LogConcDEAD: An R Package for Maximum Likelihood Estimation of a Multivariate Log-Concave Density

Madeleine Cule, Robert Gramacy and Richard Samworth
University of Cambridge

Abstract

In this document we introduce the R package LogConcDEAD (Log-concave density estimation in arbitrary dimensions). Its main function is to compute the nonparametric maximum likelihood estimator of a log-concave density. Functions for plotting, sampling from the density estimate and evaluating the density estimate are provided. All of the functions available in the package are illustrated using simple, reproducible examples with simulated data.

Keywords: log-concave density, multivariate density estimation, visualization, nonparametric statistics.

1. Introduction

1.1. About this document

This document is an introduction to the R package LogConcDEAD (log-concave density estimation in arbitrary dimensions) based on Cule et al. (2009). It aims to provide a detailed user guide based on simple, reproducible worked examples. This package is available from the Comprehensive R Archive Network at http://CRAN.R-project.org/package=LogConcDEAD. LogConcDEAD depends on MASS (Venables and Ripley 2002) for some vector operations and geometry (Grasman and Gramacy 2008) for convex hull computation. The package rgl (Adler and Murdoch 2007) is recommended for producing graphics.

This document was created using Sweave (Leisch 2002) and \LaTeX{} (Lamport 1994) using R (R Development Core Team 2008). This means that all of the code has been checked by R, and can be reproduced exactly by setting an appropriate seed (as given at the beginning of each example), or tested on different examples by using a different seed.

1.2. Log-concave density estimation

We address the fundamental statistical problem of estimating a probability density function $f_0$ from independent and identically distributed observations $X_1, \ldots, X_n$ taking values in $\mathbb{R}^d$. If a suitable parametric model is available, a common method is to use maximum likelihood to estimate the parameters of the model. Otherwise, a standard nonparametric approach
LogConcDEAD: Multivariate Log-Concave Density Estimation

is based on kernel density estimation (Wand and Jones 1995), which has been implemented in the R function density. In common with many nonparametric methods, kernel density estimation requires the careful specification of a smoothing parameter. For multivariate data, the smoothing parameter is a bandwidth matrix with up to \( \frac{1}{2}d(d + 1) \) entries to choose, meaning that this method can be especially difficult to apply in practice.

An alternative to kernel density estimation or other estimation techniques based on smoothing (all of which require the selection of a smoothing parameter, which is nontrivial especially in the multivariate case) is to impose some qualitative shape restrictions on the density. If the shape restrictions are suitable, there is enough structure to guarantee the existence of a unique and fully automatic maximum likelihood estimate, even though the class of densities may be infinite-dimensional. This therefore avoids both the restrictions of a parametric model and the difficulty of bandwidth selection in kernel density estimation. The price is some restriction on the shape of the density. However, these restrictions are less severe than those imposed by a parametric model.

Shape-constrained maximum likelihood dates back to Grenander (1956), who treated monotone densities in the context of mortality data. Recently there has been considerable interest in alternative shape constraints, including convexity, \( k \)-monotonicity and log-concavity (Groeneboom et al. 2001; Dümbgen and Rufibach 2009; Balabdaoui and Wellner 2007). However, these works have all focused on the case of univariate data.

Log-concave densities

A function \( g: \mathbb{R}^d \to [-\infty, \infty) \) is concave if

\[
g(\lambda x + (1 - \lambda)y) \geq \lambda g(x) + (1 - \lambda)g(y)
\]

for all \( x, y \in \mathbb{R}^d \) and \( \lambda \in (0, 1) \). This corresponds to what Rockafellar (1997) calls a proper concave function. We say a probability density function \( f \) is log-concave if \( \log f \) is a concave function. Several common parametric families of univariate densities are log-concave, such as Gaussian, logistic and Gumbel densities, as well as Weibull, Gamma and Beta densities for certain parameter values (An 1998). In fact, Cule et al. (2010) showed that even though the class of multivariate log-concave densities is large (infinite-dimensional), it still retains some of the simple and attractive properties of the class of Gaussian densities.

One-dimensional log-concave density estimation via maximum likelihood is discussed in Dümbgen and Rufibach (2009); computational aspects are treated in Rufibach (2007). It is in the multivariate case, however, where kernel density estimation is more difficult and parametric models less obvious, where a log-concave model may be most useful.

Theoretical and computational aspects of multivariate log-concave density estimation are treated in Cule et al. (2010). In particular, it is proved that if \( Y_1, \ldots, Y_m \) are (distinct) independent and identically distributed observations from a distribution with log-concave density \( f_0 \) on \( \mathbb{R}^d \), then (with probability 1) there is a unique log-concave density \( \hat{f}_m \) satisfying

\[
\hat{f}_m = \arg \max_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \log f(Y_i),
\]

where \( \mathcal{F} \) is the class of all log-concave densities on \( \mathbb{R}^d \). Further, it is shown that this infinite dimensional maximization problem can be reduced to that of maximizing over functions of
the form \( \bar{h}_y \) for some \( y = (y_1, \ldots, y_m) \in \mathbb{R}^m \), where
\[
\bar{h}_y(x) = \inf \{ h(x) : h \text{ is concave}, \ h(Y_i) \geq y_i, \ i = 1, \ldots, m \}. \tag{2}
\]
As discussed in Cule et al. (2010), we may think of \( \bar{h}_y \) as the function obtained by placing a pole of height \( y_i \) at \( X_i \) and stretching a rubber sheet over the top of the poles.

Therefore, to completely specify the maximum likelihood estimator, we need only specify a suitable vector \( \hat{y} \in \mathbb{R}^m \), as this defines the entire function \( \bar{h}_{\hat{y}} \). A main feature of the LogConDEAD package is that it provides an iterative algorithm for finding such an appropriate vector \( \hat{y} \).

From our knowledge of the structure of functions of the form \((2)\), we may deduce some additional properties of \( \hat{f}_m \). It is zero outside the convex hull of the data, and strictly positive inside the convex hull. Moreover, we can find a triangulation of the convex hull into simplices (triangles when \( d = 2 \), tetrahedra when \( d = 3 \), and so on) such that \( \log \hat{f}_m \) is affine on each simplex (Rockafellar 1997).

In practice our observations will be made only to a finite precision, so the observations will not necessarily be distinct. However, the same method of proof shows that, more generally, if \( X_1, \ldots, X_n \) are distinct points in \( \mathbb{R}^d \) and \( w_1, \ldots, w_n \) are strictly positive weights satisfying \( \sum_{i=1}^n w_i = 1 \), then there is a unique log-concave density \( \hat{f}_n \), which is of the form \( \hat{f}_n = \exp(\bar{h}_y) \) for some \( y \in \mathbb{R}^n \), and which satisfies
\[
\hat{f}_n = \arg \max_{f \in \mathcal{F}} \sum_{i=1}^n w_i \log f(X_i). \tag{3}
\]
The default case \( w_i = \frac{1}{n} \) corresponds to the situation described above, and is appropriate for most situations. However, the generalization (3) obtained by allowing \( w_i \neq \frac{1}{n} \) allows us to extend to binned observations. In more detail, if \( Y_1, \ldots, Y_m \) are independent and identically distributed according to a density \( f_0 \), and distinct binned values \( X_1, \ldots, X_n \) are observed, we may construct a maximum likelihood problem of the form given in (3), setting
\[
\hat{f}_n = \arg \max_{f \in \mathcal{F}} \sum_{i=1}^n w_i \log f(X_i).
\]
This generalization may also be used for a multivariate version of a log-concave EM algorithm (Chang and Walther 2007, also discussed in Cule et al. 2010).

1.3. Outline of the remainder of this document

In Section 2, we outline the algorithm used to compute the maximum likelihood estimator, including various parameters used in the computation. This is essentially an adaptation of Shor’s r-algorithm (Shor 1985) (implemented as SolvOpt by Kappel and Kuntsevich (2000)), and depends on the Quickhull algorithm for computing convex hulls (Barber et al. 1996). This section may be skipped on first reading.
In Section 3, we demonstrate the main features of the package through four simple examples (one with \( d = 1 \), two with \( d = 2 \) and one with \( d = 3 \)). This section includes a description of all of the parameters used, as well as the output structures. We also introduce the plotting functions available, as well as functions for sampling from the density estimate and for evaluating the density at a particular point.

2. Algorithm

2.1. Introduction

Recall that the maximum likelihood estimator \( \hat{f}_n \) of \( f_0 \) may be completely specified by its values at the observations \( X_1, \ldots, X_n \). Writing \( C_n \) for the convex hull of the data, Cule et al. (2010) showed that the problem of computing the estimator may be rephrased as one of finding

\[
\arg \min_{y \in \mathbb{R}^n} \sigma(y) = -\sum_{i=1}^{n} w_i y_i + \int_{C_n} \exp\{h_y(x)\} \, dx
\]

for suitable chosen weights \( w_i \), where

\[
h_y(x) = \inf\{h(x): h \text{ is concave, } h(X_i) \geq y_i, i = 1, \ldots, n\}.
\]

The function \( \sigma \) is convex, but not differentiable, so standard gradient-based convex optimization techniques such as Newton’s method are not suitable. Nevertheless, the notion of a subgradient is still valid: a subgradient at \( y \) of \( \sigma \) is any direction which defines a supporting hyperplane to \( \sigma \) at \( y \). Shor (1985) developed a theory of subgradient methods for handling convex, non-differentiable optimization problems. The \( r \)-algorithm, described in Shor (1985, Chapter 3) and implemented as \texttt{SolvOpt} in C by Kappel and Kuntsevich (2000), was found to work particularly well in practice. A main feature of the \texttt{LogConcDEAD} package is an implementation of an adaptation of this \( r \)-algorithm for the particular problem encountered in log-concave density estimation.

2.2. Shor’s \( r \)-algorithm

Our adaptation of Shor’s \( r \)-algorithm produces a sequence \( (y^t) \) with the property that

\[
\sigma(y^t) \to \min_{y \in \mathbb{R}^n} \sigma(y)
\]

as \( t \to \infty \). At each iteration, the algorithm requires the evaluation \( \sigma(y^t) \), and the subgradient at \( y^t \), denoted \( \partial \sigma(y^t) \), which determines the direction of the move to the next term \( y^{t+1} \) in the sequence.

Exact expressions for \( \sigma(y^t) \) and \( \partial \sigma(y^t) \) are provided in Cule et al. (2010). In practice, their computation requires the evaluation of convex hulls and triangulations of certain finite sets of points. This can be done in a fast and robust way via the \texttt{Quickhull} algorithm (Barber et al. 1996), available in R through the \texttt{geometry} package (Grasman and Gramacy 2008). Due to the presence of some removable singularities in the expressions for \( \sigma(y^t) \) and \( \partial \sigma(y^t) \), it is computationally more stable to use a Taylor approximation to the true values for certain
values of $y^t$ (Cule and Dümbgen 2008). The values for which a Taylor expansion (rather than
direct evaluation) is used may be controlled by the argument $J_{tol}$ to the LogConcDEAD
function mleldcd. By default this is $10^{-3}$; altering this parameter is not recommended.
Several parameters may be used to control the $r$-algorithm as detailed by Kappel and Kuntsevich (2000). In the function mlelcd, they may be controlled by the user via the arguments stepscale1, stepscale2, stepscale3, stepscale4 and desiredsize. For a detailed description of these parameters, as well as of this implementation of the $r$-algorithm, see Kappel and Kuntsevich (2000).

Stopping criteria
The implementation of the $r$-algorithm used in the main function mleldcd terminates after
the $(t + 1)$th iteration if each of the following conditions holds:

\begin{align*}
|y_{i}^{t+1} - y_{i}^{t}| & \leq \delta |y_{i}^{t}| \text{ for } i = 1, \ldots, n \quad (4) \\
|\sigma(y_{i}^{t+1}) - \sigma(y_{i}^{t})| & \leq \epsilon |\sigma(y_{i}^{t})| \quad (5) \\
\left| \int_{C \subseteq \mathbb{R}_+} \exp\{\hat{h}_{y_i}(x)\} \, dx - 1 \right| & \leq \eta \quad (6)
\end{align*}

for some small tolerances $\delta$, $\epsilon$ and $\eta$.

(4) and (5) are the criteria suggested by Kappel and Kuntsevich (2000); (6) is based on the
observation that the maximum likelihood estimator is density (Cule et al. 2010). By default,
these values are $\delta = 10^{-4}$, $\epsilon = 10^{-8}$ and $\eta = 10^{-4}$, but they may be modified by the user as
required, using the parameters $ytol$, $sigmatol$ and $integraltol$ respectively. The default
parameters have been found to work well and it is not recommended to alter them.

3. Usage
In this section we illustrate the functions available in LogConcDEAD through several simple
simulated data examples. These functions include mleldcd, which computes the maximum
likelihood estimator, as well as graphics facilities and the function rlcd for sampling from the
fitted density.

3.1. Example 1: 1-d data
For this example, we will use 100 points from a Gamma(2, 1) distribution. The seed has been
set (to 1), so this example can be reproduced exactly; you may also like to try with a different
seed.

```r
R> set.seed(1)
R> n <- 100
R> x <- sort(rgamma(n, shape=2))
R> out1 <- mlelcd(x) ## LogConcDEAD estimate
```

The visual appearance of our estimator produced by LogConcDEAD can be seen in Figure 1.
This figure is produced using the following code:
Figure 2 also illustrates the structure of the log-concave maximum likelihood estimator: its logarithm is piecewise linear with changes of slope only at observation points. This figure is produced using the following code:

```r
R> ylim <- c(-4,-1)
R> lgdtxt <- c("LogConcDEAD", "true")
R> lgdlty <- c(1,3)
R> plot(out1, uselog=TRUE, lty=1)
R> lines(x, log(x)-x, lty=3)
R> legend(x=3, y=-1, lgdtxt, lty=lgdlty)
```

For one-dimensional data, we note that the alternative active set algorithm from `logcondens` (Rufibach and Dümbgen 2006; Dümbgen et al. 2007) may also be used to compute the log-concave maximum likelihood estimator, which appears to be faster. Unsurprisingly, the output results from the two procedures are essentially the same.

### 3.2. Example 2: 2-d normal data

For this section, we will generate 100 points from a bivariate normal distribution with independent components. Again, we have set the seed (to 22) for reproducibility.

```r
R> set.seed(22)
R> d <- 2
R> n <- 100
R> x <- matrix(rnorm(n*d),ncol=d)
```

**Basic usage**

The basic command in this package is `mlelcd`, which computes the log-concave maximum likelihood estimate \( \hat{f}_n \). The `verbose` option controls the diagnostic output, which will be described in more detail below.

```r
R> out <- mlelcd(x, verbose=50)
```

<table>
<thead>
<tr>
<th>Iter</th>
<th>Function Val</th>
<th>Step Value</th>
<th>Integral</th>
<th>Grad Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>3.51842</td>
<td>0.0298976</td>
<td>0.990072</td>
<td>0.068418</td>
</tr>
<tr>
<td>100</td>
<td>3.51657</td>
<td>0.00620095</td>
<td>0.995284</td>
<td>0.057778</td>
</tr>
</tbody>
</table>

Iter # ... Function Val ... Step Value ... Integral ... Grad Norm
Figure 1: Density estimates (and true density) based on 100 i.i.d observations from a Gamma(2,1) distribution.

<table>
<thead>
<tr>
<th>Iter #</th>
<th>Function Val</th>
<th>Step Value</th>
<th>Integral</th>
<th>Grad Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>3.51625</td>
<td>0.00172758</td>
<td>0.99919</td>
<td>0.043143</td>
</tr>
<tr>
<td>200</td>
<td>3.51618</td>
<td>0.000849969</td>
<td>0.999833</td>
<td>0.045848</td>
</tr>
<tr>
<td>250</td>
<td>3.51617</td>
<td>0.000520508</td>
<td>0.999982</td>
<td>0.050969</td>
</tr>
</tbody>
</table>

SolvOpt: Normal termination.

The default print statement shows the value of the logarithm of the maximum likelihood estimator at the data points, the number of iterations of the subgradient algorithm required, and the total number of function evaluations required to reach convergence.

In the next two subsections, we will describe the input and output in more detail.

**Input**

The only input required is an $n \times d$ matrix of data points. One dimensional (vector) input will be converted to a matrix. Optionally a vector of weights $\mathbf{w}$, corresponding to $(w_1, \ldots, w_n)$ in (3), may be specified. By default this is

$$\left(\frac{1}{n}, \ldots, \frac{1}{n}\right),$$

which is appropriate for independent and identically distributed observations.
A starting value $y$ may be specified for the vector $(y_1, \ldots, y_n)$; by default a kernel density estimate (using a normal kernel and a diagonal bandwidth selected using a normal scale rule) is used. This is performed using the (internal) function `initialy`.

The parameter `verbose` controls the degree of diagnostic information provided by `SolvOpt`. The default value, $-1$, prints nothing. The value 0 prints warning messages only. If the value is $m > 0$, diagnostic information is printed every $m$th iteration. The printed information summarises the progress of the algorithm, displaying the iteration number, current value of the objective function, (Euclidean) length of the last step taken, current value of $\int \exp\{h_y(x)\} \, dx$ and (Euclidean) length of the subgradient. The last column is motivated by the fact that 0 is a subgradient only at the minimum of $\sigma$ (Rockafellar 1997, Chapter 27), and so for smooth functions a small value of the subgradient may be used as a stopping criterion. For nonsmooth functions, we may be close to the minimum even if this value is relatively large, so only the middle three columns form the basis of our stopping criteria, as described in Section 2.2.1.

The remaining optional arguments are generic parameters of the $r$-algorithm, and have already been discussed in Section 2.

**Output**

The output is an object of class "LogConcDEAD", which has the following elements:
The first two components \(x\) and \(w\) give the input data. The component \(\log\text{MLE}\) specifies the logarithm of the maximum likelihood estimator, via its values at the observation points. In this example the first 5 elements are shown, corresponding to the first 5 rows of the data matrix \(x\).

As was mentioned in Sections 1 and 2, there is a triangulation of \(C_n = \text{conv}(X_1, \ldots, X_n)\), the convex hull of the data, such that \(\log \hat{f}_n\) is affine on each simplex in the triangulation. Each simplex in the triangulation is the convex hull of a subset of \(\{X_1, \ldots, X_n\}\) of size \(d + 1\). Thus the simplices in the triangulation may be indexed by a finite set \(J\) of \((d + 1)\)-tuples, which are available via

For each \(j \in J\), there is a corresponding vector \(b_j \in \mathbb{R}^d\) and \(\beta_j \in \mathbb{R}\), which define the affine function which coincides with \(\log \hat{f}_n\) on the \(j\)th simplex in the triangulation. These values \(b_j\) and \(\beta_j\) are available in
The `LogConcDEAD` object also provides some diagnostic information on the execution of the `SolvOpt` routine: the number of iterations required, the number of function evaluations needed, the number of subgradient evaluations required (in a vector `NumberOfEvaluations`), and the minimum value of the objective function \( \sigma \) attained (\( \text{MinSigma} \)).

\[ \text{R> out$NumberOfEvaluations} \]
\[
\begin{array}{c}
[1] 282 858 283 \\
\end{array}
\]

\[ \text{R> out$MinSigma} \]
\[
\begin{array}{c}
[1] 3.516167 \\
\end{array}
\]

The indices of simplices in the convex hull \( C_n \) are available:

\[ \text{R> out$chull[1:5,]} \]
\[
\begin{array}{cc}
[1,] 18 & 46 \\
[2,] 89 & 15 \\
[3,] 33 & 46 \\
[4,] 33 & 15 \\
[5,] 100 & 20 \\
\end{array}
\]

In addition, an outward-pointing normal vector for each face of the convex hull \( C_n \) and an offset point (lying on the face of the convex hull) may be obtained.

\[ \text{R> out$outnorm[1:5,]} \]
\[
\begin{array}{cc}
[1,] 0.8500848 & 0.5266458 \\
[2,] -0.8257866 & -0.5639827 \\
[3,] 0.5578530 & -0.8299398 \\
[4,] 0.2195854 & -0.9755933 \\
[5,] -0.6899035 & 0.7239013 \\
\end{array}
\]
This information may be used to test whether or not a point lies in $C_n$, as $x \in C_n$ if and only if $p^T(x - q) \leq 0$ for every face of the convex hull, where $p$ denotes an outward normal and $q$ an offset point.

When $d = 1$, the convex hull consists simply of the minimum and maximum of the data points, and $\text{out}$$\$outnorm and $\text{out}$$\$outoffset$\$ are NULL, although $\text{out}$$\$chull$\$ still takes on the appropriate values.

### 3.3. Graphics

Various aspects of the log-concave maximum likelihood estimator can be plotted using the `plot` command, applied to an object of class "LogConcDEAD". The plots are based on interpolation over a grid, which can be somewhat time-consuming. As several will be done here, we can save the results of the interpolation separately, using the function `interplcd`, and use it to make several plots. The number of grid points may be specified using the parameter `gridlen`. By default, `gridlen`=100, which is suitable for most plots.

Where relevant, the colors were obtained by a call to `heat_hcl` in the package `colorspace` (Ihaka et al. 2008), following the recommendation of Zeileis et al. (2009). Thanks to Achim Zeileis for this helpful suggestion.

```r
R> g <- interplcd(out, gridlen=200)
R> g1 <- interpmarglcd(out, marg=1)
R> g2 <- interpmarglcd(out, marg=2)
```

The plots in Figure 3 show a contour plot of the estimator and a contour plot of its logarithm for 100 points in 2 dimensions. Note that the contours of log-concave densities enclose convex regions. This figure is produced using

```r
R> par(mfrow=c(1,2), pty="s", cex=0.7) #square plots
R> plot(out,g=g,addp=FALSE,asp=1)
R> plot(out,g=g,uselog=TRUE,addp=FALSE,asp=1)
```

If $d > 1$, we can plot one-dimensional marginals by setting the `marg` parameter. Note that the marginal densities of a log-concave density are log-concave (discussed in Cule et al. (2010), as a consequence of the theory of Prékopa (1973)). This is illustrated by Figure 4 using the following code:

```r
R> par(mfrow=c(1,2), pty="s", cex=0.7) #normal proportions
R> plot(out,marg=1,g.marg=g1)
R> plot(out,marg=2,g.marg=g2)
```

The plot type is controlled by the argument `type`, which may take the values "p" (a perspective plot), "i", "c" or "ic" (colour maps, contours or both), or "r" (a 3d plot using the `rgl` package (Adler and Murdoch 2007)). The default plot type is "ic".
Figure 3: Plots based on 100 points from a standard bivariate normal distribution

R> par(mfrow=c(1,2), pty="s", cex=0.7) #normal proportions
R> plot(out,marg=1,g.marg=g1)
R> plot(out,marg=2,g.marg=g2)

Figure 4: Plots of estimated marginal densities based on 100 points from a standard bivariate normal distribution

The \texttt{rgl} package allows user interaction with the plot (e.g. the plot can be rotated using the
mouse and viewed from different angles). Although we are unable to demonstrate this feature on paper, Figure 5 shows the type of output produced by rgl, using the following code:

\[ R> \text{plot(out, g=g, type="r")} \]

Figure 6 shows the output produced by setting `uselog = TRUE` to plot on the log scale. Here we can clearly see the structure of the log-concave density estimate. This is produced using the command

\[ R> \text{plot(out, g=g, type="r", uselog=TRUE)} \]

![Figure 5: rgl output for Example 2](image)

3.4. Other functions

In this section we will describe the use of the additional functions r1cd and d1cd.

**Sampling from the MLE**

Suppose we wish to estimate a functional of the form \( \theta(f) = \int g(x)f(x) \, dx \), for example the mean or other moments, the differential entropy \( -\int f(x) \log f(x) \, dx \), etc. Once we have obtained a density estimate \( \hat{f}_n \), such as the log-concave maximum likelihood estimator, we
may use it as the basis for a plug-in estimate $\hat{\theta}_n = \int g(x) \hat{f}_n(x) \, dx$, which may be approximated using a simple Monte Carlo procedure even if an analytic expression is not readily available. In more detail, we generate a sample $Z_1, \ldots, Z_N$ drawn from $\hat{f}_n$, and approximate $\hat{\theta}_n$ by

$$\tilde{\theta}_n = \frac{1}{N} \sum_{j=1}^{N} g(Z_j).$$

This requires the ability to sample from $\hat{f}_n$, which may be achieved given an object of class "LogConcDEAD" as follows:

```R
R> nsamp <- 1000
R> mysamp <- r1cd(nsamp,out)
```

Details of the function `r1cd` are given in Cule et al. (2010) and Gopal and Casella (2010). Once we have a random sample, plug-in estimation of various functionals is straightforward.

```R
R> colMeans(mysamp)
[1]  0.07435032  -0.23885331
R> cov(mysamp)
[,1] [,2]
[1,] 0.90688335 0.03471074
[2,] 0.03471074 0.81780823
```
Evaluation of fitted density

We may evaluate the fitted density at a point or matrix of points such as

```r
R> m <- 10
R> mypoints <- 1.5*matrix(rnorm(m*d),ncol=d)
```

using the command

```r
R> dlcd(mypoints,out)
```

```
[1] 0.09980597 0.06860964 0.00000000 0.00000000 0.05313693 0.04702261
[7] 0.09377615 0.10284397 0.00000000 0.10559162
```

Note that, as expected, the density estimate is zero for points outside the convex hull of the original data.

The `dlcd` function may be used in conjunction with `rlcd` to estimate more complicated functionals such as a $100(1 - \alpha)\%$ highest density region, defined by Hyndman (1996) as $R_\alpha = \{x \in \mathbb{R}^d : f(x) \geq f_\alpha\}$, where $f_\alpha$ is the largest constant such that $\int_{R_\alpha} f(x) \, dx \geq 1 - \alpha$. Using the algorithm outlined in Hyndman (1996, Section 3.2), it is straightforward to approximate $f_\alpha$ as follows:

```r
R> myval <- sort(dlcd(mysamp,out))
R> alpha <- c(.25,.5,.75)
R> myval[(1-alpha)*nsamp]
```

```
[1] 0.12306490 0.08408260 0.05890699
```

3.5. Example 3: 2-d binned data

In this section, we demonstrate the use of `LogConcDEAD` with binned data. The seed here has been set to 333 for reproducibility; you may wish to try these examples with other seeds.

We generate some data from a normal distribution with correlation using the package `mvtnorm` (Genz et al. 2008). As before, this may be installed and loaded using

```r
R> install.packages("mvtnorm")
R> library("mvtnorm")
R> set.seed(333)
R> sigma <- matrix(c(1,0.2,0.2,1),nrow=2)
R> d <- 2
R> n <- 100
R> y <- rmvnorm(n,sigma=0.1*sigma)
R> xall <- round(y,digits=1)
```
The matrix xall therefore contains 100 observations, rounded to 1 decimal place; there are in total 70 distinct observations. In order to compute an appropriate log-likelihood, we will use the function getweights to extract a matrix of distinct observations and a vector of weights for use in mlelcd. The result of this has two parts: a matrix x consisting of the distinct observations, and a vector w of weights. We may also use interplcd as before to evaluate the estimator on a grid for plotting purposes.

R> tmpw <- getweights(xall)
R> outw <- mlelcd(tmpw$x,w=tmpw$w)
R> gw <- interplicd(outw, gridlen=200)

In Figure 7 we plot density and log-density estimates using a contour plot as before. In contrast to the examples in Figure 3, we have addp=TRUE (the default), which superposes the observation points on the plots, and drawlabels=FALSE, which suppresses the contour labels. The code to do this is

R> par(mfrow=c(1,2), pty="s", cex=0.7) #2 square plots
R> plot(outw,g=gw,asp=1,drawlabels=FALSE)
R> plot(outw,g=gw,uselog=TRUE,asp=1,drawlabels=FALSE)

Figure 7: Density and log density estimate based on 100 points from a bivariate normal distribution in two dimensions (truncated to 1 decimal place) (Example 3)

3.6. Example 4: Higher-dimensional data

In our final example we illustrate the use of the log-concave density estimate for higher-dimensional data. For this example the seed has been set to 4444. The log-concave maximum
likelihood estimator is defined and may be computed and evaluated in exactly the same way as the 2-dimensional examples in Sections 3.2 and 3.5. This estimate will be based on 100 points.

```r
R> set.seed(4444)
R> d <- 3
R> n <- 100
R> x <- matrix(rgamma(n*d,shape=2),ncol=d)
R> out3 <- mlelcd(x)
```

The function `dmarglcd` may be used to evaluate the marginal estimate, setting the parameter `marg` appropriately. Note that, as before, the estimate is 0 outside the convex hull of the observed data.

```r
R> mypoints <- c(0,2,4)
R> dmarglcd(mypoints, out3, marg=1)
```

```
[1] 0.00000000 0.28450858 0.07586891
```

One-dimensional marginal distributions may be plotted easily, by setting the `marg` parameter to the appropriate margin as shown in Figure 8 using the following:

```r
R> par(mfrow=c(2,2),cex=0.8)
R> plot(out3, marg=1)
R> plot(out3, marg=2)
R> plot(out3, marg=3)
R> tmp <- seq(min(out3$x), max(out3$x),len=100)
R> plot(tmp, dgamma(tmp,shape=2), type="l",
+ xlab="X", ylab="true marginal density")
R> title(main="True density")
```
Figure 8: Marginal density estimates for 3 dimensional data (based on 100 points)
References


**Affiliation:**

Madeleine Cule, Robert Gramacy, Richard Samworth
Statistical Laboratory
Centre for Mathematical Sciences
Wilberforce Road
Cambridge CB3 0WG
E-mail: {mlc40,bobby,rjs57}@statslab.cam.ac.uk
URL: http://www.statslab.cam.ac.uk/~{mlc40,bobby,rjs57}