ for } \theta_i \in (a_i, b_i),$$

where a_i and b_i are the lower and upper bound for θ_i. A straightforward but fruitful 1–1 transformation that maps the range of the parameter to the real line is the logit transformation, which is given by

$$\tilde{\theta}_i = \log \frac{\theta_i - a_i}{b_i - \theta_i}, \text{ for } i = 1, \ldots, p.$$

If the parameter is only bounded on one side, the transformation degenerates to a log transformation, i.e., $\tilde{\theta}_i = \log(b_i - \theta_i)$ if a_i is $-\infty$ and $\tilde{\theta}_i = \log(\theta_i - a_i)$ if b_i is ∞. The `bsl` function provides an easy-to-use log/logit transformation for independent and bounded parameters. The argument `logitTransformBound` is a p by 2 matrix containing the lower and upper bounds for each parameter. The only argument that needs to be changed is `covRandWalk`. There is no need to do any reparameterization of the prior or simulation functions.

It is also possible to code a customized parameter transformation by editing the simulation function directly. In this case, the prior function should also be changed subject to the reparameterization.

Parallel computation

As a simulation-based method, the computational cost of BSL is mostly driven by the speed of the simulation process. Parallel computation is vital in complex applications where vectorization is not possible. In our **BSL** package, the n independent model simulations can be distributed across workers on a multi-core machine. The R package **doParallel** (Microsoft Corporation and Weston 2020a) provides a way to set up the CPU cores for parallel computation. For example, the following code determines all the available cores of a CPU and sets up the clusters for parallel jobs as well as registers the backend for persistent reproducible parallel loops.

```
R> ncores <- detectCores()
R> cl <- makeCluster(ncores - 1)
R> registerDoParallel(cl)
R> registerDoRNG(1)
```

This should be turned on prior to running the main `bsl` function if parallel computing is desired. To enable parallel computing within **BSL**, set `parallel` to `TRUE` in the `bsl` function. We utilize the function `foreach` of package **foreach** (Microsoft Corporation and Weston 2020b) to run parallel simulations in our **BSL** package. All other arguments supported by `foreach` can be passed with `parallelArgs` in the `bsl` and `selectPenalty` functions. For illustration, the following code runs the standard MCMC BSL with parallel computing,

```
R> resultMa2BSLParallel <- bsl(y = ma2$data, n = 500, M = 100000,
+    model = ma2Model, covRandWalk = ma2$cov, method = "BSL",
+    parallel = TRUE, verbose = TRUE)
```

Note that parallel computing introduces additional communication time between workers. If the model simulation process is straightforward, parallel computing might increase the overall running time rather than reduce it. Thus this is not recommended in the MA(2) example. Vectorization can be more effective in such cases. Once parallel computing is completed, the following code shuts down the parallel cores.

```
R> stopCluster(cl)
R> registerDoSEQ()
```

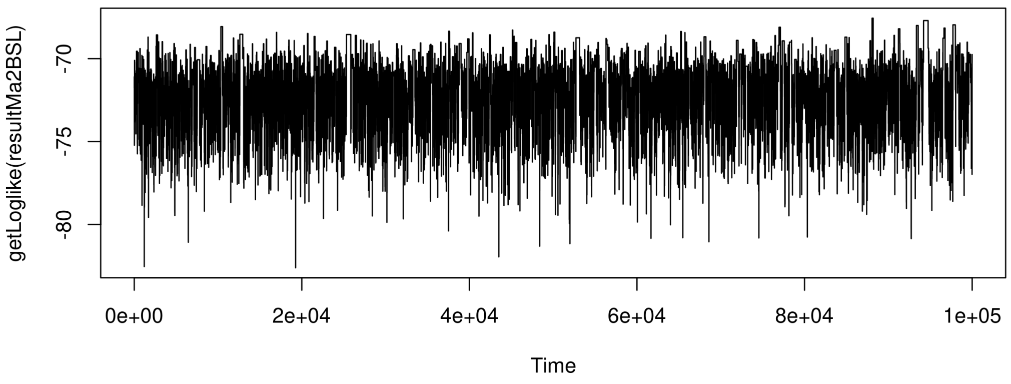

Figure 2: Trace plot of the synthetic log-likelihood estimates for the MA(2) example.

3.4. Interpret and visualize the BSL result

The output of the function `bsl` is saved as an S4 object '`BSL`', which includes `theta` (approximate posterior samples), `loglike` (the MCMC chain of estimated log-likelihood values), `acceptanceRate` (acceptance rate of MCMC) for inspection of the Markov chain, as well as several other arguments which help to analyze the result. A full list of the returned values can be checked with `help(bsl, package = "BSL")`. We provide the basic `show`, `summary` and `plot` methods for the '`BSL`' class.

The following code provides some common MCMC diagnostics for the MCMC BSL result of the MA(2) example. The column title ESS in the `summary` result stands for the effective sample size of the approximate posterior samples, as estimated by the R package **coda** (Plummer, Best, Cowles, and Vines 2006). Here we only show the trace plot of the synthetic log-likelihood estimates (Figure 2), however we also recommend other specialized R packages (such as **coda**, Plummer *et al.* 2006, and **plotMCMC**, Magnusson and Stewart 2020) for other visualizations and quantitative diagnostics of MCMC convergence.

```
R> resultMa2BSL


Call:
bsl(y = ma2$data, n = 500, M = 1e+05, model = ma2Model, covRandWalk = ma2$cov,
    method = "BSL", verbose = 0L)

Summary of theta:
       theta[1]            theta[2]
[1,]  Min.   :-0.01612    Min.   :-0.57911
[2,]  1st Qu.: 0.47552    1st Qu.: 0.00946
[3,]  Median : 0.58007    Median : 0.10854
[4,]  Mean   : 0.58050    Mean   : 0.11086
[5,]  3rd Qu.: 0.68505    3rd Qu.: 0.21559
[6,]  Max.   : 1.17830    Max.   : 0.65760
Summary of loglikelihood:
   Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 -82.61   -72.84   -71.59   -71.69   -70.37   -67.56
```

```
Acceptance Rate:
[1]  0.1463
Early Rejection Rate:
[1]  0.00114

R> summary(resultMa2BSL)

            n acc. rate (%)  ESS theta[1]  ESS theta[2]
          500            15          1970          1686

R> plot.ts(getLoglike(resultMa2BSL))
```

The `plot` function for '`BSL`' draws the approximate univariate posterior distribution for each parameter with either the default R **graphics** (Figure 3) or **ggplot2** (Figure 4, Wickham 2016).

```
R> plot(resultMa2BSL, which = 1, thetaTrue = c(0.6, 0.2), thin = 30)
R> mtext("Approximate Univariate Posteriors", line = 1, cex = 1.5)
R> plot(resultMa2BSL, which = 2, thetaTrue = c(0.6, 0.2), thin = 30,
+    options.density = list(color = "coral4", fill = "coral", alpha = 0.5),
+    options.theme = list(panel.background = element_rect(fill = "beige"),
+      plot.margin = grid::unit(rep(0.05, 4), "npc")))
```

It is often of interest to compare multiple `bsl` results at the same time, for example, the following code summarizes the BSL, uBSL, semiBSL, BSL with graphical lasso and BSL with Warton results.

```
R> ma2Results <- list(resultMa2BSL, resultMa2uBSL, resultMa2SemiBSL,
+    resultMa2rBSLM, resultMa2rBSLV, resultMa2BSLasso,
+    resultMa2BSLWarton, resultMa2BSLWhitening)
R> names(ma2Results) <- c("BSL", "uBSL", "semiBSL", "rBSLM", "rBSLV",
+    "BSLasso", "BSLWarton", "BSLWhitening")
R> t(sapply(ma2Results, summary))

               n acc. rate (%) ESS theta[1] ESS theta[2]
BSL          500            15         1970         1686
uBSL         500            14         2129         1992
semiBSL      500            13         1437         1353
rBSLM        500            23         2158         1788
rBSLV        500            38         3632         3328
BSLasso      300            28         2556         2431
BSLWarton    300            30         3139         2226
BSLWhitening 300            25         3663         3025
```

It can be seen that BSL, uBSL and semiBSL have very similar acceptance rates and ESS values, which indicates that their performance is very close. Both shrinkage methods increase the acceptance rate and ESS with smaller n compared to BSL.

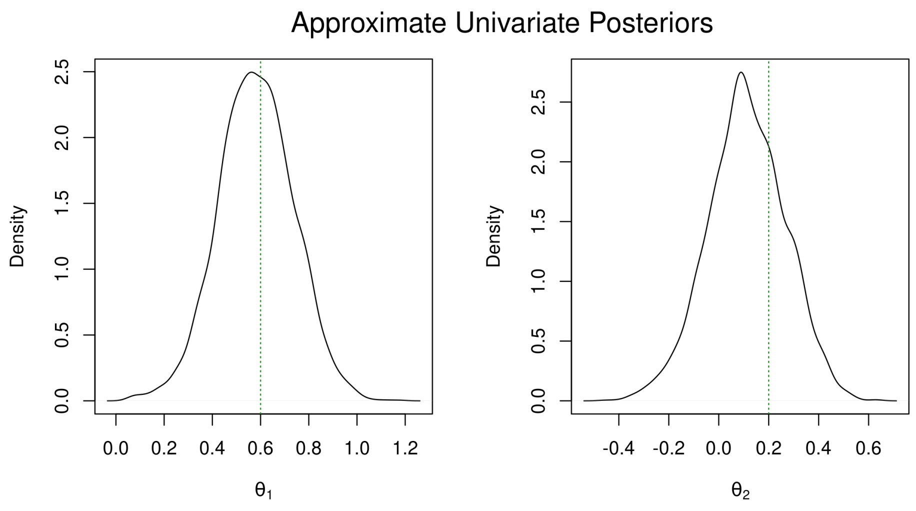

Figure 3: Approximate posterior distributions using the BSL estimator for the MA(2) example with **graphics**.

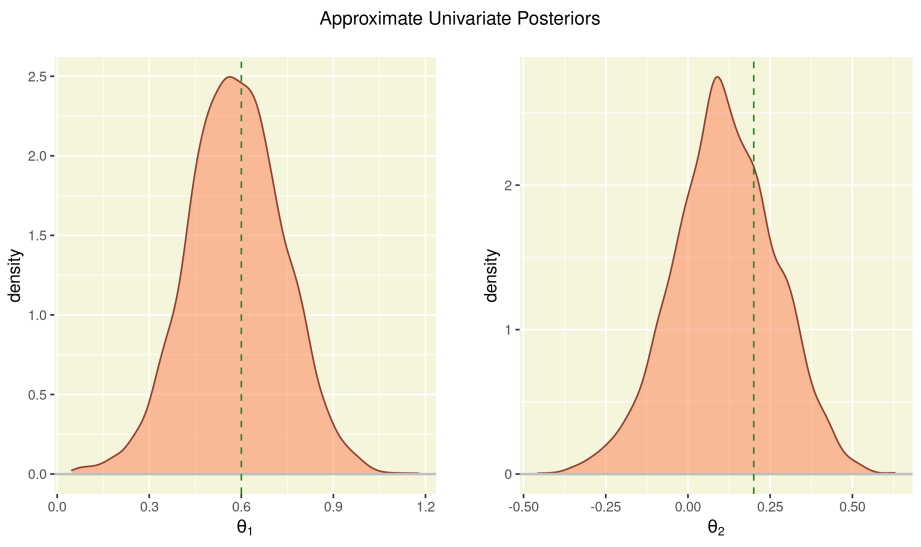

Figure 4: Approximate posterior distributions using the BSL estimator for the MA(2) example with **ggplot2**.

The function `combinePlotsBSL` plots multiple approximate posterior distributions in the same figure. The arguments are similar to the `plot` function of ‘`BSL`’, except the object to be plotted should be a list of multiple ‘`BSL`’ objects. Figures 5 and 6 give an example of the `combinePlotsBSL` function.

```
R> par(mar = c(5, 4, 1, 2), oma = c(0, 1, 2, 0))
R> combinePlotsBSL(ma2Results, which = 1, thetaTrue = c(0.6, 0.2),
+    thin = 30, lty = 1:8, lwd = rep(2, 8), legendNcol = 4)
R> mtext("Approximate Univariate Posteriors", outer = TRUE, cex = 1.5)
```

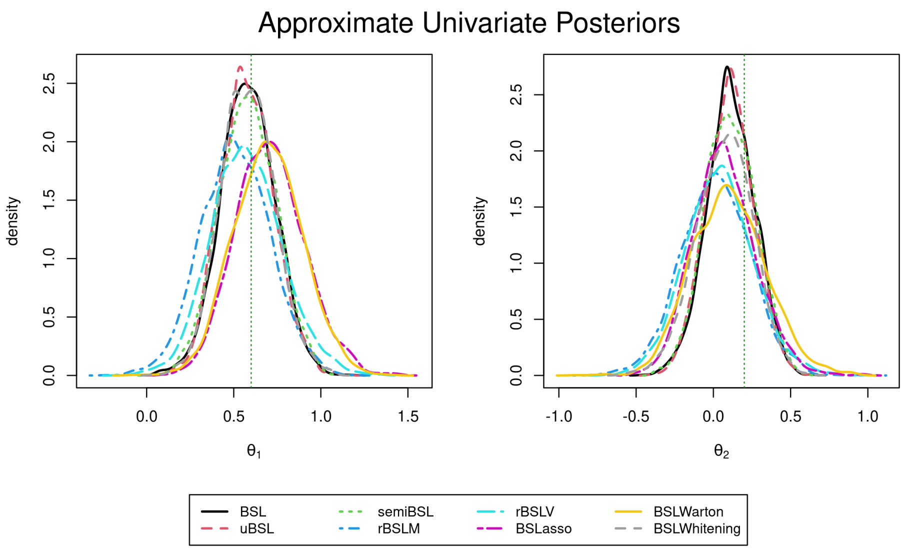

Figure 5: Approximate posterior distributions using various BSL estimators for the MA(2) example with **graphics**.

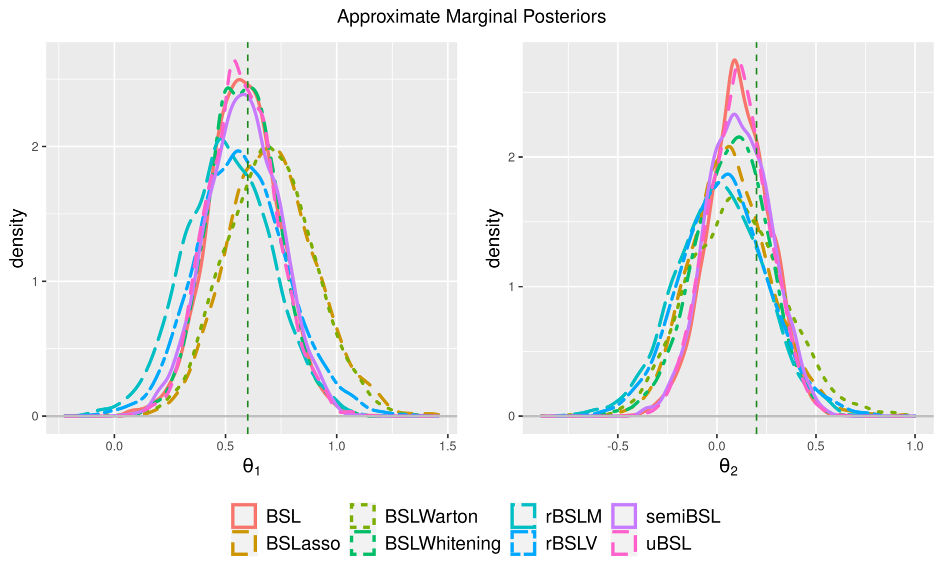

Figure 6: Approximate posterior distributions using various BSL estimators for the MA(2) example with **ggplot2**.

```
R> combinePlotsBSL(ma2Results, which = 2, thetaTrue = c(0.6, 0.2),
+    thin = 30, options.linetype = list(values = 1:8),
+    options.size = list(values = rep(1, 8)),
+    options.theme = list(plot.margin = grid::unit(rep(0.03, 4), "npc"),
+      axis.title = ggplot2::element_text(size = 12),
+      axis.text = ggplot2::element_text(size = 8),
+      legend.text = ggplot2::element_text(size = 12))
```

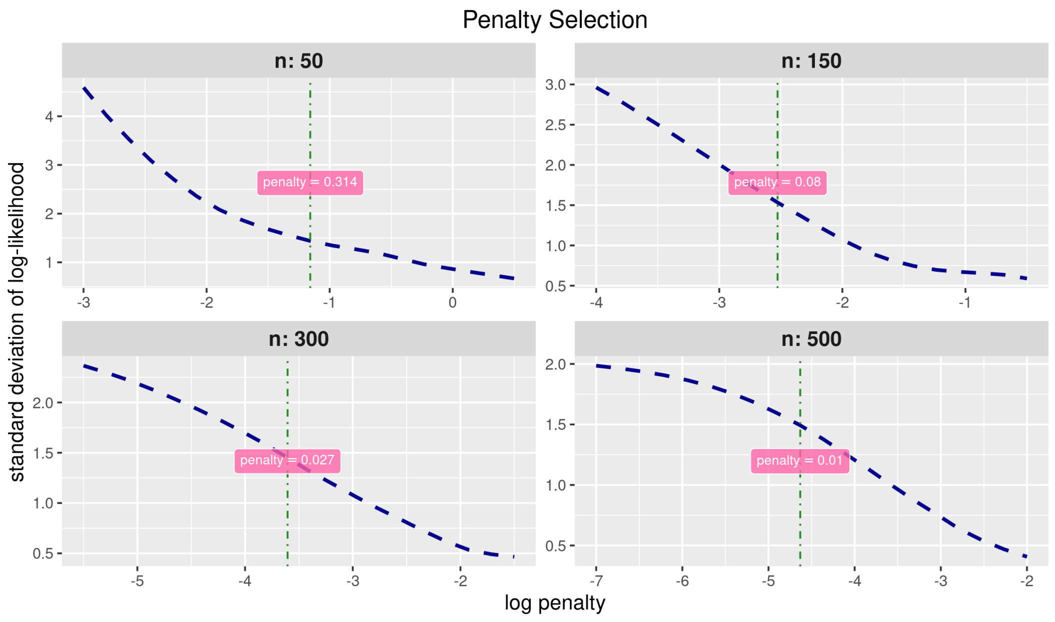

Figure 7: Penalty selecting result for BSL with graphical lasso for the MA(2) example.

3.5. Selecting the penalty parameter for shrinkage

If shrinkage is desired in estimating the SL, the corresponding penalty parameter value must be selected prior to running the `bsl` function. This can be done via the `selectPenalty` function. Multiple choices for the number of simulations n can be tested at the same time; simulations from the largest value of n are re-used for smaller values of n by subsampling. The basic arguments of `selectPenalty` include the summary statistic of the observed data `ssy`; a vector of the number of simulations n to test; a list of the candidate penalty values `lambda_all` corresponding to each n; a point estimate of the parameter `theta`; the number of repeats `M`; the target standard deviation `sigma`; the model of interest `model`; the SL estimator `method` and the shrinkage estimation method `shrinkage`. Example code is given below (Figure 7) for selecting the λ of glasso with the standard BSL estimator for the MA(2) example.

```
R> ssy <- ma2_sum(ma2$data)
R> lambda_all <- list(exp(seq(-3, 0.5, length.out = 20)),
+    exp(seq(-4, -0.5, length.out = 20)),
+    exp(seq(-5.5, -1.5, length.out = 20)),
+    exp(seq(-7, -2, length.out = 20)))
R> set.seed(100)
R> selectPenaltyMA2 <- selectPenalty(ssy = ssy, n = c(50, 150, 300, 500),
+    lambda_all, theta = c(0.6, 0.2), M = 100, sigma = 1.5,
+    model = ma2Model, method = "BSL", shrinkage = "glasso")
```

The result is of S3 class ‘`penbsl`’. This class has support for `print` and `plot` functions.

```
R> selectPenaltyMA2

Call:
selectPenalty(ssy = ssy, n = c(50, 150, 300, 500), lambda = lambda_all,
    M = 100, sigma = 1.5, model = ma2Model, theta = c(0.6, 0.2),
    method = "BSL", shrinkage = "glasso", verbose = 0L)

Penalty selected based on the standard deviation of the loglikelihood:
     n penalty stdLoglike
11  50 0.31415       1.44
29 150 0.07995       1.53
50 300 0.02718       1.45
70 500 0.00974       1.49

R> plot(selectPenaltyMA2)
```

4. Toad example

In this section, we use the **BSL** package to find posterior distributions for a complex model fitted to a real dataset. The model is proposed by Marchand, Boenke, and Green (2017) to simulate the movements of an amphibian called Fowler's Toads. Marchand *et al.* (2017) also used ABC for parameter inference to bypass the evaluation of the intractable likelihood. Real data are available from the supplementary material of Marchand *et al.* (2017).

4.1. Model description

The model generally assumes that the toads hide in their refuge sites during the day, and only move around for foraging at night. At the end of the foraging period, the location of a toad is Δx away from the previous refuge site. Here, Δx, also called the overnight displacement, follows a Lévy-alpha stable distribution with stability parameter $0 < \alpha \leq 2$ and scale parameter $\gamma > 0$. The distribution is known to be increasingly heavy-tailed when α is less than 2. Upon the end of nighttime foraging, the toad either returns to one of the previous refuge sites or builds a new one at the current location.

The probability of return is given by p_0. Marchand *et al.* (2017) develop three different return models (random return, nearest return and distance-based return) to describe the return strategy. Here for illustrative purposes, we use the random return model where the refuge site is selected at random from all the previous sites. Multiple visits increase the chance of being selected.

We use the real data provided in the supplementary material of Marchand *et al.* (2017). The GPS locations are collected in the daytime when the toads are assumed to rest in the refuge sites. For simplicity, the 2D locations are projected to a straight line, so that locations can be represented by scalar values. The observation matrix $\boldsymbol{Y}$ contains the locations of $n_t = 66$ toads over $n_d = 63$ days, i.e., $\boldsymbol{Y}$ is of dimension $n_t \times n_d$. Roughly 81% of the real observation matrix $\boldsymbol{Y}$ is made up of missing entries. We also set the same entries as `NA` for new simulations. The parameter of interest is $\boldsymbol{\theta} = (\alpha, \gamma, p_0)^\top$. We place a uniform prior over $(1, 2) \times (0, 100) \times (0, 0.9)$ in this example.

We reduce the observation matrix down to four sets of displacements of lags $1, 2, 4$ and 8 days. For instance, the displacements for a lag of one day can be written as $\boldsymbol{y}_1 = \{|\Delta y| = |\boldsymbol{Y}_{i,j} - \boldsymbol{Y}_{i+1,j}|; 1 \leq i \leq n_d - 1, 1 \leq j \leq n_t\}$. Following Marchand *et al.* (2017), we classify the displacements as returns and non-returns by whether the absolute distance of a displacement is less than 10 meters or not. We consider a summary statistic that contains the frequency of returns, the median of the non-returns, and the log differences of adjacent p quantiles, where $p = 0, 0.1, \ldots, 1$.

4.2. Approximate the posterior with BSL

As usual, we need to set up the 'MODEL' object for the toad example. The simulation function toad_sim calls an **Rcpp** function that uses C++ in the backend. The summary statistic function toad_sum computes the quantile summary statistic just described in Section 4.1.

```
R> data("toad", package = "BSL")
R> toadModel <- newModel(fnSim = toad_sim, fnSum = toad_sum,
+    theta0 = toad$theta0, fnLogPrior = toad_prior,
+    simArgs = toad$sim_args_real,
+    thetaNames = expression(alpha, gamma, p[0]))

*** initialize "MODEL" ***
has simulation function: TRUE
has summary statistics function: TRUE
has initial guess / point estimate of the parameter: TRUE
running a short simulation test ... success
*** end initialize ***
```

The printed messages indicate that model is a valid 'MODEL' object for **BSL**. However, it is also recommended to look at the density distributions of the summary statistic to detect any possible problems or malfunctions associated with the model. Figure 8 plots the marginal KDE distributions of 1000 simulated summary statistics. We will utilize parallel computation when possible in the toad example. Suppose we have set up the CPU cores for parallel computing as described in the section on parallel computing of Section 3.2.

```
R> sim <- simulation(toadModel, n = 1000, theta = toad$theta0, seed = 10,
+    parallel = TRUE)
R> par(mfrow = c(6, 8), mar = c(3, 1.5, 0.5, 0.5))
R> for (i in 1:48) plot(density(sim$ssx[, i]), main = "", xlab = "")
```

Figure 8 implies that the marginal summary statistics are roughly normal. Before running the bsl function, we also need to define the following: covWalk, the covariance matrix for normal random walk, and paraBound, the matrix of the upper and lower bounds for a logit transformation (as described in the section on parameter transformation in Section 3.3).

```
R> covWalk <- toad$cov
R> paraBound <- matrix(c(1, 2, 0, 100, 0, 0.9), 3, 2, byrow = TRUE)
```

Now we can run standard BSL, uBSL, semiBSL, rBSL-M and rBSL-V with the following code.

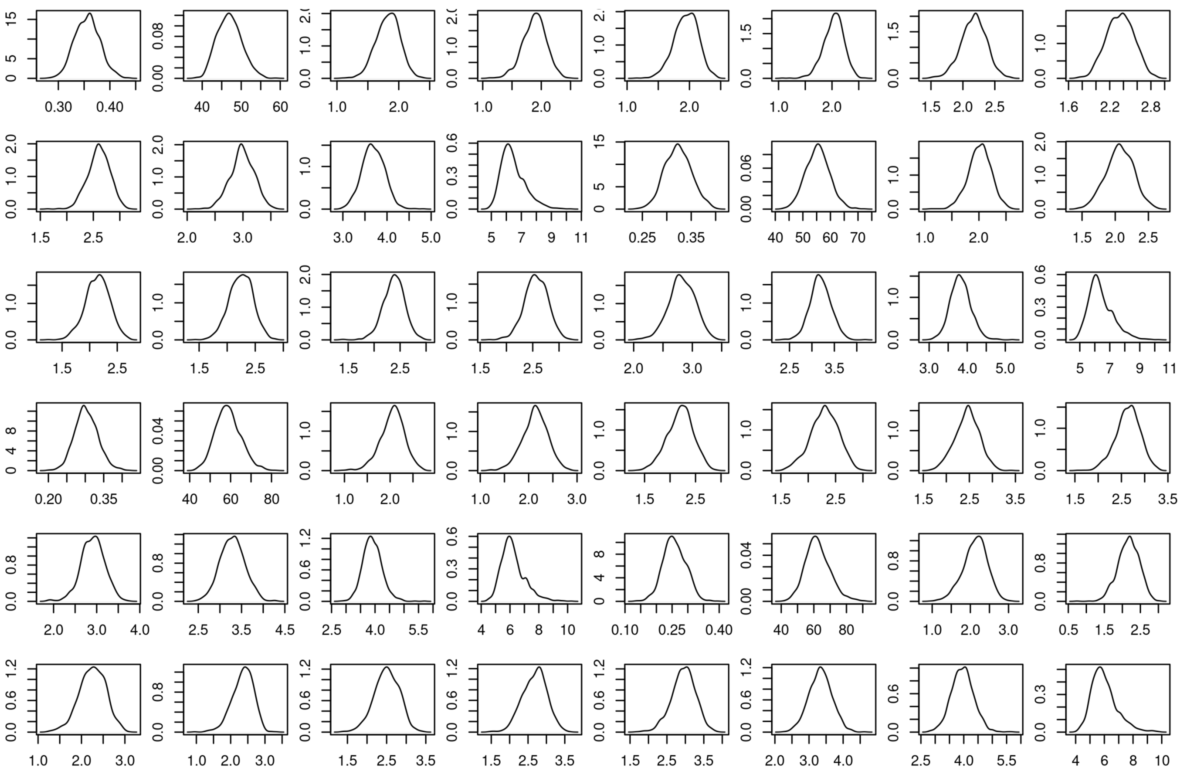
Figure 8: Marginal distributions of the summary statistic for the toad example.

```
R> resultToadBSL <- bsl(toad$data_real, n = 500, M = 50000,
+    model = toadModel, method = "BSL", covRandWalk = covWalk,
+    logitTransformBound = paraBound, parallel = TRUE)
R> resultToaduBSL <- bsl(toad$data_real, n = 500, M = 50000,
+    model = toadModel, method = "uBSL", covRandWalk = covWalk,
+    logitTransformBound = paraBound, parallel = TRUE)
R> resultToadSemiBSL <- bsl(toad$data_real, n = 500, M = 50000,
+    model = toadModel, method = "semiBSL", covRandWalk = covWalk,
+    logitTransformBound = paraBound, parallel = TRUE)
R> resultToadrBSLM <- bsl(toad$data_real, n = 500, M = 50000,
+    model = toadModel, method = "BSLmisspec", misspecType = "mean",
+    tau = 0.5, covRandWalk = covWalk, logitTransformBound = paraBound,
+    parallel = TRUE)
R> resultToadrBSLV <- bsl(toad$data_real, n = 500, M = 50000,
+    model = toadModel, method = "BSLmisspec", misspecType = "variance",
+    tau = 0.5, covRandWalk = covWalk, logitTransformBound = paraBound,
+    parallel = TRUE)
R> resultToadBSLWhitening <- bsl(toad$data_real, n = 500, M = 50000,
+    model = toadModel, method = "BSL", shrinkage = "Warton",
+    penalty = 0.12, whitening = TRUE,  covRandWalk = covWalk,
+    logitTransformBound = paraBound, parallel = TRUE)
```

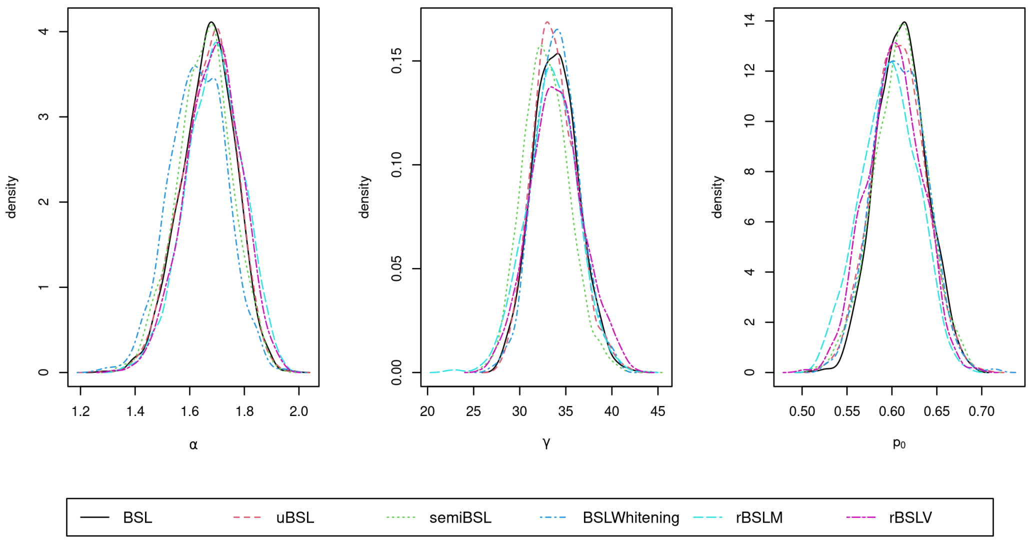

Figure 9: Approximate marginal posterior distributions using various BSL methods for the toad example.

We may summarize the above BSL results into a matrix.

```
R> toadResults <- list(resultToadBSL, resultToaduBSL, resultToadSemiBSL,
+    resultToadBSLWhitening, resultToadrBSLM, resultToadrBSLV)
R> names(toadResults) <- c("BSL", "uBSL", "semiBSL", "BSLWhitening",
+    "rBSLM", "rBSLV")
R> t(sapply(toadResults, summary))
```

```
               n acc. rate (%) ESS alpha ESS gamma ESS p[0]
BSL          500            33      1411       941      604
uBSL         500            33      1427       955      533
semiBSL      500            29      1075       775      417
BSLWhitening 500            60      2409      1794      976
rBSLM        500            32      1068       645      476
rBSLV        500            50      1983      1066      791
```

The overlaid marginal posteriors plot is shown in Figure 9.

```
R> combinePlotsBSL(toadResults, which = 1, thin = 30, burnin = 5000)
```

We also compare our results from vanilla BSL with the ABC approach of Marchand *et al.* (2017). We obtain similar posterior medians for all three parameters, with the following BSL results (with ABC results shown in parentheses) for α, γ and p_0, respectively: 1.67 (1.70), 34 (34) and 0.61 (0.60). The 95% credible interval results are 1.46–1.85 (1.41–1.94), 29–39 (26–42) and 0.55–0.67 (0.53–0.65). It can be seen that the intervals are similar for p_0, but BSL leads to more precise estimates for α and γ based on the tighter credible intervals. The ABC approach of Marchand *et al.* (2017) uses a lower dimensional summary statistic as often demanded by ABC, which can incur information loss. In contrast, BSL can be run feasibly for the higher dimensional summary statistic, and thus capturing more information contained in the full dataset, reducing uncertainty in the parameter estimates.

5. Discussion

This paper has presented the first comprehensive software package for Bayesian synthetic likelihood methods. The package implements four different types of synthetic likelihood estimators (standard, unbiased, semi-parametric and "robust") and includes functionality for two different types of shrinkage covariance estimators to reduce the number of required model simulations. The package includes functions for extracting useful statistics and visualizations of the results after running `bsl`. It also includes four built-in examples to illustrate the functionality of the package.

Apart from the MA(2) and toad example described in Sections 3 and 4, respectively, the package also include two other examples: a discrete-time stochastic cell biology model, `cell`; and the multivariate g-and-k quantile distribution `mgnk` (Drovandi and Pettitt 2011). The former example features a high dimensional summary statistic, while the latter example can involve a large number of parameters. Additional descriptions and example code can be found in the package documentation.

We use multivariate normal random walk MCMC as the sampling method to explore the parameter space for all the current BSL methods in the package, which means that sampling with **BSL** can be slow when model simulation is computationally intensive and potentially inefficient when there are a large number of parameters. More sophisticated proposal distributions for MCMC BSL would be an interesting research direction for future work. Unfortunately, alternative Monte Carlo schemes like rejection sampling and sequential Monte Carlo (Del Moral *et al.* 2006) are less appealing in BSL than in ABC because BSL's normality assumption can fail in regions with very low posterior support. Considering a BSL analogue of the Hamiltonian Monte Carlo (HMC) ABC algorithm of Meeds, Leenders, and Welling (2015) may be an interesting direction for future work. The downside of HMC in the likelihood-free context is that efficiency is reduced by the requirement to estimate rather than directly evaluate the gradient of the log-likelihood. Future research for this package could consider incorporating the variational Bayesian synthetic likelihood methods (Ong *et al.* 2018a; Ong, Nott, Tran, Sisson, and Drovandi 2018b), which scale to a large number of summary statistics and/or parameters at the expense of assuming a multivariate normal approximation of the posterior.

We believe that our package complements other likelihood-free packages in the literature. BSL has some appealing features relative to ABC, however there are applications where the distribution of the summary statistic will not be regular enough for BSL to be effective, and ABC (and its corresponding packages) may be preferred in these cases.

The focus of this paper is parameter estimation in the intractable likelihood setting. Because BSL uses summary statistics, it can suffer from the same issues as ABC in the model section setting (Robert, Cornuet, Marin, and Pillai 2011). In particular, it can be difficult to find statistics informative about discriminating between competing models. In order to increase the chance of identifying useful statistics, the ABC approaches of Prangle, Fearnhead, Cox, Biggs, and French (2014) and Pudlo, Marin, Estoup, Cornuet, Gautier, and Robert (2016) consider an initial large number of summary statistics. To reduce the dimension, they use classification methods estimated from a large number of (model indicator, parameter, statistic) triples simulated from a reference distribution. Then, ABC model choice proceeds using the lower dimensional summary statistic produced by the classification method. Extending BSL to the model selection setting requires further research.

We welcome R developers worldwide to help contribute to the **BSL** package. We have made the source code available at `https://github.com/LeahPrice/BSL` to allow other researchers to contribute to the **BSL** package.

Computational details

The results in this paper were obtained using R 4.0.3 with the **BSL** 3.2.3 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`.

Acknowledgments

CD was supported by the Australian Research Council. LFS would like to acknowledge support from EPSRC grant EP/S00159X/1. The authors are also affiliated with the QUT Centre for Data Science and the Australian Research Council Centre of Excellence for Mathematical and Statistics Frontiers (ACEMS). Computational resources and services used in this work were provided by the HPC and Research Support Group, Queensland University of Technology, Brisbane, Australia.

References

An Z, Nott DJ, Drovandi C (2020). "Robust Bayesian Synthetic Likelihood via a Semi-Parametric Approach." *Statistics and Computing*, **30**(3), 543–557. `doi:10.1007/s11222-019-09904-x`.

An Z, South LF, Drovandi CC (2022). ***BSL**: Bayesian Synthetic Likelihood.* R package version 3.2.3, URL `https://CRAN.R-project.org/package=BSL`.

An Z, South LF, Nott DJ, Drovandi CC (2019). "Accelerating Bayesian Synthetic Likelihood with the Graphical Lasso." *Journal of Computational and Graphical Statistics*, **28**(2), 471–475. `doi:10.1080/10618600.2018.1537928`.

Andrieu C, Doucet A, Holenstein R (2010). "Particle Markov Chain Monte Carlo Methods." *Journal of the Royal Statistical Society B*, **72**(3), 269–342. `doi:10.1111/j.1467-9868.2009.00736.x`.

Andrieu C, Roberts GO (2009). "The Pseudo-Marginal Approach for Efficient Monte Carlo Computations." *The Annals of Statistics*, **37**(2), 697–725. `doi:10.1214/07-aos574`.

Barbu CM, Sethuraman K, Billig EMW, Levy MZ (2018). "Two-Scale Dispersal Estimation for Biological Invasions via Synthetic Likelihood." *Ecography*, **41**(4), 661–672. `doi:10.1111/ecog.02575`.

Blum MGB (2010). "Approximate Bayesian Computation: A Nonparametric Perspective." *Journal of the American Statistical Association*, **105**(491), 1178–1187. doi:10.1198/jasa.2010.tm09448.

Boudt K, Cornelissen J, Croux C (2012). "The Gaussian Rank Correlation Estimator: Robustness Properties." *Statistics and Computing*, **22**(2), 471–483. doi:10.1007/s11222-011-9237-0.

Chen P, Ye ZS, Zhai Q (2020). "Parametric Analysis of Time-Censored Aggregate Lifetime Data." *IISE Transactions*, **52**(5), 516–527. doi:10.1080/24725854.2019.1628374.

Csillery K, Francois O, Blum MGB (2012). "**abc**: An R Package for Approximate Bayesian Computation (ABC)." *Methods in Ecology and Evolution*, **3**(3), 475–479. doi:10.1111/j.2041-210x.2011.00179.x.

Del Moral P, Doucet A, Jasra A (2006). "Sequential Monte Carlo Samplers." *Journal of the Royal Statistical Society B*, **68**(3), 411–436. doi:10.1111/j.1467-9868.2006.00553.x.

Doucet A, Pitt MK, Deligiannidis G, Kohn R (2015). "Efficient Implementation of Markov Chain Monte Carlo When Using an Unbiased Likelihood Estimator." *Biometrika*, **102**(2), 295–313. doi:10.1093/biomet/asu075.

Drovandi CC, Grazian C, Mengersen K, Robert C (2018). "Approximating the Likelihood in Approximate Bayesian Computation." In SA Sisson, Y Fan, M Beaumont (eds.), *Handbook of Approximate Bayesian Computation*. Chapman & Hall/CRC. doi:10.1201/9781315117195.

Drovandi CC, Pettitt AN (2011). "Likelihood-Free Bayesian Estimation of Multivariate Quantile Distributions." *Computational Statistics & Data Analysis*, **55**(9), 2541–2556. doi:10.1016/j.csda.2011.03.019.

Drovandi CC, Pettitt AN, Lee A (2015). "Bayesian Indirect Inference Using a Parametric Auxiliary Model." *Statistical Science*, **30**(1), 72–95. doi:10.1214/14-sts498.

Dutta R, Schoengens M, Pacchiardi L, Ummadisingu A, Widmer N, Künzli P, Onnela JP, Mira A (2021). "**ABCpy**: A High-Performance Computing Perspective to Approximate Bayesian Computation." *Journal of Statistical Software*, **100**(7), 1–38. doi:10.18637/jss.v100.i07.

Eddelbuettel D (2013). *Seamless R and C++ Integration with* ***Rcpp***. Springer-Verlag.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Everitt RG (2017). "Bootstrapped Synthetic Likelihood." arXiv:1711.05825 [stat.CO], URL https://arxiv.org/abs/1711.05825.

Fasiolo M, Wood S (2021). ***synlik**: Synthetic Likelihood Methods for Intractable Likelihoods*. R package version 0.1.4, URL https://CRAN.R-project.org/package=synlik.

Frazier DT, Drovandi C (2021). "Robust Approximate Bayesian Inference with Synthetic Likelihood." *Journal of Computational and Graphical Statistics*, **30**(4), 958–976. doi:10.1080/10618600.2021.1875839.

Frazier DT, Nott DJ, Drovandi C, Kohn R (2021). "Bayesian Inference Using Synthetic Likelihood: Asymptotics and Adjustments." arXiv:1902.04827 [stat.CO], URL `https://arxiv.org/abs/1902.04827/v4`.

Frazier DT, Robert CP, Rousseau J (2020). "Model Misspecification in Approximate Bayesian Computation: Consequences and Diagnostics." *Journal of the Royal Statistical Society B*, **82**(2), 421–444. `doi:10.1111/rssb.12356`.

Friedman J, Hastie T, Tibshirani R (2008). "Sparse Inverse Covariance Estimation with the Graphical Lasso." *Biostatistics*, **9**(3), 432–441. `doi:10.1093/biostatistics/kxm045`.

Friedman J, Hastie T, Tibshirani R (2019). ***glasso**: Graphical Lasso: Estimation of Gaussian Graphical Models*. R package version 1.11, URL `https://CRAN.R-project.org/package=glasso`.

Ghurye SG, Olkin I (1969). "Unbiased Estimation of Some Multivariate Probability Densities and Related Functions." *The Annals of Mathematical Statistics*, **40**(4), 1261–1271. `doi:10.1214/aoms/1177697501`.

Gutmann MU, Corander J (2016). "Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models." *Journal of Machine Learning Research*, **17**(1), 4256–4302.

Hartig F, Dislich C, Wiegand T, Huth A (2014). "Technical Note: Approximate Bayesian Parameterization of a Process-Based Tropical Forest Model." *Biogeosciences*, **11**, 1261–1272. `doi:10.5194/bg-11-1261-2014`.

Izenman AJ (1991). "Recent Developments in Nonparametric Density Estimation." *Journal of the American Statistical Association*, **86**(413), 205–224. `doi:10.1080/01621459.1991.10475021`.

Jabot F, Faure T, Dumoulin N (2013). "**EasyABC**: Performing Efficient Approximate Bayesian Computation Sampling Schemes Using R." *Methods in Ecology and Evolution*, **4**(7), 684–687. `doi:10.1111/2041-210x.12050`.

Kane MJ, Emerson J, Weston S (2013). "Scalable Strategies for Computing with Massive Data." *Journal of Statistical Software*, **55**(14), 1–19. `doi:10.18637/jss.v055.i14`.

Karabatsos G (2018). "On Bayesian Testing of Additive Conjoint Measurement Axioms Using Synthetic Likelihood." *Psychometrika*, **83**(2), 321–332. `doi:10.1007/s11336-017-9581-x`.

Kassambara A (2019). ***ggcorrplot**: Visualization of a Correlation Matrix Using **ggplot2***. R package version 0.1.3, URL `https://CRAN.R-project.org/package=ggcorrplot`.

Kessy A, Lewin A, Strimmer K (2018). "Optimal Whitening and Decorrelation." *The American Statistician*, **72**(4), 309–314. `doi:10.1080/00031305.2016.1277159`.

Leclercq F (2018). "Bayesian Optimization for Likelihood-Free Cosmological Inference." *Physical Review D*, **98**(6), 063511. `doi:10.1103/physrevd.98.063511`.

Lintusaari J, Vuollekoski H, Kangasrääsiö A, Skytén K, Järvenpää M, Marttinen P, Gutmann MU, Vehtari A, Corander J, Kaski S (2018). "ELFI: Engine for Likelihood-Free Inference." *Journal of Machine Learning Research*, **19**(16), 1–7.

Magnusson A, Stewart I (2020). ***plotMCMC**: MCMC Diagnostic Plots.* R package version 2.0-1, URL https://CRAN.R-project.org/package=plotMCMC.

Marchand P, Boenke M, Green DM (2017). "A Stochastic Movement Model Reproduces Patterns of Site Fidelity and Long-Distance Dispersal in a Population of Fowler's Toads (*Anaxyrus Fowleri*)." *Ecological Modelling*, **360**, 63–69. doi:10.1016/j.ecolmodel.2017.06.025.

Meeds E, Leenders R, Welling M (2015). "Hamiltonian ABC." In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pp. 582–591.

Microsoft Corporation, Weston S (2020a). ***doParallel**: Foreach Parallel Adaptor for the **parallel** Package.* R package version 1.0.16, URL https://CRAN.R-project.org/package=doParallel.

Microsoft Corporation, Weston S (2020b). ***foreach**: Provides Foreach Looping Construct for R.* R package version 1.5.1, URL https://CRAN.R-project.org/package=foreach.

Neal RM (2003). "Slice Sampling." *The Annals of Statistics*, **31**(3), 705–767. doi:10.1214/aos/1056562461.

Ong VMH, Nott DJ, Tran MN, Sisson SA, Drovandi CC (2018a). "Likelihood-Free Inference in High Dimensions with Synthetic Likelihood." *Computational Statistics & Data Analysis*, **128**, 271–291. doi:10.1016/j.csda.2018.07.008.

Ong VMH, Nott DJ, Tran MN, Sisson SA, Drovandi CC (2018b). "Variational Bayes with Synthetic Likelihood." *Statistics and Computing*, **28**(4), 971–988. doi:10.1007/s11222-017-9773-3.

Picchini U, Forman JL (2019). "Bayesian Inference for Stochastic Differential Equation Mixed Effects Models of a Tumour Xenography Study." *Journal of the Royal Statistical Society C*, **68**(4), 887–913. doi:10.1111/rssc.12347.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://CRAN.R-project.org/doc/Rnews/.

Prangle D, Fearnhead P, Cox MP, Biggs PJ, French NP (2014). "Semi-Automatic Selection of Summary Statistics for ABC Model Choice." *Statistical Applications in Genetics and Molecular Biology*, **13**(1), 67–82. doi:10.1515/sagmb-2013-0012.

Price LF, Drovandi CC, Lee A, Nott DJ (2018). "Bayesian Synthetic Likelihood." *Journal of Computational and Graphical Statistics*, **27**(1), 1–11. doi:10.1080/10618600.2017.1302882.

Priddle J, Sisson S, Drovandi C (in press). "Efficient Bayesian Synthetic Likelihood with Whitening Transformations." *Journal of Computational and Graphical Statistics*. doi:10.1080/10618600.2021.1979012.

Pudlo P, Marin JM, Estoup A, Cornuet JM, Gautier M, Robert CP (2016). "Reliable ABC Model Choice via Random Forests." *Bioinformatics*, **32**(6), 859–866. doi:10.1093/bioinformatics/btv684.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robert CP, Cornuet JM, Marin JM, Pillai NS (2011). "Lack of Confidence in Approximate Bayesian Computation Model Choice." *Proceedings of the National Academy of Sciences of the United States of America*, **108**(37), 15112–15117. doi:10.1073/pnas.1102900108.

Sisson SA, Fan Y, Beaumont M (2018). *Handbook of Approximate Bayesian Computation*. 1st edition. Chapman & Hall/CRC. doi:10.1201/9781315117195.

Strimmer K, Jendoubi T, Kessy A, Lewin A (2019). ***whitening**: Whitening and High-Dimensional Canonical Correlation Analysis*. R package version 1.1.1, URL https://CRAN.R-project.org/package=whitening.

Van Rossum G, *et al.* (2011). *Python Programming Language*. URL https://www.python.org/.

Warton DI (2008). "Penalized Normal Likelihood and Ridge Regularization of Correlation and Covariance Matrices." *Journal of the American Statistical Association*, **103**(481), 340–349. doi:10.1198/016214508000000021.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag, New York. doi:10.1007/978-0-387-98141-3.

Wood SN (2010). "Statistical Inference for Noisy Nonlinear Ecological Dynamic Systems." *Nature*, **466**(7310), 1102–1107. doi:10.1038/nature09319.

Affiliation:

Christopher Drovandi
School of Mathematical Sciences
Faculty of Science
Queensland University of Technology
Brisbane Queensland 4000, Australia
E-mail: c.drovandi@qut.edu.au
URL: https://chrisdrovandi.weebly.com/

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/

January 2022, Volume 101, Issue 11 *Submitted:* 2019-06-30
doi:10.18637/jss.v101.i11 *Accepted:* 2021-02-08