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

**David Ardia**
University of Fribourg

**Lennart F. Hoogerheide**
Erasmus University

**Herman K. van Dijk**
Erasmus University

**Abstract**

This paper presents the R package **AdMit** which 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 two examples. The first aims at illustrating in detail the use of the functions provided by the package in a bivariate bimodal distribution. The second shows the relevance of the adaptive mixture 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.

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

This paper has been published in the *Journal of Statistical Software*:

Ardia D., Hoogerheide L.F., van Dijk H.K. (2009). *Journal of Statistical Software* **29**(3), pp.1–32.

The R code of the article is available from:

http://www.jstatsoft.org/v29/i03/.

# 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 $\beta$ a constant. The uncertainty of many estimators of $\beta$ is usually represented by a symmetric Student-$t$ density (see, e.g., Heij *et al.* 2004, 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 $\beta$ 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., Hoogerheide *et al.* 2007; Hoogerheide and van Dijk 2008b). 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 Hoogerheide 2006; Hoogerheide *et al.* 2007; Hoogerheide and van Dijk 2008b, for details) and presents its R implementation (R Development Core Team 2008) with the package **AdMit** (Ardia *et al.* 2008). 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.

Hoogerheide (2006) and Hoogerheide *et al.* (2007) 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, Zeevi and Meir (1997) 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 candidate 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. Second, we consider an empirical application with the Bayesian estimation of a mixture of ARCH model applied to foreign exchange log-returns, and show the relevance of the AdMit methodology compared to standard procedures such as using a unimodal candidate in importance and M-H sampling or the Griddy-Gibbs algorithm. In particular, we illustrate that it is worthwhile to invest some computing time in an accurate importance or candidate density. This investment may become *profitable* in the sense of much quicker convergence or more reliable sampling results, especially to depict the parameter uncertainty in the tails of the joint posterior distribution.

The outline of the paper is as follows: Section 2 presents the principles of the AdMit algorithm. Section 3 presents the functions provided by the package with an illustration of a bivariate non-elliptical distribution. Section 4 compares the performance of the AdMit approach with standard strategies in a mixture of ARCH(1) model. Section 5 concludes.

## 2. Adaptive mixture of Student-$t$ distributions

The adaptive mixture of Student-$t$ distributions method developed in Hoogerheide (2006) and Hoogerheide *et al.* (2007) 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 $\eta_h$ $(h = 1, \ldots, H)$ 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 $\nu$ 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 $H$, $\eta_h$, $\boldsymbol{\mu}_h$ 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 $\nu$ 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, Geweke (1989) 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. Hoogerheide and van Dijk (2008a) 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 $\Theta$. 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 $\eta_h$ $(h = 1, \ldots, H)$ 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 $\eta_h$ $(h = 1, \ldots, H)$, 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 (1) is time consuming. The reason is that this concerns the optimization of a non-linear function of $\eta_h$ $(h = 1, \ldots, H)$ 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 $\eta_h$ $(h = 1, \ldots, H)$ 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 Hammersley and Handscomb (1965), was introduced in econometrics and statistics by Kloek and van Dijk (1978). 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, $w(\boldsymbol{\theta}) = k(\boldsymbol{\theta})/q(\boldsymbol{\theta})$, $\mathsf{E}_p$ denotes the expectation with respect to the target density $p(\boldsymbol{\theta})$ and $\mathsf{E}_q$ 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 (3):

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

where $\{\boldsymbol{\theta}^{[i]} \,|\, 1, \ldots, N\}$ is a sample of draws from the importance density $q(\boldsymbol{\theta})$. Under certain conditions (see Geweke 1989), $\widehat{g}$ 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 Metropolis *et al.* (1953) and generalized by Hastings (1970). 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 Gelman and Meng (1991) 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 Gelman and Meng 1991):

$$p(x_1, x_2) \propto \exp\left(-\tfrac{1}{2}[A x_1^2 x_2^2 + x_1^2 + x_2^2 - 2B x_1 x_2 - 2C_1 x_1 - 2C_2 x_2]\right) \;, \tag{5}$$

where $A$, $B$, $C_1$ and $C_2$ are constants. Equation (5) can be rewritten as:

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

with:

$$\boldsymbol{\mu} = \left( \frac{BC_2 + C_1}{1 - B^2} \quad \frac{BC_1 + C_2}{1 - B^2} \right)' \quad \text{and} \quad \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} 1 & -B \\ -B & 1 \end{pmatrix},$$

so the term $Ax_1^2 x_2^2$ 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** (Ardia 2007) 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 (2) 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 $\eta_h$ $(h = 1, \ldots, H)$, 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 $\eta_h$ 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 (5). 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 * C1 * x[,1] - 2 * C2 * x[,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 = 0.02)
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 the left-hand side of Figure 1. We notice the bimodal banana shape 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 (5). 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:

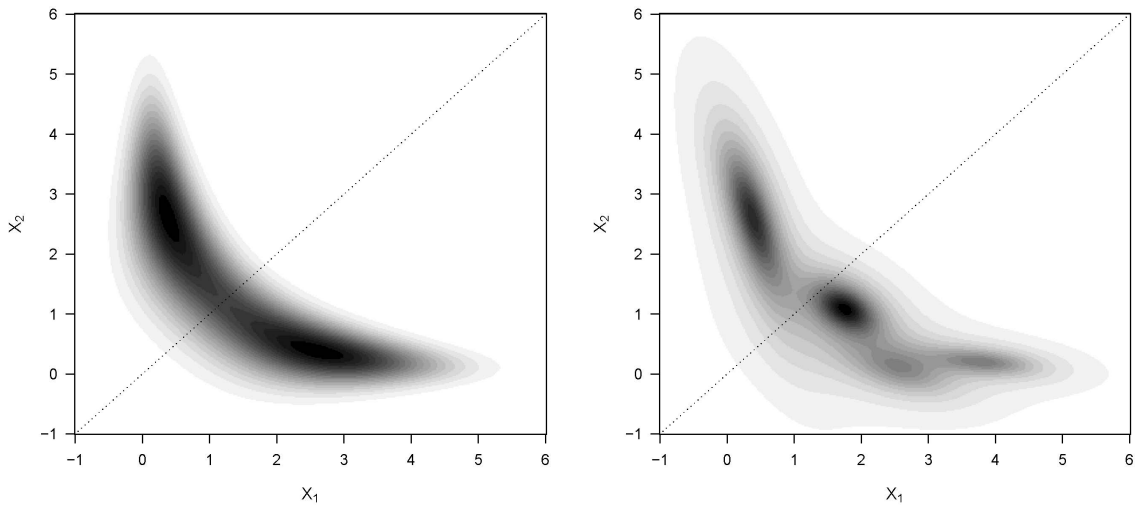


Figure 1: Left: plot of the Gelman and Meng (1991) kernel function. Right: plot of the four-component Student-$t$ mixture approximation estimated by the function `AdMit`.

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

$CV
[1] 4.8224 1.3441 0.8892 0.8315

$mit
$mit$p
  cmp1   cmp2   cmp3   cmp4
0.4464 0.1308 0.2633 0.1595

$mit$mu
        k1      k2
cmp1 0.382 2.61803
cmp2 3.828 0.20337
cmp3 1.762 1.08830
cmp4 2.592 0.06723

$mit$Sigma
       k1k1     k1k2     k2k1    k2k2
cmp1 0.2292 -0.40000 -0.40000 1.57082
cmp2 0.8477 -0.08619 -0.08619 0.07277
cmp3 0.2832 -0.10489 -0.10489 0.22971
cmp4 0.7063 -0.18383 -0.18383 0.23474

$mit$df
```

```
[1] 1

$summary
  H METHOD.mu TIME.mu METHOD.p TIME.p     CV
1 1      BFGS    0.01      NONE   0.00 4.8224
2 2      BFGS    0.04   NLMINB2   0.05 1.3441
3 3      BFGS    0.09   NLMINB2   0.11 0.8892
4 4      BFGS    0.11   NLMINB2   0.24 0.8315
```

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 (see Section 6 for computational details). The value of the coefficient of variation decreases from 4.8224 to 0.8315. A plot of the four-component approximation is displayed in the right-hand side of Figure 1. 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 (5). 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,
+          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 2.

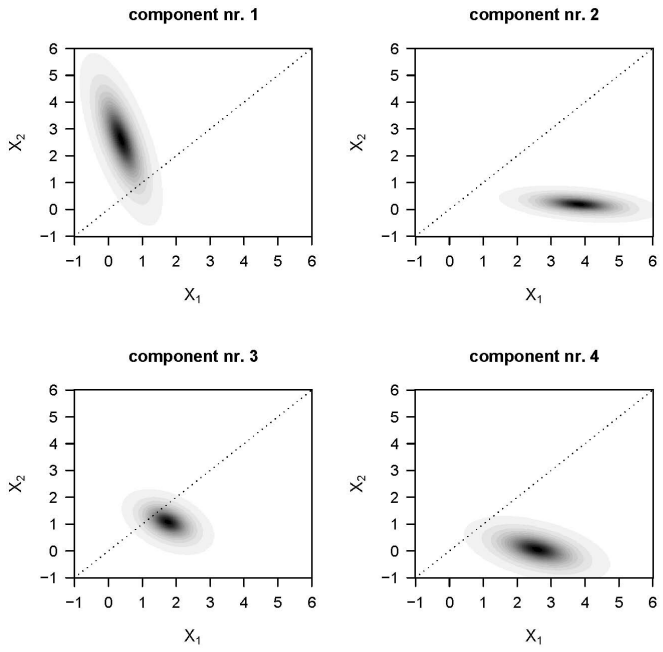


Figure 2: Student-$t$ components of the four-component mixture approximation estimated by the function `AdMit`.

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 2.1). 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 (3); `mit` is a list providing information 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] 1.458 1.460

$NSE
[1] 0.004892 0.004912

$RNE
[1] 0.6388 0.6309
```

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 (4). 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 Geweke (1989). 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, 1.459))
R> print(outAdMitIS)

$ghat
[1]  1.536 -1.166 -1.166  1.531

$NSE
[1] 0.006507 0.004644 0.004644 0.007391

$RNE
[1] 0.9128 0.7532 0.7532 0.7033

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

       [,1]   [,2]
[1,]  1.536 -1.166
[2,] -1.166  1.531
```

`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 -0.7607
[2,] -0.7607  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 2.2). 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       1.283e+00  1.669e+00
2       1.603e+00  9.873e-01
3       1.223e+00  1.872e+00
4       1.223e+00  1.872e+00
5       1.030e+00  2.306e+00
6       1.030e+00  2.306e+00
7       2.767e+00  5.180e-02
8       2.767e+00  5.180e-02
9       1.857e+00  7.651e-01
10      1.857e+00  7.651e-01
11      1.857e+00  7.651e-01
12      1.857e+00  7.651e-01
13      1.857e+00  7.651e-01
14      1.857e+00  7.651e-01
15      5.118e-01  1.941e+00
16      2.992e+00  7.749e-01
```

```
17      2.992e+00  7.749e-01
18      2.992e+00  7.749e-01
19      3.158e+00  2.375e-01
20      3.158e+00  2.375e-01
 [ reached getOption("max.print") -- omitted 99980 rows ]]

$accept
[1] 0.5272
```

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** (Plummer *et al.* 2008) 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 1.466 1.239 0.003937       0.006903
X2 1.461 1.242 0.003948       0.005751
```

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
0.3253 0.4714
```

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. Illustration II: Bayesian estimation of a mixture of ARCH(1) model

In this section, we consider the Bayesian estimation of a mixture of ARCH model. We use this example model in order to compare candidate distributions in case of a non-elliptical, four-dimensional posterior distribution in a parameter space with a restricted domain. In particular, we compare the performance of importance sampling and the independence chain M-H algorithm using a candidate density constructed by the function `AdMit` with a naive (standard) Cauchy distribution. We also consider the Griddy-Gibbs sampler of Ritter and Tanner (1992) as a benchmark.

In this application, we set the `control` parameter `IS = TRUE` in the function `AdMit`. The reason is that the (default) optimization step would quite possibly lead to unreliable scale matrices due to the pronounced restrictions on the parameter space (i.e., in the sense that most of the candidate mass might be outside of the allowed parameter region). Also, note that for a high dimensional distribution, avoiding the optimization step can substantially speed up the algorithm. The results for the four-dimensional highly non-elliptical posterior suggest the method's useful applicability in higher dimensions.

Mixture of ARCH and GARCH models have received a lot of attention in recent years as they provide an explanation for the high persistence in volatility observed with single-regime GARCH models (see, e.g., Lamoureux and Lastrapes 1990). Furthermore, these models allow for a sudden change in the (unconditional) volatility level which may lead to significant improvements in volatility forecasts (see, e.g., Dueker 1997; Klaassen 2002; Marcucci 2005).

A two-component mixture of ARCH(1) model for log-returns $\{y_t\}$ may be written as:

$$\begin{aligned} y_t &= \varepsilon_t h_t^{1/2} \quad \text{for } t = 1, \ldots, T \\ \varepsilon_t &\overset{i.i.d.}{\sim} \mathcal{N}(0,1) \\ h_t &= \begin{cases} \omega_1 + \alpha y_{t-1}^2 & \text{with probability } p \\ \omega_2 + \alpha y_{t-1}^2 & \text{with probability } (1-p)\,, \end{cases} \end{aligned} \tag{6}$$

where $\omega_1, \omega_2 > 0$, $\alpha \geqslant 0$ to ensure a positive conditional variance in each regime; $\mathcal{N}(0,1)$ is the standard Normal distribution. Model specification (6) allows to reproduce the so-called volatility clustering observed in financial returns, i.e., the fact that large changes tend to be followed by large changes (of either sign) and small changes tend to be followed by small changes. Moreover, it allows for sudden changes in the unconditional variance of the process; in the first regime, the unconditional variance is $\omega_1/(1-\alpha)$ while it is $\omega_2/(1-\alpha)$ in the second regime, provided that $\alpha < 1$. We emphasize that model (6) is used for illustrative purposes only. The assumption that the *state* (high/low volatility) is independent over time is

unrealistic and the number of regimes should be investigated. However, checking for possible misspecification of model (6) is beyond the scope of the present paper.

In order to write the likelihood function, we define the vector of observations $\mathbf{y} = (y_1 \cdots y_T)'$ and we regroup the model parameters into the vector $\boldsymbol{\theta} = (\omega_1\ \omega_2\ \alpha\ p)'$ for notational purposes. The likelihood function of $\boldsymbol{\theta}$ is then given by:

$$\mathcal{L}(\boldsymbol{\theta} \,|\, \mathbf{y}) \propto \prod_{t=2}^{T} \left\{ \frac{p}{(\omega_1 + \alpha y_{t-1}^2)^{1/2}} \exp\left[ -\frac{1}{2} \frac{y_t^2}{(\omega_1 + \alpha y_{t-1}^2)} \right] + \frac{1-p}{(\omega_2 + \alpha y_{t-1}^2)^{1/2}} \exp\left[ -\frac{1}{2} \frac{y_t^2}{(\omega_2 + \alpha y_{t-1}^2)} \right] \right\} . \tag{7}$$

We use the following proper prior densities on the model parameters:

$$\begin{aligned} p(\omega_\bullet) &\propto \phi(\omega_\bullet \,|\, 0.0, 2.0) \mathbb{I}_{\{\omega_\bullet > 0\}} \\ p(\alpha) &\propto \phi(\alpha \,|\, 0.2, 0.5) \mathbb{I}_{\{0 \leqslant \alpha < 1\}} \\ p(p) &\propto \mathbb{I}_{\{0 \leqslant p \leqslant 1\}} \ , \end{aligned} \tag{8}$$

where $\phi(\bullet \,|\, \mu, \sigma)$ denotes the Normal density with mean $\mu$ and standard deviation $\sigma$ and where we recall that $\mathbb{I}_{\{\bullet\}}$ is the indicator function which equals one if the constraint holds and zero otherwise. In addition, we require prior independence for the model parameters except for $\omega_1$ and $\omega_2$, where we require $\omega_1 < \omega_2$ for identification purposes. The prior constraint on $\alpha_1$ ensures that the model (6) is covariance-stationary in each regime. A kernel function of the joint posterior distribution is then constructed by combining the likelihood function and the joint prior via the Bayes rule. Details regarding the implementation of the kernel function for this model are provided in the Appendix.

It is important to note that we have a lack of conjugacy between the likelihood function and the joint prior density so that the joint posterior is of unknown form. Moreover, the simple Gibbs sampler cannot be used for this model since the full conditionals are also of unknown form. Alternative estimation techniques are thus required. In what follows, we consider the following strategies:

AdMit IS importance sampling using an adaptive mixture of Student-$t$ distributions as the importance density. First use the function `AdMit` with `control` parameter `IS = TRUE` (i.e., the mode and the scale matrix of the Student-$t$ components are estimated with the importance sampling weights, as in (2)). Then perform importance sampling using the function `AdMitIS` with `N = 50000` draws.

AdMit M-H independence chain M-H using an adaptive mixture of Student-$t$ distributions as the candidate density. Use the same mixture approximation as for AdMit IS, but instead of using the function `AdMitIS`, perform independence chain M-H sampling using the function `AdMitMH` with `N = 51000` draws. The first 1'000 draws are discarded as a burn-in sample.

$t_1$ IS importance sampling using a Student-$t$ distribution with one degree of freedom (i.e., Cauchy) as the importance density. First use the function `AdMit` with `control` parameter `Hmax = 1`. Then perform importance sampling using the function `AdMitIS` with `N = 50000` draws.

$t_1$ M-H independence chain M-H using a Student-$t$ distribution with one degree of freedom (i.e., Cauchy) as the candidate density. Use the same approximation as for $t_1$ IS, but instead of using the function `AdMitIS`, perform independence chain M-H sampling using the function `AdMitMH` with `N = 51000` draws. The first 1'000 draws are discarded as a burn-in sample.

GG Griddy-Gibbs sampler. Update each parameter by inversion from the full conditional distribution computed on a grid of the parameter space. Use the following grids for the model parameters:

| | |
|---|---|
| $\omega_1$ | `seq(from = 0.001, to = 0.25, by = 0.002)` |
| $\omega_2$ | `seq(from = 0.001, to = 2.0, by = 0.01)` |
| $\alpha$ | `seq(from = 0.0, to = 0.99, by = 0.008)` |
| $p$ | `seq(from = 0.0, to = 1.0, by = 0.008)` |

The kernel function is evaluated for each parameter in turn for the different values on the grid, and then a new draw is generated using the function `sample` with the corresponding probabilities (i.e., the normalized kernel values on the grid). Therefore, the approach is not *strictly speaking* the Griddy-Gibbs of Ritter and Tanner (1992) which consists in updating each parameter by inversion from the full conditional distribution computed by a deterministic integration rule since we generate new draws from a discrete distribution. However, an additional interpolation step would have slowed down even more the generation of the model parameters (which is already very slow as shown later in this section). We generate a chain of length 51'000 and discard the first 1'000 draws as a burn-in sample.

More advanced approaches have been proposed to perform an efficient Bayesian estimation of regime-switching GARCH type models. However, their implementation costs are far from negligible. The interested reader is referred to Ardia (2008) for further detail. Finally, we point out that the permutation sampler of Frühwirth-Schnatter (2001) or the permutation-augmented sampler of Geweke (2007) may be used in the context of mixture models. They are partly used to explore the unconstrained joint posterior distribution in order to find suitable identification constraints. This is not necessary here as we required $\omega_1 < \omega_2$.

We apply our Bayesian estimation methods to daily observations of the Deutschmark vs British Pound (DEM/GBP) foreign exchange log-returns. The sample period is from January 3, 1984, to December 31, 1991, for a total of 1'974 observations. The nominal returns are expressed in percent as in Bollerslev and Ghysels (1996). This data set has been proposed as an informal benchmark for GARCH time series software validation (see, e.g., McCullough and Renfro 1998) and is available from the R package **fEcofin** (Wuertz 2008) using `data("dem2gbp")`. From this time series, the first 250 observations are used to illustrate the Bayesian approach. The time series is shown in Figure 3.

The five estimation strategies are initialized with the mode of the kernel function: $\omega_1 = 0.0350$, $\omega_2 = 0.2782$, $\alpha = 0.2129$ and $p = 0.5826$. The function `AdMit` finds a four-component mixture approximation; the coefficient of variation at each iteration is 3.618 1.776 1.435 and 1.430.

Table 1 reports the estimation results for the five strategies. From this table, we note that the posterior mean estimates given by the five methodologies are very similar. This is also the case for the posterior standard deviations, except for parameter $\omega_2$. The smaller values obtained with the $t_1$ approach may suggest that the tails of the marginal posterior for $\omega_2$ is not

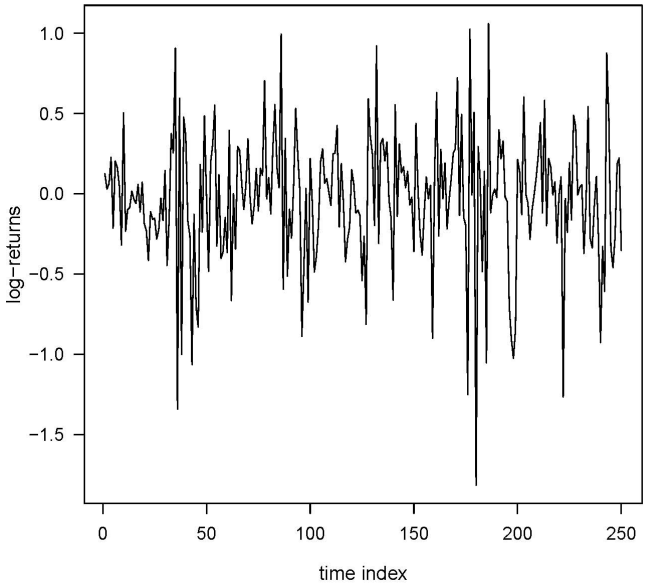


Figure 3: DEM/GBP foreign exchange log-returns (in percent, first 250 observations of the `dem2gbp` data set).

fully covered by the Student-$t$ candidate. The Griddy-Gibbs sampler is extremely slow (i.e., 3 hours) compared to the adaptive approach (i.e., 7.1 minutes) and the naive approach (i.e., 30 seconds). This illustrates that for complex problems the Griddy-Gibbs is hardly usable as a real-time method. AdMit clearly requires more time than the naive approach (i.e., 14 times slower) because of the time required for fitting the adaptive mixture candidate (i.e., 7 minutes). However, its efficiency is much better, where the largest differences between the strategies are observed for parameter $\omega_2$. In the importance sampling case, RNE is more than 14 times larger for the AdMit approach. Figure 4 illustrates the differences between both methods. AdMit requires 422 seconds for fitting the mixture candidate but after 40 seconds of sampling it already outperforms (in the sense of a higher precision) the naive approach in estimating the posterior mean of $\omega_2$.

Regarding the M-H strategy for these candidates, we also notice the better efficiency for AdMit. The autocorrelation in the MCMC output for the naive approach is much higher than for AdMit, especially for parameter $\omega_2$, as illustrated in Figure 5 and Figure 6. We note that both acceptance rates are rather high. In practice, for highly non-elliptical posterior distributions in econometric models, independence chain M-H often leads to acceptance rates below 10%. Apparently, the high autocorrelation observed for the $t_1$ M-H approach is caused by a too small candidate scale matrix; a lot of draws are generated in a small area of the parameter space which are generally accepted. Incidentally, the $t_1$ M-H sequence gets stuck in a point far away from the posterior mode (i.e., there occurs a long sequence of rejections) which implies slowly decaying autocorrelations.

The improvement of the AdMit approach over the naive approach is even more clear when focusing on the tails of the joint posterior distribution. On the left-hand side of Figure 7,

| | AdMit | | $t_1$ | | |
|---|---|---|---|---|---|
| | IS | M-H | IS | M-H | GG |
| $\mathsf{E}(\omega_1 \mid \mathbf{y})$ | 0.0452 | 0.0457 | 0.0454 | 0.0454 | 0.0450 |
| NSE ($\times 100$) | 0.0159 | 0.0272 | 0.0435 | 0.0469 | 0.0189 |
| RNE | 0.2636 | 0.0925 | 0.0361 | 0.0311 | 0.1869 |
| $\sqrt{\mathsf{VAR}(\omega_1 \mid \mathbf{y})}$ | 0.0182 | 0.0185 | 0.0185 | 0.0185 | 0.0182 |
| $\rho_1(\omega_1)$ | – | 0.731 | – | 0.780 | 0.641 |
| $\rho_{10}(\omega_1)$ | – | 0.094 | – | 0.229 | 0.070 |
| $\mathsf{E}(\omega_2 \mid \mathbf{y})$ | 0.3488 | 0.3519 | 0.3415 | 0.3390 | 0.3457 |
| NSE ($\times 100$) | 0.1503 | 0.2170 | 0.4843 | 0.4766 | 0.1501 |
| RNE | 0.1908 | 0.0885 | 0.0135 | 0.0119 | 0.1671 |
| $\sqrt{\mathsf{VAR}(\omega_2 \mid \mathbf{y})}$ | 0.1468 | 0.1443 | 0.1259 | 0.1160 | 0.1372 |
| $\rho_1(\omega_2)$ | – | 0.731 | – | 0.877 | 0.610 |
| $\rho_{10}(\omega_2)$ | – | 0.133 | – | 0.471 | 0.054 |
| $\mathsf{E}(\alpha \mid \mathbf{y})$ | 0.2324 | 0.2316 | 0.2320 | 0.2310 | 0.2330 |
| NSE ($\times 100$) | 0.0787 | 0.1179 | 0.1159 | 0.1613 | 0.0571 |
| RNE | 0.2998 | 0.1326 | 0.1392 | 0.0699 | 0.5754 |
| $\sqrt{\mathsf{VAR}(\alpha \mid \mathbf{y})}$ | 0.0964 | 0.0960 | 0.0967 | 0.0953 | 0.0969 |
| $\rho_1(\alpha)$ | – | 0.701 | – | 0.727 | 0.235 |
| $\rho_{10}(\alpha)$ | – | 0.056 | – | 0.111 | 0.009 |
| $\mathsf{E}(p \mid \mathbf{y})$ | 0.6361 | 0.6389 | 0.6337 | 0.6344 | 0.6347 |
| NSE ($\times 100$) | 0.1103 | 0.1736 | 0.2741 | 0.3143 | 0.1546 |
| RNE | 0.2893 | 0.1186 | 0.0465 | 0.0345 | 0.1471 |
| $\sqrt{\mathsf{VAR}(p \mid \mathbf{y})}$ | 0.1326 | 0.1336 | 0.1321 | 0.1306 | 0.1326 |
| $\rho_1(p)$ | — | 0.710 | – | 0.751 | 0.785 |
| $\rho_{10}(p)$ | – | 0.082 | – | 0.188 | 0.083 |
| acceptance rate | – | 0.309 | – | 0.284 | – |
| total time (sec.) | 432 | 432 | 30 | 30 | 10'885 |
| time estimation (sec.) | 422 | | 20 | | – |
| time sampling (sec.) | 10 | 10 | 10 | 10 | 10'885 |

Table 1: Posterior results for the five estimation strategies. AdMit: four-component mixture approximation; $t_1$: Student-$t$ distribution with one degree of freedom; IS: importance sampling (i.e., using the function `AdMitIS`); M-H: independence chain Metropolis-Hastings algorithm (i.e., using the function `AdMitMH`); GG: Griddy-Gibbs sampler; $\mathsf{E}(\bullet \mid \mathbf{y})$: posterior mean estimate; NSE: numerical standard error of the posterior mean estimate; RNE: relative numerical efficiency of the posterior mean estimate; $\sqrt{\mathsf{VAR}(\bullet \mid \mathbf{y})}$: posterior standard deviation estimate; $\rho_k(\bullet)$: autocorrelation at lag $k$ in the MCMC output. The number of draws is 50'000 for the five estimation strategies.

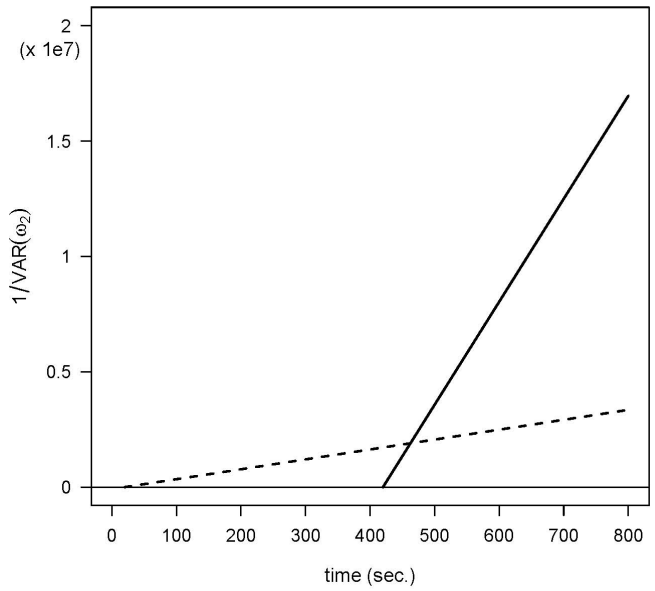


Figure 4: Precision of the importance sampling estimator of posterior mean for parameter $\omega_2$, i.e., $1/\mathsf{VAR}\big[\mathsf{E}(\omega_2 \,|\, \mathbf{y})\big]$, for the four-component mixture candidate (in solid line) and for the $t_1$ candidate (in dotted line).

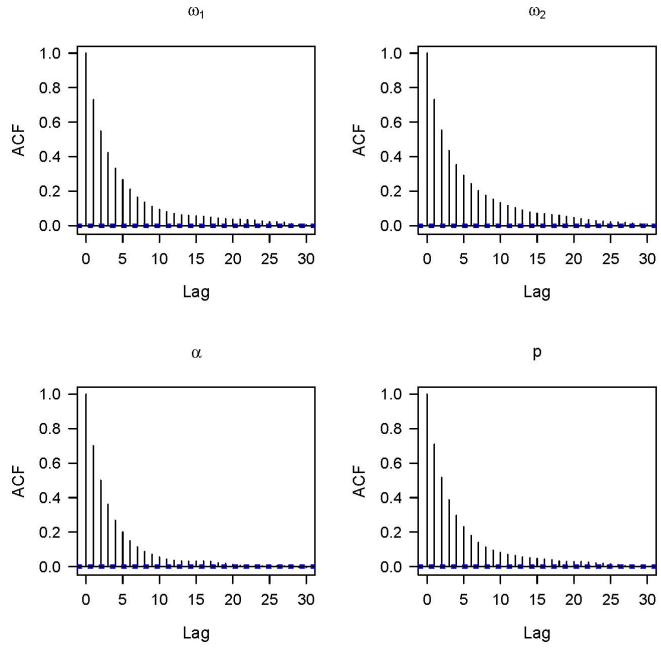


Figure 5: Autocorrelation function of the model parameters in the AdMit MH approach (i.e., using a four-component mixture approximation as the candidate density in the independence chain M-H algorithm).

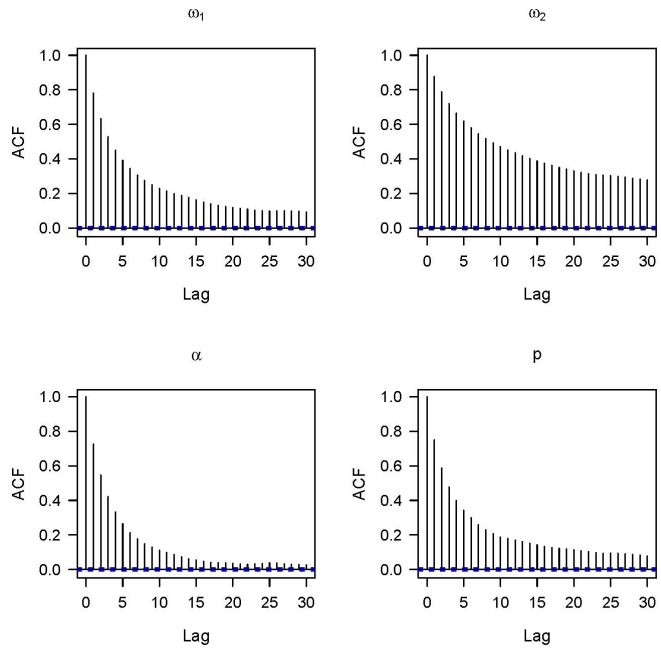


Figure 6: Autocorrelation function of the model parameters in the $t_1$ MH approach (i.e., using a $t_1$ approximation as the candidate density in the independence chain M-H algorithm).

we present the (natural) logarithm of the four-component mixture density. We note the non-elliptical shape for high values of $p$ where some components of the mixture drag some of the candidate probability mass to the right-hand side of the plot. The right-hand side of the figure displays 50'000 draws for $(\omega_2\ p)'$ generated by the independence chain M-H using the four-component mixture as the candidate density. We notice the banana shape of the marginal distribution of $(\omega_2\ p)'$. For large values of $p$, the likelihood has a small information content for parameter $\omega_2$ so that the posterior of $\omega_2$ tends to its diffuse prior. In particular, we can notice a non-negligible number of draws in the quadrant $[\omega_2 > 1; p > 0.8]$. Figure 8 presents the same type of graphs for the $t_1$ candidate. The left-hand side clearly shows the elliptical shape of the candidate density. On the right-hand side, only two draws are located in the quadrant $[\omega_2 > 1; p > 0.8]$. In this case, the naive approach is not able to detect well the mass of the joint posterior in this region. Also, far too few draws are generated in the quadrant $[0.8 < \omega_2 \leqslant 1; p > 0.8]$ compared to the AdMit approach. The marginal distribution obtained with the Griddy-Gibbs displayed in Figure 9 underlines the importance of the additional components in reproducing the non-elliptical shapes of the joint posterior. The additional time required by AdMit compared to the naive approach is therefore useful and acts as a way to robustify the Bayesian estimation of this model.

In Table 2, we report the estimated probability $\mathsf{P}(\omega_2 > \omega_2^* \,|\, p > p^*, \mathbf{y})$ for different values of $\omega_2^*$ and $p^*$ in the upper-right tail of the marginal distribution for $(\omega_2\ p)'$. The probabilities are estimated using the 50'000 draws generated by the AdMit M-H, $t_1$ M-H and Griddy-Gibbs strategies. The 95% confidence intervals (CI) of the estimated probabilities are obtained using a robust estimate of the numerical standard error (i.e., using `Time-series SE` of the `summary` function provided by the R package **coda**). From this table, we notice that the

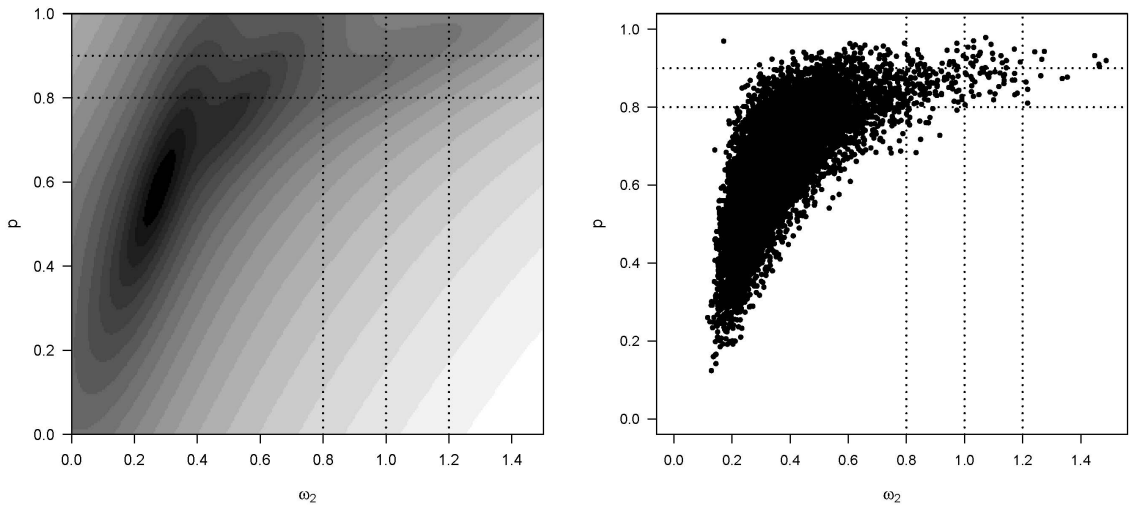


Figure 7: Left: Plot of the (natural) logarithm of the four-component mixture density. Right: 50'000 draws from the marginal distribution of $(\omega_2\ p)'$ obtained with the AdMit MH strategy (i.e., using a four-component mixture approximation as the candidate density in the independence chain M-H algorithm).

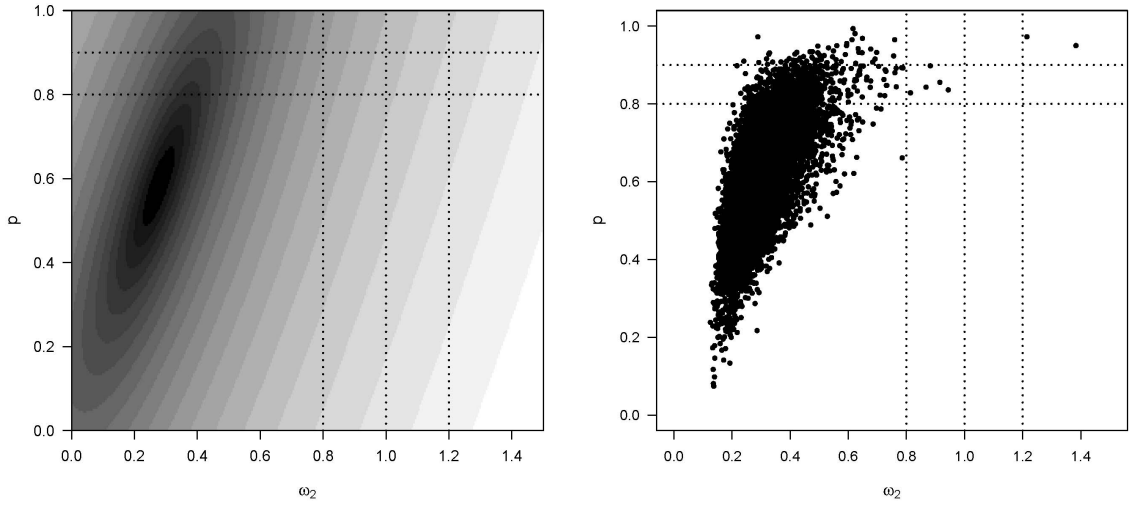


Figure 8: Left: Plot of the (natural) logarithm of the $t_1$ candidate density. Right: 50'000 draws from the marginal distribution of $(\omega_2\ p)'$ obtained with the $t_1$ M-H strategy (i.e., using a $t_1$ approximation as the candidate density in the independence chain M-H algorithm).

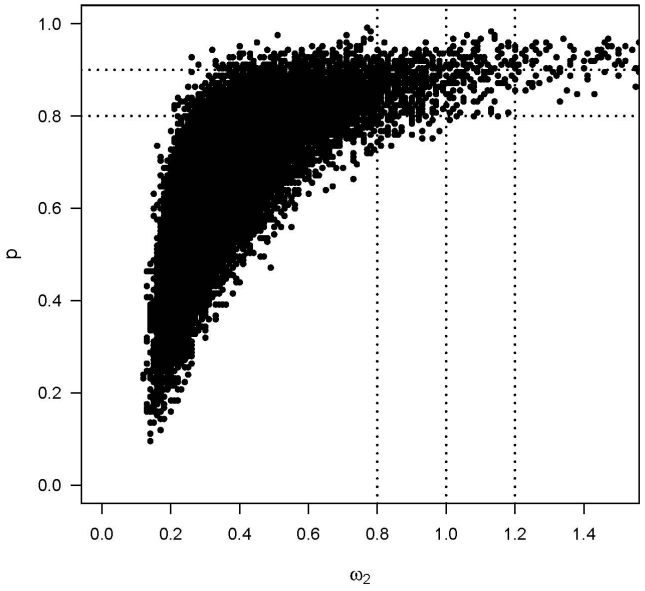


Figure 9: 50'000 draws from the marginal distribution of $(\omega_2 \; p)'$ using the Griddy-Gibbs sampler.

$t_1$ approximation completely underestimates the probability compared to the Griddy-Gibbs approach. Most of the CI given by this approach are the same due to the small amount of draws in the upper-right quadrant of the marginal distribution. These should obviously be smaller for larger $\omega_2^*$. On the other hand, the CI provided by AdMit M-H overlap the CI of the Griddy-Gibbs in every cases. The probability estimates in the extreme tail are therefore not significantly different between the AdMit M-H approach and the Griddy-Gibbs sampler.

## 5. 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 two examples. The first illustrated in detail the use of the functions provided by the package in a bivariate bimodal distribution. The second showed 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 was 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.

In a recent paper, Hoogerheide and van Dijk (2008b) illustrate the usefulness of the AdMit approach both in a bivariate posterior in an instrumental variable model and in a eight-dimensional posterior in a mixture model. We believe that this approach may be applicable

| | | $\omega_2^* = 0.8$ | | $\omega_2^* = 1.0$ | | $\omega_2^* = 1.2$ | |
|---|---|---|---|---|---|---|---|
| | $p^*$ | [ 95% CI | ] | [ 95% CI | ] | [ 95% CI | ] |
| AdMit M-H | 0.8 | 0.0934 | 0.1502 | 0.0259 | 0.0807 | -0.0010 | 0.0393 |
| | 0.9 | 0.4055 | 0.6697 | 0.2119 | 0.4679 | 0.0395 | 0.2677 |
| $t_1$ M-H | 0.8 | -0.0008 | 0.0262 | -0.0042 | 0.0132 | -0.0042 | 0.0132 |
| | 0.9 | -0.0024 | 0.1231 | -0.0024 | 0.1231 | -0.0024 | 0.1231 |
| GG | 0.8 | 0.1087 | 0.1308 | 0.0400 | 0.0561 | 0.0168 | 0.0263 |
| | 0.9 | 0.4013 | 0.4816 | 0.2206 | 0.2977 | 0.1093 | 0.1786 |

Table 2: Estimation of the probability $\mathsf{P}(\omega_2 > \omega_2^* \,|\, p > p^*, \mathbf{y})$ for different values of $\omega_2^*$ and $p^*$. AdMit M-H: independence chain M-H algorithm using a four-component mixture approximation as the candidate density; $t_1$: independence chain M-H algorithm using a Student-$t$ distribution with one degree of freedom as the candidate density; GG: Griddy-Gibbs sampler; 95% CI: 95% confidence intervals of the estimated probability $\mathsf{P}(\omega_2 > \omega_2^* \,|\, p > p^*, \mathbf{y})$ obtained using robust standard errors (i.e., using `Time-series SE` of the `summary` function provided by the R package **coda**). The number of draws in the joint posterior sample is 50'000 for the three estimation strategies.

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.

## 6. Computational details

The results in this paper were obtained using R 2.8.1 (R Development Core Team 2008) with the packages **AdMit** 1.01-01 (Ardia *et al.* 2008), **coda** 0.13-3 (Plummer *et al.* 2008), **fEcofin** 270.75 (Wuertz 2008) and **mvtnorm** 0.9-3 (Genz *et al.* 2008). R itself and all packages used are available from CRAN at `http://CRAN.R-project.org/`. Computations were performed on a Genuine Intel® dual core CPU T2400 1.83Ghz processor. Code outputs were obtained using `options(digits = 4, max.print = 40, prompt = "R> ")`. Since the functions `AdMit`, `AdMitIS` and `AdMitMH` highly rely on evaluations of the function `KERNEL`, we strongly advise the users to implement this function in a vectorized fashion. Also, implementation in lower-level languages like C or Fortran is possible using the functions `.C` and `.Fortran`.

## 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.

## References

Ardia D (2007). ***DEoptim**: Differential Evolution Optimization in R*. R package version 1.3-0, URL http://CRAN.R-project.org/package=DEoptim.

Ardia D (2008). *Financial Risk Management with Bayesian Estimation of GARCH Models: Theory and Applications*, volume 612 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, Germany. ISBN 978-3-540-78656-6. doi:10.1007/978-3-540-78657-3.

Ardia D, Hoogerheide LF, van Dijk HK (2008). ***AdMit**: Adaptive Mixture of Student-t Distributions in R*. R package version 1.01-01, URL http://CRAN.R-project.org/package=AdMit.

Bollerslev T, Ghysels E (1996). "Periodic Autoregressive Conditional Heteroscedasticity." *Journal of Business & Economic Statistics*, **14**(2), 139–151.

Dueker MJ (1997). "Markov Switching in GARCH Processes and Mean-Reverting Stock-Market Volatility." *Journal of Business & Economic Statistics*, **15**(1), 26–34.

Frühwirth-Schnatter S (2001). "Fully Bayesian Analysis of Switching Gaussian State Space Models." *Annals of the Institute of Statistical Mathematics*, **53**(1), 31–49.

Gelman A, Meng XL (1991). "A Note on Bivariate Distributions That Are Conditionally Normal." *The American Statistician*, **45**(2), 125–126.

Genz A, Bretz F, Hothorn T (2008). ***mvtnorm**: Multivariate Normal and t Distributions in R*. R package version 0.9-3, URL http://CRAN.R-project.org/package=mvtnorm.

Geweke JF (1989). "Bayesian Inference in Econometric Models Using Monte Carlo Integration." *Econometrica*, **57**(6), 1317–1339.

Geweke JF (2007). "Interpretation and Inference in Mixture Models: Simple MCMC Works." *Computational Statistics & Data Analysis*, **51**(4), 3529–3550. doi:10.1016/j.csda.2006.11.026.

Hammersley JM, Handscomb DC (1965). *Monte Carlo Methods*. Chapman & Hall. ISBN 0412158701.

Hastings WK (1970). "Monte Carlo Sampling Methods Using Markov Chains and their Applications." *Biometrika*, **57**(1), 97–109. doi:10.1093/biomet/57.1.97.

Heij C, de Boer P, Franses PH, Kloek T, van Dijk HK (2004). *Econometric Methods with Applications in Business and Economics*. Oxford University Press, Oxford, UK. ISBN 0199268010.

Hoogerheide LF (2006). *Essays on Neural Network Sampling Methods and Instrumental Variables*. Ph.D. thesis, Tinbergen Institute, Erasmus University Rotterdam. Book nr. 379 of the Tinbergen Institute Research Series.

Hoogerheide LF, Kaashoek JF, van Dijk HK (2007). "On the Shape of Posterior Densities and Credible Sets in Instrumental Variable Regression Models with Reduced Rank: An Application of Flexible Sampling Methods using Neural Networks." *Journal of Econometrics*, **139**(1), 154–180. doi:10.1016/j.jeconom.2006.06.009.

Hoogerheide LF, van Dijk HK (2008a). "Bayesian Forecasting of Value at Risk and Expected Shorfall Using Adaptive Importance Sampling." *Technical Report 2008-092/4*, Tinbergen Institute, Erasmus University Rotterdam. URL http://www.tinbergen.nl/discussionpapers/08092.pdf.

Hoogerheide LF, van Dijk HK (2008b). "Possibly Ill-Behaved Posteriors in Econometric Models: On the Connection between Model Structures, Non-elliptical Credible Sets and Neural Network Simulation Techniques." *Technical Report 2008-036/4*, Tinbergen Institute, Erasmus University Rotterdam. URL http://www.tinbergen.nl/discussionpapers/08036.pdf.

Klaassen F (2002). "Improving GARCH Volatility Forecasts with Regime-Switching GARCH." *Empirical Economics*, **27**(2), 363–394. doi:10.1007/s001810100100.

Kloek T, van Dijk HK (1978). "Bayesian Estimates of Equation System Parameters: An Application of Integration by Monte Carlo." *Econometrica*, **46**(1), 1–19.

Lamoureux CG, Lastrapes WD (1990). "Persistence in Variance, Structural Change, and the GARCH Model." *Journal of Business & Economic Statistics*, **8**(2), 225–243.

Marcucci J (2005). "Forecasting Stock Market Volatility with Regime-Switching GARCH Models." *Studies in Nonlinear Dynamics & Econometrics*, **9**(4), 1–53. Article nr. 6, URL http://www.bepress.com/snde/vol9/iss4/art6/.

McCullough BD, Renfro CG (1998). "Benchmarks and Software Standards: A Case Study of GARCH Procedures." *Journal of Economic & Social Measurement*, **25**(2), 59–71.

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953). "Equations of State Calculations by Fast Computing Machines." *Journal of Chemical Physics*, **21**(6), 1087–1092.

Plummer M, Best N, Cowles K, Vines K (2008). ***coda**: Output Analysis and Diagnostics for MCMC in R*. R package version 0.13-3, URL http://CRAN.R-project.org/package=coda.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Ritter C, Tanner MA (1992). "Facilitating the Gibbs Sampler: the Gibbs Stopper and the Griddy-Gibbs Sampler." *Journal of the American Statistical Association*, **87**(419), 861–868.

Wuertz D (2008). ***fEcofin**: Rmetrics – Economic and Financial Data Sets in R*. R package version 270-75, URL http://CRAN.R-project.org/package=fEcofin.

Zeevi AJ, Meir R (1997). "Density Estimation Through Convex Combinations of Densities: Approximation and Estimation Bounds." *Neural Networks*, **10**(1), 99–109. doi:10.1016/S0893-6080(96)00037-8.

# A. Mixture of ARCH(1) model

The implementation of the kernel function for the mixture of ARCH(1) model is realized in two steps. First, the prior (8) is implemented with the function `PRIOR`. The function `PRIOR` tests whether the constraints are fulfilled, and outputs a $(N_s \times 2)$ matrix whose first column indicates if the constraint is satisfied, and the second column returns the value of the prior at the corresponding point:

```
R> PRIOR <- function(omega1, omega2, alpha, p, log = TRUE)
+  {
+    c1 <- (omega1 > 0.0 & omega2 > 0.0 & alpha >= 0.0)
+    c2 <- (alpha < 1.0)
+    c3 <- (p > 0.0 & p < 1.0)
+    c4 <- (omega1 < omega2)
+    r1 <- c1 & c2 & c3 & c4
+    r2 <- rep.int(-Inf, length(omega1))
+    tmp <- dnorm(omega1[r1==TRUE], 0.0, 2.0, log = TRUE)
+    tmp <- tmp + dnorm(omega2[r1==TRUE], 0.0, 2.0, log = TRUE)
+    r2[r1==TRUE] <- tmp + dnorm(alpha[r1==TRUE], 0.2, 0.5, log = TRUE)
+    if (!log)
+      r2 <- exp(r2)
+    cbind(r1, r2)
+  }
```

The function `PRIOR` is coded outside the kernel function to render the program more readable and more flexible (i.e., it is more easy to modify the constraints or the hyperparameters).

The `KERNEL` function is obtained by combining the prior (8) and the likelihood function (7). We provide here a full implementation in R; an alternative coding, calling C code, is provided in the code of this article (available on the JSS website). The function `KERNEL` requires as inputs: `theta` is a $(N_s \times d)$ matrix of draws, where in our case $d = 4$; `y` is a vector of observations. The function returns a vector of $N_s$ (natural logarithm) kernel values.

```
R> KERNEL <- function(theta, y, log = TRUE)
+  {
+    if (is.vector(theta))
+      theta <- matrix(theta, nrow = 1)
+    N <- nrow(theta)
+    pos <- 2:length(y)
+
+    prior <- PRIOR(theta[,1], theta[,2], theta[,3], theta[,4])
+
+    d <- rep.int(-Inf,N)
+    for (i in 1:N)
+    {
+      if (prior[i,1] == TRUE)
+      {
+        h1 <- c(NA, theta[i,1] + theta[i,3] * y[pos-1]^2)
```

```
+         tmp1 <- -0.5 * y[pos]^2 / h1[pos] - 0.5 * log(h1[pos])
+         h2 <- c(NA, theta[i,2] + theta[i,3] * y[pos-1]^2)
+         tmp2 <- -0.5 * y[pos]^2 / h2[pos] - 0.5 * log(h2[pos])
+         tmp <- log(theta[i,4] * exp(tmp1) + (1-theta[i,4]) * exp(tmp2))
+         d[i] <- sum(tmp) + prior[i,2]
+       }
+     }
+     if (!log)
+       d <- exp(d)
+     as.numeric(d)
+   }
```

**Affiliation:**

David Ardia
Department of Quantitative Economics
University of Fribourg
CH 1700 Fribourg, Switzerland
E-mail: david.ardia@unifr.ch
URL: http://perso.unifr.ch/david.ardia/

Lennart F. Hoogerheide
Econometric and Tinbergen Institutes
Erasmus University Rotterdam
NL 3000 DR Rotterdam, The Netherlands
URL: http://people.few.eur.nl/lhoogerheide/

Herman K. van Dijk
Econometric and Tinbergen Institutes
Erasmus University Rotterdam
NL 3000 DR Rotterdam, The Netherlands
URL: http://people.few.eur.nl/hkvandijk/