Adaptive Mixture of Student-t Distributions as a Flexible Candidate Distribution for Efficient Simulation: The R Package AdMit

David Ardia
University of Neuchâtel
Laval University

Lennart F. Hoogerheide
Vrije Universität Amsterdam

Herman K. van Dijk
Erasmus University

Abstract

This introduction to the R package **AdMit** is a shorter version of **?**, published in the *Journal of Statistical Software*. The package provides flexible functions to approximate a certain target distribution and to efficiently generate a sample of random draws from it, given only a kernel of the target density function. The core algorithm consists of the function `AdMit` which fits an adaptive mixture of Student-t distributions to the density of interest. Then, importance sampling or the independence chain Metropolis-Hastings algorithm is used to obtain quantities of interest for the target density, using the fitted mixture as the importance or candidate density. The estimation procedure is fully automatic and thus avoids the time-consuming and difficult task of tuning a sampling algorithm. The relevance of the package is shown in an example of a bivariate bimodal distribution.

Keywords: adaptive mixture, Student-t distributions, importance sampling, independence chain Metropolis-Hastings algorithm, Bayesian inference, R software.

concordance:AdMit.tex:AdMit.Rnw:1 103 1 1 6 481 1 1 10 12 0 1 2 4 1 1 5 4 0 2 1 1 3 2 0 2 1 3 0 1 2 4 1 1 12 1 2 8 1 1 2 1 0 2 1 37 0 1 2 34 1 1 5 4 0 1 1 1 3 2 0 2 1 3 0 1 2 2 1 1 11 1 2 8 1 1 2 1 0 1 14 16 0 1 2 3 1 1 16 1 2 27 1 1 2 1 0 2 1 13 0 1 2 29 1 1 12 14 0 2 2 1 0 2 1 12 0 2 1 8 0 1 2 6 1 1 2 9 0 1 2 16 1 1 2 1 0 2 1 31 0 1 2 17 1 1 5 1 0 2 1 8 0 1 2 4 1 1 2 8 0 1 2 49 1

1. Introduction

In scientific analysis one is usually interested in the effect of one variable, say, education ($= x$), on an other variable, say, earned income ($= y$). In the standard linear regression model this effect of x on y is assumed constant, i.e., $\mathsf{E}(y) = \beta x$, with β a constant. The uncertainty of many estimators of β is usually represented by a symmetric Student-t density (see, e.g., **?**, Chapter 3). However, in many realistic models the effect of x on y is a function of several deeper structural parameters. In such cases, the uncertainty of the estimates of β may be rather non-symmetric. More formally, in a Bayesian procedure, the target or posterior density may exhibit rather non-elliptical shapes (see, e.g., **??**). Hence, in several cases of scientific analysis, one deals with a target distribution that has very non-elliptical contours and that it is not a member of a known class of distributions. Therefore, there exists a need for flexible and efficient simulation methods to approximate such target distributions.

This article illustrates the adaptive mixture of Student-t distributions (AdMit) procedure

(see **???**, for details) and presents its R implementation (**?**) with the package **AdMit** (**?**). The AdMit procedure consists of the construction of a mixture of Student-t distributions which approximates a target distribution of interest. The fitting procedure relies only on a kernel of the target density, so that the normalizing constant is not required. In a second step this approximation is used as an importance function in importance sampling or as a candidate density in the independence chain Metropolis-Hastings (M-H) algorithm to estimate characteristics of the target density. The estimation procedure is fully automatic and thus avoids the difficult task, especially for non-experts, of tuning a sampling algorithm. The R package **AdMit** is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=AdMit`.

In a *standard* case of importance sampling or the independence chain M-H algorithm, the candidate density is unimodal. If the target distribution is multimodal then some draws may have huge weights in the importance sampling approach and a second mode may be completely missed in the M-H strategy. As a consequence, the convergence behavior of these Monte Carlo integration methods is rather uncertain. Thus, an important problem is the choice of the importance or candidate density, especially when little is known a priori about the shape of the target density. For both importance sampling and the independence chain M-H, it holds that the candidate density should be *close* to the target density, and it is especially important that the tails of the candidate should not be thinner than those of the target.

? and **?** mention several reasons why mixtures of Student-t distributions are natural candidate densities. First, they can provide an accurate approximation to a wide variety of target densities, with substantial skewness and high kurtosis. Furthermore, they can deal with multi-modality and with non-elliptical shapes due to asymptotes. Second, this approximation can be constructed in a quick, iterative procedure and a mixture of Student-t distributions is easy to sample from. Third, the Student-t distribution has fatter tails than the Normal distribution; especially if one specifies Student-t distributions with few degrees of freedom, the risk is small that the tails of the candidate are thinner than those of the target distribution. Finally, **?** showed that under certain conditions any density function may be approximated to arbitrary accuracy by a convex combination of *basis* densities; the mixture of Student-t distributions falls within their framework. One sufficient condition ensuring the feasibility of the approach is that the target density function is continuous on a compact domain. It is further allowed that the target density is not defined on a compact set, but with tails behaving like a Student-t distribution. Furthermore, it is even allowed that the target tends to infinity at a certain value as long as the function is square integrable. In practice, a non-expert user sometimes does not know whether the necessary conditions are satisfied. However, one can check the behaviour of the relative numerical efficiency as robustness check; if the necessary conditions are not satisfied, this will tend to zero as the number of draws increases (even if the number of components in the approximation becomes larger). Obviously, if the provided target density kernel does not correspond to a proper distribution, the approximation will not converge to a sensible result. These cases of improper distributions should be discovered before starting a Monte Carlo simulation.

The R package **AdMit** consists of three main functions: `AdMit`, `AdMitIS` and `AdMitMH`. The first one allows the user to fit a mixture of Student-t distributions to a given density through its kernel function. The next two functions perform importance sampling and independence chain M-H sampling using the fitted mixture estimated by `AdMit` as the importance or can-

didate density, respectively. To illustrate the use of the package, we first apply the AdMit methodology to a bivariate bimodal distribution. We describe in detail the use of the functions provided by the package and document the relevance of the methodology to reproduce the shape of non-elliptical distributions.

The outline of the paper is as follows: Section **??** presents the principles of the AdMit algorithm. Section **??** presents the functions provided by the package with an illustration of a bivariate non-elliptical distribution. Section **??** concludes.

2. Adaptive mixture of Student-t distributions

The adaptive mixture of Student-t distributions method developed in **?** and **?** constructs a mixture of Student-t distributions in order to approximate a given target density $p(\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$. The density of a mixture of Student-t distributions can be written as:

$$q(\boldsymbol{\theta}) = \sum_{h=1}^{H} \eta_h \, t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu) \;,$$

where η_h $(h = 1, \ldots, H)$ are the mixing probabilities of the Student-t components, $0 \leqslant \eta_h \leqslant 1$ $(h = 1, \ldots, H)$, $\sum_{h=1}^{H} \eta_h = 1$, and $t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu)$ is a d-dimensional Student-t density with mode vector $\boldsymbol{\mu}_h$, scale matrix $\boldsymbol{\Sigma}_h$, and ν degrees of freedom:

$$t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu) = \frac{\Gamma\left(\frac{\nu+d}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)(\pi\nu)^{d/2}} \times (\det \boldsymbol{\Sigma}_h)^{-1/2} \left(1 + \frac{(\boldsymbol{\theta} - \boldsymbol{\mu}_h)' \boldsymbol{\Sigma}_h^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_h)}{\nu}\right)^{-(\nu+d)/2} .$$

The adaptive mixture approach determines H, η_h, $\boldsymbol{\mu}_h$ and $\boldsymbol{\Sigma}_h$ $(h = 1, \ldots, H)$ based on a kernel function $k(\boldsymbol{\theta})$ of the target density $p(\boldsymbol{\theta})$. It consists of the following steps:

Step 0 – Initial step Compute the mode $\boldsymbol{\mu}_1$ and scale $\boldsymbol{\Sigma}_1$ of the first Student-t distribution in the mixture as $\boldsymbol{\mu}_1 = \arg\max_{\boldsymbol{\theta} \in \Theta} \log k(\boldsymbol{\theta})$, the mode of the log kernel function, and $\boldsymbol{\Sigma}_1$ as minus the Hessian of $\log k(\boldsymbol{\theta})$ evaluated at its mode $\boldsymbol{\mu}_1$. Then draw a set of N_s points $\boldsymbol{\theta}^{[i]}$ $(i = 1, \ldots, N_s)$ from this first stage candidate density $q(\boldsymbol{\theta}) = t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, \nu)$, with small ν to allow for fat tails.

Comment: In the rest of this paper, we use Student-t distributions with one degrees of freedom (i.e., $\nu = 1$) since:

1. it enables the method to deal with fat-tailed target distributions;
2. it makes it easier for the iterative procedure to detect modes that are far apart.

After that add components to the mixture, iteratively, by performing the following steps:

Step 1 – Evaluate the distribution of weights Compute the importance sampling weights $w(\boldsymbol{\theta}^{[i]}) = k(\boldsymbol{\theta}^{[i]})/q(\boldsymbol{\theta}^{[i]})$ for $i = 1, \ldots, N_s$. In order to determine the number of components H of the mixture we make use of a simple diagnostic criterion: *the coefficient of variation*, i.e., the standard deviation divided by the mean, of the importance sampling

weights $\{w(\boldsymbol{\theta}^{[i]}) \,|\, i = 1, \ldots, N_s\}$. If the relative change in the coefficient of variation of the importance sampling weights caused by adding one new Student-t component to the candidate mixture is small, e.g., less than 10%, then the algorithm stops and the current mixture $q(\boldsymbol{\theta})$ is the approximation. Otherwise, the algorithm goes to step 2.

Comment: Notice that $q(\boldsymbol{\theta})$ is a proper density, whereas $k(\boldsymbol{\theta})$ is a density kernel. So, the procedure does not provide an approximation to the kernel $k(\boldsymbol{\theta})$ but provides an approximation to the density of which $k(\boldsymbol{\theta})$ is a kernel.

Comment: There are several reasons for using the coefficient of variation of the importance sampling weights. First, it is a natural, intuitive measure of quality of the candidate as an approximation to the target. If the candidate and the target distributions coincide, all importance sampling weights are equal, so that the coefficient of variation is zero. For a poor candidate that not even roughly approximates the target, some importance sampling weights are huge while most are (almost) zero, so that the coefficient of variation is high. The better the candidate approximates the target, the more evenly the weight is divided among the candidate draws, and the smaller the coefficient of variation of the importance sampling weights. Second, **?** argues that a reasonable objective in the choice of an importance density is the minimization of:

$$\mathsf{E}_p[w(\boldsymbol{\theta})] = \int \frac{k(\boldsymbol{\theta})^2}{q(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} = \int \left[\frac{k(\boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right]^2 q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = \mathsf{E}_q[w(\boldsymbol{\theta})^2] \,,$$

or equivalently, the minimization of the coefficient of variation:

$$\frac{\left(\mathsf{E}_q[w(\boldsymbol{\theta})^2] - \mathsf{E}_q[w(\boldsymbol{\theta})]^2\right)^{1/2}}{\mathsf{E}_q[w(\boldsymbol{\theta})]} \,,$$

since:

$$\mathsf{E}_q[w(\boldsymbol{\theta})] = \int \frac{k(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = \int k(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}$$

does not depend on $q(\boldsymbol{\theta})$.

The reason for quoting the coefficient of variation rather than the standard deviation is that the standard deviation of the *scaled* weights (i.e., adding up to one) depends on the number of draws, whereas the standard deviation of the *unscaled* weights depends on the scaling constant $\int k(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}$ (i.e., typically the marginal likelihood). The coefficient of variation of the importance sampling weights, which is equal for scaled and unscaled weights, reflects the quality of the candidate as an approximation to the target (not depending on number of draws or $\int k(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}$). The coefficient of variation is the function one would minimize if one desires to estimate $\mathsf{P}(\boldsymbol{\theta} \in D)$, where $D \subset \Theta$, if the true value is $\mathsf{P}(\boldsymbol{\theta} \in D) = 0.5$. Different functions should be minimized for different quantities of interest. However, it is usually impractical to perform a separate tuning algorithm for the importance density for each quantity of interest. Fortunately, in practice the candidate resulting from the minimization of the coefficient of variation performs well for estimating common quantities of interest such as posterior moments. **?** propose a different approach for forecasting extreme quantiles where one substantially improves on the usual strategy by generating relatively far too many extreme candidate draws.

Step 2a – Iterate on the number of components Add another Student-t distribution with density $t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu)$ to the mixture with $\boldsymbol{\mu}_h = \arg\max_{\boldsymbol{\theta} \in \Theta} \log w(\boldsymbol{\theta})$ and $\boldsymbol{\Sigma}_h$

equal to minus the inverse Hessian of $\log w(\boldsymbol{\theta})$. Here, $q(\boldsymbol{\theta})$ denotes the density of the mixture of $(h-1)$ Student-t distributions obtained in the previous iteration of the procedure. An obvious initial value for the maximization procedure for computing $\boldsymbol{\mu}_h$ is the point $\boldsymbol{\theta}^{[i]}$ with the highest weight in the sample $\{w(\boldsymbol{\theta}^{[i]}) \,|\, i = 1, \ldots, N_s\}$. The idea behind this choice is that the new Student-t component should *cover* a region where the weights $w(\boldsymbol{\theta})$ are relatively large. The point where the weight function $w(\boldsymbol{\theta})$ attains its maximum is an obvious choice for $\boldsymbol{\mu}_h$, while the scale matrix $\boldsymbol{\Sigma}_h$ is the covariance matrix of the local Normal approximation to the distribution with density kernel $w(\boldsymbol{\theta})$ around the point $\boldsymbol{\mu}_h$.

Comment: There are several reasons for the use of minus the inverse Hessian of $\log w(\boldsymbol{\theta})$ as the scale matrix for the new component. First, suppose that $k(\boldsymbol{\theta})$ is a posterior kernel under a flat prior, and that the first candidate distribution would be a uniform distribution (or a Student-t with a huge scale matrix). Then $\log w(\boldsymbol{\theta})$ takes its maximum likelihood estimator and minus its inverse Hessian is an asymptotically valid estimate for the maximum likelihood estimator's covariance matrix. Second, since $\log w(\boldsymbol{\theta})$ takes its maximum at $\boldsymbol{\mu}_h$, its Hessian is negative definite (unless it is located at a boundary, in which case we do not use this scale matrix). Therefore, minus the inverse Hessian is a positive definite matrix that can be used as a covariance or scale matrix. Moreover, we want to add candidate probability mass to those areas of the parameter space where $w(\boldsymbol{\theta})$ is relatively high, i.e., where there is relatively little candidate probability mass. This is the reason for choosing the mode $\boldsymbol{\mu}_h$ of the new candidate component at the maximum of $w(\boldsymbol{\theta})$. Especially in those directions where $w(\boldsymbol{\theta})$ decreases slowy (i.e., moving away from $\boldsymbol{\mu}_h$) we want to add candidate probability mass also further away from $\boldsymbol{\mu}_h$. This is reflected by larger elements of minus the inverse Hessian of $\log w(\boldsymbol{\theta})$ at $\boldsymbol{\mu}_h$. Note that $w(\boldsymbol{\theta})$ is generally not a kernel of a proper density on Θ. However, we also do not require this. We only make use of its local behaviour around its maximum at $\boldsymbol{\mu}_h$, reflected by minus the inverse Hessian of $\log w(\boldsymbol{\theta})$. That is, we specify a Student-t distribution that locally behaves the same as the ratio $w(\boldsymbol{\theta})$.

Comment: To improve the algorithm's ability to detect distant modes of a multimodal target density we consider one additional initial value for the optimization and we use the point corresponding to the highest value of the weight function among the two optima as the mode $\boldsymbol{\mu}_h$ of the new component in the candidate mixture.

Step 2b – Optimize the mixing probabilities Choose the probabilities η_h $(h = 1, \ldots, H)$ in the mixture $q(\boldsymbol{\theta}) = \sum_{h=1}^{H} \eta_h \, t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu)$ by minimizing the (squared) coefficient of variation of the importance sampling weights. First, draw N_p points $\boldsymbol{\theta}_h^{[i]}$ $(i = 1, \ldots, N_p)$ from each component $t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu)$ $(h = 1 \ldots, H)$. Then, minimize:

$$\mathsf{E}[w(\boldsymbol{\theta})^2]/\mathsf{E}[w(\boldsymbol{\theta})]^2 \tag{1}$$

with respect to η_h $(h = 1, \ldots, H)$, where:

$$\mathsf{E}[w(\boldsymbol{\theta})^m] = \frac{1}{N_p} \sum_{i=1}^{N_p} \sum_{h=1}^{H} \eta_h \, w(\boldsymbol{\theta}_h^{[i]})^m \quad (m = 1, 2) \,,$$

and:

$$w(\boldsymbol{\theta}_h^{[i]}) = \frac{k(\boldsymbol{\theta}_h^{[i]})}{\sum_{l=1}^{H} \eta_l \, t_d(\boldsymbol{\theta}_h^{[i]} \,|\, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l, \nu)} \,.$$

Comment: Minimization of (**??**) is time consuming. The reason is that this concerns the optimization of a non-linear function of η_h ($h = 1, \ldots, H$) where H takes the values $2, 3, \ldots$ in the consecutive iterations of the algorithm. Evaluating the function itself requires already NH evaluations of the kernel and NH^2 evaluations of the Student-t densities. The computation of (analytically evaluated) derivatives of the function with respect to η_h ($h = 1, \ldots, H$) takes even more time. One way to reduce the amount of computing time required for the construction of the approximation is to use different numbers of draws in different steps. One can use a relatively small sample of N_p draws for the optimization of the mixing probabilities and a large sample of N_s draws in order to evaluate the quality of the current candidate mixture at each iteration (in the sense of the coefficient of variation of the corresponding importance sampling weights) and in order to obtain an initial value for the algorithm that is used to optimize the weight function (that yields the mode of a new Student-t component in the mixture). Note that it is not necessary to find the *globally optimal* values of the mixing probabilities; a *good* approximation to the target density is all that is required.

Step 2c – Draw from the mixture Draw a sample of N_s points $\boldsymbol{\theta}^{[i]}$ ($i = 1, \ldots, N_s$) from the new mixture of Student-t distributions, $q(\boldsymbol{\theta}) = \sum_{h=1}^{H} \eta_h \, t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu)$, and go to step 1; in order to draw a point from the density $q(\boldsymbol{\theta})$ first use a draw from the uniform distribution $\mathcal{U}(0, 1)$ to determine which component $t_d(\boldsymbol{\theta} \,|\, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h, \nu)$ is chosen, and then draw from this d-dimensional Student-t distribution.

Comment: It may occur that one is dissatisfied with diagnostics like the coefficient of variation of the importance sampling weights corresponding to the final candidate density resulting from the procedure above. In that case the user may start all over again the procedure with a larger number of points N_s. The idea behind this strategy is that the larger N_s, the easier it is for the method to detect and approximate the shape of the target density kernel, and to specify the Student-t distributions of the mixture adequately.

If the region of integration $\Theta \subseteq \mathbb{R}^d$ is bounded, it may occur in step 2 that $w(\boldsymbol{\theta})$ attains its maximum at a boundary of the integration region. In this case minus the inverse Hessian of $\log w(\boldsymbol{\theta})$ evaluated at its mode $\boldsymbol{\mu}_h$ may be a very poor scale matrix; in fact this matrix may not even be positive definite. In such situations, $\boldsymbol{\mu}_h$ and $\boldsymbol{\Sigma}_h$ are obtained as the estimated mean and covariance based on a subset of draws corresponding to a certain percentage of largest weights. More precisely, $\boldsymbol{\mu}_h$ and $\boldsymbol{\Sigma}_h$ are obtained using the sample $\{\boldsymbol{\theta}^{[i]} \,|\, i = 1, \ldots, N_s\}$ from $q(\boldsymbol{\theta})$ we already have:

$$\begin{aligned} \boldsymbol{\mu}_h &= \sum_{j \in J_c} \frac{w(\boldsymbol{\theta}^{[j]})}{\sum_{j \in J_c} w(\boldsymbol{\theta}^{[j]})} \boldsymbol{\theta}^{[j]} \\ \boldsymbol{\Sigma}_h &= \sum_{j \in J_c} \frac{w(\boldsymbol{\theta}^{[j]})}{\sum_{j \in J_c} w(\boldsymbol{\theta}^{[j]})} (\boldsymbol{\theta}^{[j]} - \boldsymbol{\mu}_h)(\boldsymbol{\theta}^{[j]} - \boldsymbol{\mu}_h)' \,, \end{aligned} \tag{2}$$

where J_c denotes the set of indices corresponding to the c percents of the largest weights in the sample $\{w(\boldsymbol{\theta}^{[i]}) \,|\, i = 1, \ldots, N_s\}$. Since our aim is to detect regions with too little candidate probability mass (e.g., a distant mode), the percentage c is typically a low value, i.e., 5%, 15% or 30%. Moreover, the estimated $\boldsymbol{\Sigma}_h$ can be scaled by a given factor for robustness. Different percentages and scaling factors could be used together, leading to different coefficients of

variation at each step of the adaptive procedure. The matrix leading to the smallest coefficient of variation could then be selected as the scale matrix $\boldsymbol{\Sigma}_h$ for the new mixture component.

Once the adaptive mixture of Student-t distributions has been fitted to the target density $p(\boldsymbol{\theta})$ through the kernel function $k(\boldsymbol{\theta})$, the approximation $q(\boldsymbol{\theta})$ is used in importance sampling or in the independence chain Metropolis-Hastings (M-H) algorithm to obtain quantities of interest for the target density $p(\boldsymbol{\theta})$ itself.

2.1. Background on importance sampling

Importance sampling, due to **?**, was introduced in econometrics and statistics by **?**. It is based on the following relationship:

$$\mathsf{E}_p\big[g(\boldsymbol{\theta})\big] = \frac{\int g(\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}{\int p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}} = \frac{\int g(\boldsymbol{\theta})w(\boldsymbol{\theta})q(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}}{\int w(\boldsymbol{\theta})q(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}} = \frac{\mathsf{E}_q\big[g(\boldsymbol{\theta})w(\boldsymbol{\theta})\big]}{\mathsf{E}_q\big[w(\boldsymbol{\theta})\big]} \,, \tag{3}$$

where $g(\boldsymbol{\theta})$ is a given (integrable with respect to p) function, $w(\boldsymbol{\theta}) = k(\boldsymbol{\theta})/q(\boldsymbol{\theta})$, E_p denotes the expectation with respect to the target density $p(\boldsymbol{\theta})$ and E_q denotes the expectation with respect to the (importance) approximation $q(\boldsymbol{\theta})$. The importance sampling estimator of $\mathsf{E}_p\big[g(\boldsymbol{\theta})\big]$ is then obtained as the sample counter-part of the right-hand side of (**??**):

$$\widehat{g} = \frac{\sum_{i=1}^{N} g(\boldsymbol{\theta}^{[i]})w(\boldsymbol{\theta}^{[i]})}{\sum_{i=1}^{N} w(\boldsymbol{\theta}^{[i]})} \,, \tag{4}$$

where $\{\boldsymbol{\theta}^{[i]} \,|\, 1, \ldots, N\}$ is a sample of draws from the importance density $q(\boldsymbol{\theta})$. Under certain conditions (see **?**), $\widehat{g}$ is a consistent estimator of $\mathsf{E}_p\big[g(\boldsymbol{\theta})\big]$. The choice of the function $g(\boldsymbol{\theta})$ allows to obtain different quantities of interest for $p(\boldsymbol{\theta})$. For instance, the mean estimate of $p(\boldsymbol{\theta})$, denoted by $\overline{\boldsymbol{\theta}}$, is obtained with $g(\boldsymbol{\theta}) = \boldsymbol{\theta}$; the covariance matrix estimate is obtained using $g(\boldsymbol{\theta}) = (\boldsymbol{\theta} - \overline{\boldsymbol{\theta}})(\boldsymbol{\theta} - \overline{\boldsymbol{\theta}})'$; the estimated probability that $\boldsymbol{\theta}$ belongs to a domain $D \subseteq \Theta$ using $g(\boldsymbol{\theta}) = \mathbb{I}_{\{\boldsymbol{\theta} \in D\}}$, where $\mathbb{I}_{\{\bullet\}}$ denotes the indicator function which is equal to one if the constraint holds and zero otherwise.

2.2. Background on the independence chain Metropolis-Hastings algorithm

The Metropolis-Hastings (M-H) algorithm is a Markov chain Monte Carlo (MCMC) approach that has been introduced by **?** and generalized by **?**. MCMC methods construct a Markov chain converging to a target distribution $p(\boldsymbol{\theta})$. After a *burn-in* period, which is required to make the influence of initial values negligible, draws from the Markov chain are considered as (correlated) draws from the target distribution itself.

In the independence chain M-H algorithm, a Markov chain of length N is constructed by the following procedure. First, one chooses a feasible initial state $\boldsymbol{\theta}^{[0]}$. Then, one repeats the following steps N times (for $i = 1, \ldots, N$). A candidate value $\boldsymbol{\theta}^{\star}$ is drawn from the candidate density $q(\boldsymbol{\theta}^{\star})$ and a random variable U is drawn from the uniform distribution $\mathcal{U}(0,1)$. Then the acceptance probability:

$$\xi(\boldsymbol{\theta}^{[i-1]}, \boldsymbol{\theta}^{\star}) = \min\left\{\frac{w(\boldsymbol{\theta}^{\star})}{w(\boldsymbol{\theta}^{[i-1]})}, 1\right\}$$

is computed, where $w(\boldsymbol{\theta}) = k(\boldsymbol{\theta})/q(\boldsymbol{\theta})$, $k(\boldsymbol{\theta})$ being a kernel of the target density $p(\boldsymbol{\theta})$. If $U < \xi(\boldsymbol{\theta}^{[i-1]}, \boldsymbol{\theta}^{\star})$, the transition to the candidate value is accepted, i.e., $\boldsymbol{\theta}^{[i]} = \boldsymbol{\theta}^{\star}$. Otherwise the transition is rejected, and the next state is again $\boldsymbol{\theta}^{[i]} = \boldsymbol{\theta}^{[i-1]}$.

3. Illustration I: the Gelman-Meng distribution

This section presents the functions provided by the R package **AdMit** with an illustration of a bivariate bimodal distribution. This distribution belongs to the class of conditionally Normal distributions proposed by **?** with the property that the joint density is not Normal. In the notation of the previous section, we have $\boldsymbol{\theta} = (X_1 \; X_2)'$.

Let X_1 and X_2 be two random variables, for which X_1 is Normally distributed given X_2 and vice versa. Then, the joint distribution, after location and scale transformations in each variable, can be written as (see **?**):

$$p(x_1, x_2) \propto \exp\left(-\tfrac{1}{2}[Ax_1^2x_2^2 + x_1^2 + x_2^2 - 2Bx_1x_2 - 2C_1x_1 - 2C_2x_2]\right) \;, \tag{5}$$

where A, B, C_1 and C_2 are constants. Equation (**??**) can be rewritten as:

$$p(x_1, x_2) \propto \exp\left(-\tfrac{1}{2}\left[Ax_1^2x_2^2 + (x - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(x - \boldsymbol{\mu})\right]\right) \;,$$

with:

$$\boldsymbol{\mu} = \begin{pmatrix} \dfrac{BC_2 + C_1}{1 - B^2} & \dfrac{BC_1 + C_2}{1 - B^2} \end{pmatrix}' \quad \text{and} \quad \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} 1 & -B \\ -B & 1 \end{pmatrix} \;,$$

so the term $Ax_1^2x_2^2$ causes deviations from the bivariate Normal distribution. In what follows, we consider the symmetric case in which $A = 1$, $B = 0$, $C_1 = C_2 = 3$.

The core function provided by the R package **AdMit** is the function `AdMit`. The arguments of the function are the following:

```
AdMit(KERNEL, mu0, Sigma0 = NULL, control = list(), ...)
```

`KERNEL` is a kernel function $k(\boldsymbol{\theta})$ of the target density $p(\boldsymbol{\theta})$ on which the approximation is constructed. This function must contain the logical argument `log`. When `log = TRUE`, the function `KERNEL` returns the (natural) logarithm value of the kernel function; this is used for numerical stability. `mu0` is the starting value of the first stage optimization $\boldsymbol{\mu}_1 = \arg\max_{\boldsymbol{\theta}\in\Theta} \log k(\boldsymbol{\theta})$; it is a vector whose length corresponds to the length of the first argument in `KERNEL`. If one experiences misconvergence of the first stage optimization, one could first use an alternative (robust) optimization algorithm and use its output for `mu0`. For instance, the `DEoptim` function provided by the R package **DEoptim** (**?**) performs the optimization (minimization) of a function using an evolutionary (genetic) approach. `Sigma0` is the (symmetric positive definite) scale matrix of the first component. If a matrix is provided by the user, then it is used as the scale matrix of the first component and `mu0` is used as the mode of the first component. `control` is a list of tuning parameters. The most important parameters are: `Ns` (default: `1e+05`), the number of draws used for evaluating the importance sampling weights; `Np` (default: `1e+03`), the number of draws used for optimizing the mixing probabilities; `CVtol` (default: `0.1`), the tolerance of the relative change of the coefficient of variation; `df` (default: `1`), the degrees of freedom of the mixture components; `Hmax` (default: `10`), the maximum number of components of the mixture; `IS` (default: `FALSE`), indicates if the scale matrices $\boldsymbol{\Sigma}_h$

should always be estimated by importance sampling as in (??) without first trying to compute minus the inverse Hessian; `ISpercent` (default: `c(0.05, 0.15, 0.30)`), a vector of percentages of largest weights used in the importance sampling approach; `ISscale` (default: `c(1, 0.25, 4)`), a vector of scaling factors used to rescale the scale matrix obtained by importance sampling. Hence, when the argument `IS = TRUE`, nine scale matrices are constructed by default and the matrix leading to the smallest coefficient of variation is selected by the adaptive mixture procedure as $\boldsymbol{\Sigma}_h$. For details on the other `control` parameters, the reader is referred to the documentation file of `AdMit` (by typing `?AdMit`). Finally, the last argument of `AdMit` is `...` which allows the user to pass additional arguments to the function `KERNEL`. In econometric models for instance, the kernel may depend on a vector of observations $\mathbf{y} = (y_1 \cdots y_T)'$ which can be passed to the function `KERNEL` via this argument.

For the numerical optimization of the mode $\boldsymbol{\mu}_h$ and the estimation of the scale matrix $\boldsymbol{\Sigma}_h$ (i.e., when the `control` parameter `IS = FALSE`), the function `optim` is used with the option `BFGS` (the function `nlminb` cannot be used since it does not estimate the Hessian matrix at optimum). If the optimization procedure does not converge, the algorithm automatically switches to the `Nelder-Mead` approach which is more robust but slower. If still misconvergence occurs or if the Hessian matrix at optimum is not symmetric positive definite, the algorithm automatically switches to the importance sampling approach for this component.

For the numerical optimization of the mixing probabilities η_h ($h = 1, \ldots, H$), we rely on the function `nlminb` (for speed purposes) and apply the optimization on a reparametrized domain. More precisely, we optimize $(H-1)$ components in $\mathbb{R}^{(H-1)}$ instead of H components in $[0,1]^H$ with the summability constraint $\sum_{h=1}^{H} \eta_h$. If the optimization process does not converge, then the algorithm uses the function `optim` with method `Nelder-Mead` (or method `BFGS` for univariate optimization) which is more robust but slower. If still misconvergence occurs, the starting value is kept as the output of the procedure. The starting value corresponds to a mixing probability `weightNC` for η_h while the probabilities $\eta_1, \ldots, \eta_{H-1}$ are the probabilities of the previous mixture scaled by (`1 - weightNC`). The `control` parameter `weightNC` is set to `0.1` by default, i.e., a 10% probability is assigned to the new mixture component as a starting value. Finally, note that `AdMit` uses `C` and analytically evaluated derivatives to speed up the numerical optimization.

Let us come back to our bivariate conditionally Normal distribution. First, we need to define the kernel function in (??). This is achieved as follows:

```
R> GelmanMeng <- function(x, A = 1, B = 0, C1 = 3, C2 = 3, log = TRUE)
+   {
+     if (is.vector(x))
+       x <- matrix(x, nrow = 1)
+     r <- -0.5 * (A * x[,1]^2 * x[,2]^2 + x[,1]^2 + x[,2]^2 - 2 * B * x[,1] * x[,2] - 2 *
+     if (!log)
+       r <- exp(r)
+     as.vector(r)
+   }
```

Note that the argument `log` is set to `TRUE` by default so that the function outputs the (natural) logarithm of the kernel function. Moreover, the function is vectorized to speed up the computations. The argument `x` is therefore a matrix and the function outputs a vector. We strongly

advise the user to implement the kernel function in this fashion. A plot of `GelmanMeng` may be obtained as follows:

```
R> PlotGelmanMeng <- function(x1, x2)
+   {
+     GelmanMeng(cbind(x1, x2), log = FALSE)
+   }
R> x1 <- x2 <- seq(from = -1.0, to = 6.0, by = 0.05)
R> z <- outer(x1, x2, FUN = PlotGelmanMeng)
R> image(x1, x2, z, las = 1, col = gray((20:0)/20),
+         cex.axis = 1.1, cex.lab = 1.2,
+         xlab = expression(X[1]), ylab = expression(X[2]))
R> box()
R> abline(a = 0, b = 1, lty = "dotted")
```

The plot of `GelmanMeng` is displayed in Figure **??**. We notice the bimodal banana shape of the kernel function.

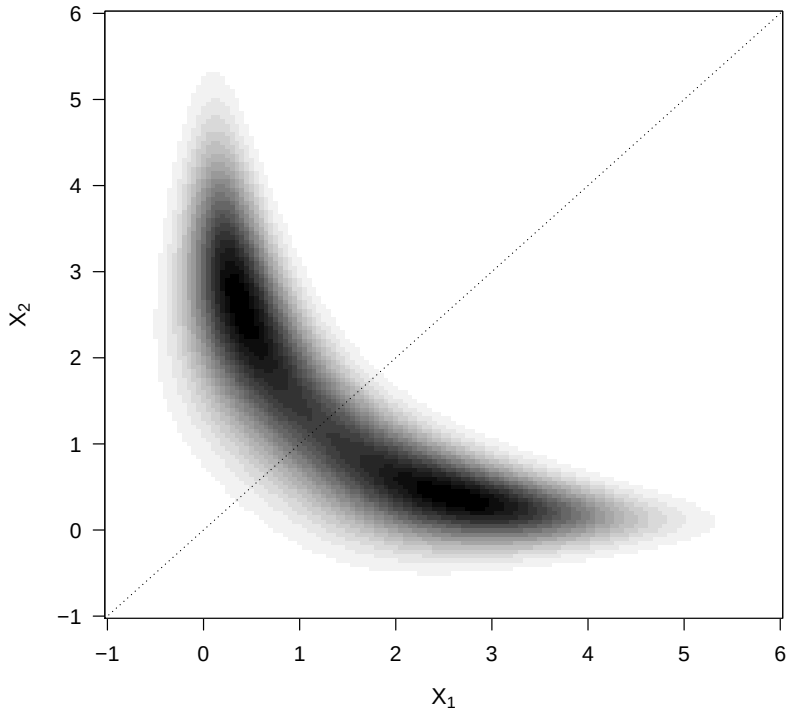

Figure 1: Plot of the **?** kernel function.

Let us now use the function `AdMit` to find a suitable approximation for the density function $p(\boldsymbol{\theta})$ whose kernel is (**??**). We set the seed of the pseudo-random number generator to a given

number and use the starting value `mu0 = c(0.0, 0.1)` for the first stage optimization. The result of the function is assigned to the object `outAdMit` and printed out:

```
R> set.seed(1234)
R> outAdMit <- AdMit(KERNEL = GelmanMeng, mu0 = c(0.0, 0.1))
R> print(outAdMit)

$CV
[1] 4.9544 1.3404 0.8904 0.8366

$mit
$mit$p
  cmp1   cmp2   cmp3   cmp4
0.4462 0.1418 0.2605 0.1515

$mit$mu
        k1      k2
cmp1 0.382 2.61803
cmp2 3.828 0.20337
cmp3 1.754 1.09411
cmp4 2.557 0.07151

$mit$Sigma
       k1k1     k1k2     k2k1    k2k2
cmp1 0.2292 -0.40000 -0.40000 1.57082
cmp2 0.8477 -0.08619 -0.08619 0.07277
cmp3 0.2823 -0.10573 -0.10573 0.23153
cmp4 0.7283 -0.19394 -0.19394 0.24948

$mit$df
[1] 1


$summary
  H METHOD.mu TIME.mu METHOD.p TIME.p     CV
1 1      BFGS    0.00     NONE   0.00 4.9544
2 2      BFGS    0.03   NLMINB   0.02 1.3404
3 3      BFGS    0.03   NLMINB   0.03 0.8904
4 4      BFGS    0.05   NLMINB   0.09 0.8366
```

The output of the function `AdMit` is a list. The first component is `CV`, a vector of length H which gives the value of the coefficient of variation at each step of the adaptive fitting procedure. The second component is `mit`, a list which consists of four components giving information on the fitted mixture of Student-t distributions: `p` is a vector of length H of mixing probabilities, `mu` is a $H \times d$ matrix whose rows give the modes of the mixture components, `Sigma` is a $H \times d^2$ matrix whose rows give the scale matrices (in vector form) of the mixture components and `df` is the degrees of freedom of the Student-t components. The third component of the list returned by `AdMit` is `summary`. This is a data frame containing information

on the adaptive fitting procedure: `H` is the component's number; `METHOD.mu` indicates which algorithm is used to estimate the mode and the scale matrix of the component (i.e., `USER`, `BFGS`, `Nelder-Mead` or `IS`); `TIME.mu` gives the computing time required for this optimization; `METHOD.p` gives the method used to optimize the mixing probabilities (i.e., `NONE`, `NLMINB`, `BFGS` or `Nelder-Mead`); `TIME.p` gives the computing time required for this optimization; `CV` gives the coefficient of variation of the importance sampling weights. When importance sampling is used (i.e., `IS = TRUE`), `METHOD.mu` is of the type `IS 0.05-0.25` indicating in this particular case, that importance sampling is used with the 5% largest weights and with a scaling factor of 0.25. Hence, if the `control` parameters `ISpercent` and `ISscale` are vectors of sizes d_1 and d_2, then $d_1 d_2$ matrices are considered for each component `H`, and the matrix leading to the smallest coefficient of variation is kept as the scale matrix $\mathbf{\Sigma}_h$ for this component. Time outputs `TIME.mu` and `TIME.p` are provided since it might be useful, as a robustness check, to see the computing time required for separate ingredients of the fitting procedure, that is the optimization of the modes and the optimization of the mixing probabilities. A very long computing time might indicate a numerical failure at some stage of the optimization process.

For the kernel function `GelmanMeng`, the approximation constructs a mixture of four components. The computing time required for the construction of the approximation is 4.4 seconds. The value of the coefficient of variation decreases from 4.8224 to 0.8315. A plot of the four-component approximation is displayed in Figure **??**. This graph is produced using the function `dMit` which returns the density of the mixture given by the output `outAdMit$mit`:

```
R> PlotMit <- function(x1, x2, mit)
+   {
+     dMit(cbind(x1, x2), mit = mit, log = FALSE)
+   }
R> z <- outer(x1, x2, FUN = PlotMit, mit = outAdMit$mit)
R> image(x1, x2, z, las = 1, col = gray((20:0)/20),
+         cex.axis = 1.1, cex.lab = 1.2,
+         xlab = expression(X[1]), ylab = expression(X[2]))
R> box()
R> abline(a = 0, b = 1, lty = "dotted")
```

The plot suggests that the four-component mixture provides a good approximation of the density function whose kernel is (**??**).

We can also use the mixture information `outAdMit$mit` to display each of the mixture components separately:

```
R> par(mfrow = c(2,2))
R> for (h in 1:4)
+   {
+     mith <- list(p = 1,
+                  mu = outAdMit$mit$mu[h,,drop = FALSE],
+                  Sigma = outAdMit$mit$Sigma[h,,drop = FALSE],
+                  df = outAdMit$mit$df)
+     z <- outer(x1, x2, FUN = PlotMit, mit = mith)
+     image(x1, x2, z, las = 1, col = gray((20:0)/20),
+           cex.axis = 1.1, cex.lab = 1.2,
```

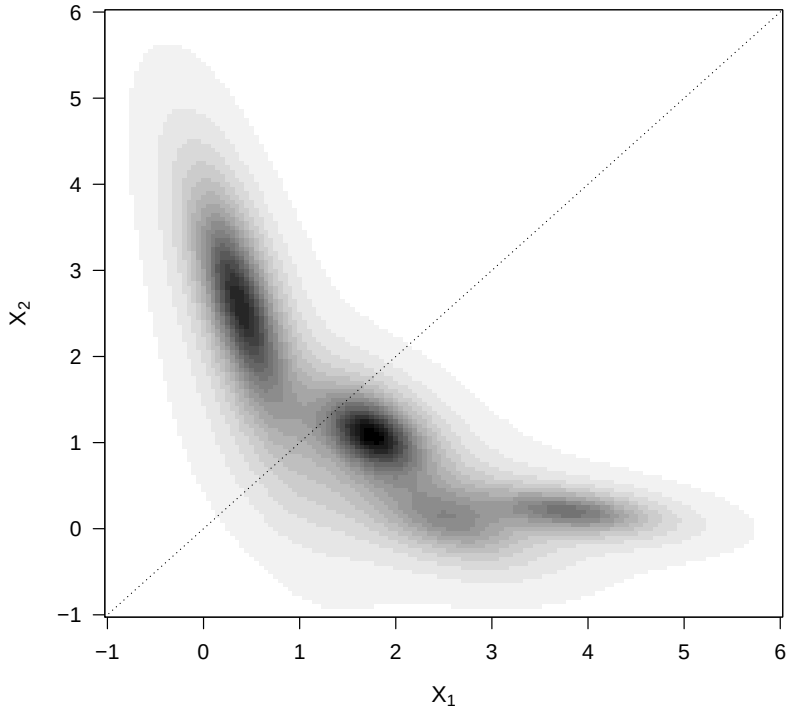

Figure 2: Plot of the four-component Student-t mixture approximation estimated by the function `AdMit`.

```
+           xlab = expression(X[1]), ylab = expression(X[2]))
+     box()
+     abline(a = 0, b = 1, lty = "dotted")
+     title(main = paste("component nr.", h))
+   }
```

Plots of the four components are displayed in Figure **??**.

Once the adaptive mixture of Student-t distributions is fitted to the density $p(\boldsymbol{\theta})$ using a kernel $k(\boldsymbol{\theta})$, the approximation $q(\boldsymbol{\theta})$ provided by `AdMit` is used as the importance sampling density in importance sampling or as the candidate density in the independence chain M-H algorithm.

The first function provided by the R package **AdMit** which allows to find quantities of interest for the density $p(\boldsymbol{\theta})$ using the output `outAdMit$mit` of `AdMit` is the function `AdMitIS`. This function performs importance sampling using the mixture approximation as the importance density (see Section **??**). The arguments of the function `AdMitIS` are the following:

```
AdMitIS(N = 1e+05, KERNEL, G = function(theta){theta}, mit = list(), ...)
```

`N` is the number of draws used in importance sampling; `KERNEL` is a kernel function $k(\boldsymbol{\theta})$ of the target density $p(\boldsymbol{\theta})$; `G` is the function $g(\boldsymbol{\theta})$ in (**??**); `mit` is a list providing information

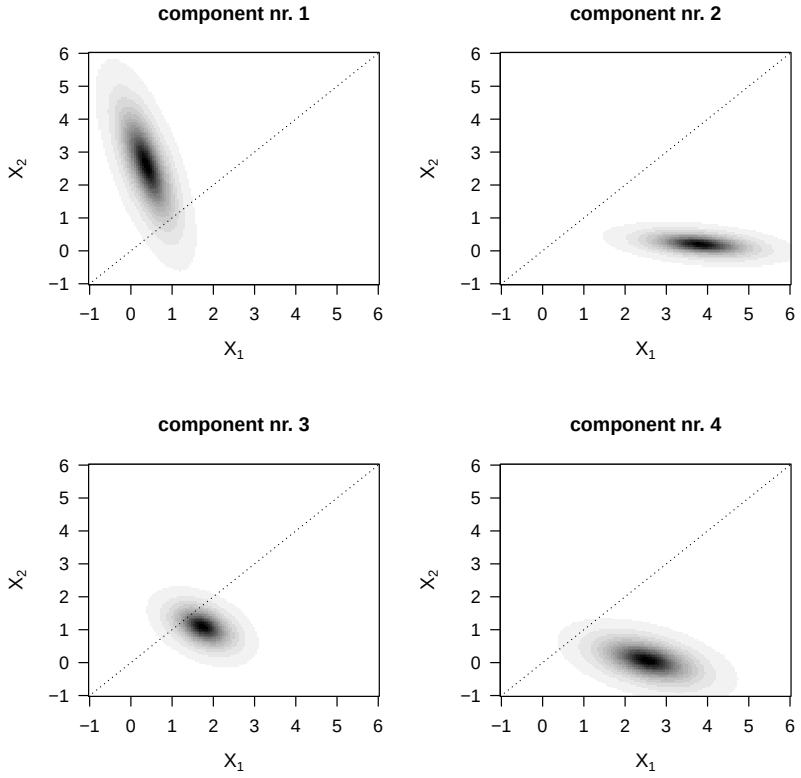

Figure 3: Student-t components of the four-component mixture approximation estimated by the function `AdMit`.

on the mixture approximation (i.e., typically the component `mit` in the output of the `AdMit` function); `...` allows additional parameters to be passed to the function `KERNEL` and/or `G`.

Let us apply the function `AdMitIS` to the kernel `GelmanMeng` using the approximation `outAdMit$mit`:

```
R> set.seed(1234)
R> outAdMitIS <- AdMitIS(KERNEL = GelmanMeng, mit = outAdMit$mit)
R> print(outAdMitIS)

$ghat
[1] 1.457 1.461

$NSE
[1] 0.004846 0.004916

$RNE
[1] 0.6450 0.6281
```

The output of the function `AdMitIS` is a list. The first component is `ghat`, the importance sampling estimator of $\mathsf{E}_p\big[g(\boldsymbol{\theta})\big]$ in (**??**). This is a vector whose length corresponds to the

length of the output of the function `G`. The second component is `NSE`, a vector containing the numerical standard errors (i.e., the square root of the variance of the estimates that can be expected if the simulations were to be repeated) of the components of `ghat`. The third component is `RNE`, a vector containing the relative numerical efficiencies of the components of `ghat` (i.e., the ratio between an estimate of the variance of an estimator based on direct sampling and the importance sampling estimator's estimated variance with the same number of draws). `RNE` is an indicator of the efficiency of the chosen importance function; if target and importance densities coincide, `RNE` equals one, whereas a very poor importance density will have a `RNE` close to zero. Both `NSE` and `RNE` are estimated by the method given in `?`. For estimating $\mathsf{E}_p[g(\boldsymbol{\theta})]$ the N candidate draws are approximately as 'valuable' as `RNE` $\times N$ independent draws from the target would be.

The computing time required to perform importance sampling on `GelmanMeng` using the four-component mixture `outAdMit$mit` is 0.7 seconds, where most part of the computing time is required for the N evaluations of the function `KERNEL` at the sampled values $\{\boldsymbol{\theta}^{[i]} \,|\, i = 1, \ldots, N\}$. The true values for $\mathsf{E}_p(X_1)$ and $\mathsf{E}_p(X_2)$ are 1.459. We notice that the importance sampling estimates are close to the true values and we note the good efficiency of the estimation.

By default, the function `G` is `function(theta){theta}` so that the function outputs a vector containing the mean estimates for the components of $\boldsymbol{\theta}$. Alternative functions may be provided by the user to obtain other quantities of interest for $p(\boldsymbol{\theta})$. The only requirement is that the function outputs a matrix. For instance, to estimate the covariance matrix of $\boldsymbol{\theta}$, we could define the following function:

```
R> G.cov <- function(theta, mu)
+   {
+     G.cov_sub <- function(x)
+       (x - mu) %*% t(x - mu)
+     theta <- as.matrix(theta)
+     tmp <- apply(theta, 1, G.cov_sub)
+     if (length(mu) > 1)
+       t(tmp)
+     else
+       as.matrix(tmp)
+   }
```

Applying the function `AdMitIS` with `G.cov` leads to:

```
R> set.seed(1234)
R> outAdMitIS <- AdMitIS(KERNEL = GelmanMeng, G = G.cov, mit = outAdMit$mit, mu = c(1.459,
R> print(outAdMitIS)

$ghat
[1]  1.515 -1.152 -1.152  1.518

$NSE
[1] 0.006373 0.004616 0.004616 0.007359

$RNE
[1] 0.9376 0.7566 0.7566 0.7000
```

```
R> V <- matrix(outAdMitIS$ghat, 2, 2)
R> print(V)

       [,1]   [,2]
[1,]  1.515 -1.152
[2,] -1.152  1.518
```

V is the covariance matrix estimate. For this estimation, we have used the real mean values, i.e., `mu = c(1.459, 1.459)`, so that NSE and RNE of the covariance matrix elements are correct. In general, those mean values are unknown and we have to resort to the importance sampling estimates. In this case, the numerical standard errors of the estimated covariance matrix elements are (generally slightly) downward biased.

The function `cov2cor` can be used to obtain the correlation matrix corresponding to the covariance matrix:

```
R> cov2cor(V)

        [,1]    [,2]
[1,]  1.0000 -0.7599
[2,] -0.7599  1.0000
```

The second function provided by the R package **AdMit** which allows to find quantities of interest for the target density $p(\boldsymbol{\theta})$ using the output `outAdMit$mit` of `AdMit` is the function `AdMitMH`. This function uses the mixture approximation as the candidate density in the independence chain M-H algorithm (see Section **??**). The arguments of the function `AdMitMH` are the following:

```
AdMitMH(N = 1e+05, KERNEL, mit = list(), ...)
```

`N` is the length of the MCMC sequence of draws; `KERNEL` is a kernel function $k(\boldsymbol{\theta})$ of the target density $p(\boldsymbol{\theta})$; `mit` is a list providing information on the mixture approximation (i.e., traditionally the component `mit` in the output of the function `AdMit`); `...` allows additional parameters to be passed to the function `KERNEL`.

Let us apply the function `AdMitMH` to the kernel `GelmanMeng` using the approximation `outAdMit$mit`:

```
R> set.seed(1234)
R> outAdMitMH <- AdMitMH(KERNEL = GelmanMeng, mit = outAdMit$mit)
R> print(outAdMitMH)

$draws
               k1         k2
1       2.622e-01  2.768e+00
2       2.622e-01  2.768e+00
3       2.622e-01  2.768e+00
4       2.622e-01  2.768e+00
5       2.622e-01  2.768e+00
```

```
6       1.243e-01  2.578e+00
7       7.479e-01  2.699e+00
8       7.479e-01  2.699e+00
9       7.479e-01  2.699e+00
10      7.479e-01  2.699e+00
11     -3.903e-02  4.916e+00
12      5.862e-02  3.806e+00
13      1.539e+00  9.873e-01
14      2.911e-01  3.749e+00
15      1.339e+00  4.096e-01
16      2.703e+00  9.444e-01
17      1.564e-01  2.058e+00
18      1.564e-01  2.058e+00
19      1.960e+00  9.848e-01
20      2.676e+00  6.795e-02
 [ getOption("max.print") est atteint -- 99980 lignes omises ]

$accept
[1] 0.5274
```

The output of the function `AdMitMH` is a list of two components. The first component is `draws`, a $N \times d$ matrix containing draws from the target density $p(\boldsymbol{\theta})$ in its rows. The second component is `accept`, the acceptance rate of the independence chain M-H algorithm.

In our example, the computing time required to generate a MCMC chain of size `N = 1e+05` (i.e., the default value) takes 0.8 seconds. Note that as for the function `AdMitIS`, the most important part of the computing time is required for evaluations of the `KERNEL` function. Part of the `AdMitMH` function is implemented in `C` in order to accelerate the generation of the MCMC output. The rather high acceptance rate above 50% suggests that the mixture approximates the target density quite well.

The `R` package **coda** (**?**) can be used to check the convergence of the MCMC chain and obtain quantities of interest for $p(\boldsymbol{\theta})$. Here, for simplicity, we discard the first 1'000 draws as a burn-in sample and transform the output `outAdMitMH$draws` in a `mcmc` object using the function `as.mcmc` provided by **coda**. A summary of the MCMC chain can be obtained using `summary`:

```
R> draws <- as.mcmc(outAdMitMH$draws[1001:1e5,])
R> colnames(draws) <- c("X1", "X2")
R> summary(draws)$stat

    Mean    SD Naive SE Time-series SE
X1 1.457 1.231 0.003912       0.006518
X2 1.465 1.236 0.003927       0.006547
```

We note that the mean estimates are close to the values obtained with the function `AdMitIS`. The relative numerical efficiency can be computed from the output of the function `summary` by dividing the square of the (robust) numerical standard error of the mean estimates (i.e.,

`Time-series SE`) by the square of the naive estimator of the numerical standard error (i.e., `Naive SE`):

```
R> summary(draws)$stat[,3]^2 / summary(draws)$stat[,4]^2

    X1     X2
0.3602 0.3598
```

These relative numerical efficiencies reflect the good quality of the candidate density in the independence chain M-H algorithm.

Finally, note that for more flexibility, the functions `AdMitIS` and `AdMitMH` require the arguments `N` and `KERNEL`. Therefore, the number of sampled values `N` in importance sampling or in the independence chain M-H algorithm can be different from the number of draws `Ns` used to fit the Student-t mixture approximation. In addition, the same mixture approximation can be used for different kernel functions. This can be useful, typically in Bayesian times series econometrics, to update a joint posterior distribution with the arrival of new observations. In this case, the previous mixture approximation (i.e., fitted on a kernel function which is based on T observations) can be used as the candidate density to approximate the updated joint posterior density which accounts for the new observations (i.e., whose kernel function is based on $T + k$ observations where $k \geqslant 1$).

4. Concluding remarks

This paper presented the R package **AdMit** which provides functions to approximate and sample from a certain target distribution given only a kernel of the target density function. The estimation procedure is fully automatic and thus avoids the time-consuming and difficult task of tuning a sampling algorithm. The relevance of the package has been shown in an example of a bivariate bimodal distribution.

Interested reader are referred to **?** for a more complete description of the R package **AdMit**. In particular, we show the relevance of the AdMit procedure through the Bayesian estimation of a mixture of ARCH model fitted to foreign exchange log-returns data. The methodology is compared to standard cases of importance sampling and the Metropolis-Hastings algorithm using a naive candidate and with the Griddy-Gibbs approach. Both for investigating means and tails of the joint posterior distribution the adaptive approach is preferable.

We believe that this approach may be applicable in many fields of research and hope that the R package **AdMit** will be fruitful for many researchers like econometricians or applied statisticians.

Finally, if you use R or **AdMit**, please cite the software in publications.

Acknowledgments

The authors acknowledge three anonymous reviewers and Achim Zeileis for numerous helpful suggestions that have led to substantial improvements of the paper. The first author is grateful to the Swiss National Science Foundation (under grant #FN PB FR1-121441) for financial support. The third author gratefully acknowledges the financial assistance from the

Netherlands Organization of Research (under grant #400-07-703). Any remaining errors or shortcomings are the authors' responsibility.

Affiliation:

David Ardia
University of Neuchâtel, Switzerland Laval University, Qúbec (Qúbec), Canada
E-mail: david.ardia.ch@gmail.com