Package 'logcondens'

July 22, 2025

Type Package

Title Estimate a Log-Concave Probability Density from Iid Observations

Version 2.1.8

Date 2023-08-21

Author Kaspar Rufibach <kaspar.rufibach@gmail.com> and Lutz Duembgen <duembgen@stat.unibe.ch>

Maintainer Kaspar Rufibach <kaspar.rufibach@gmail.com>

Depends R (>= 2.10)

Imports ks, graphics, stats

Description Given independent and identically distributed observations X(1), ..., X(n), compute the maximum likelihood estimator (MLE) of a density as well as a smoothed version of it under the assumption that the density is log-concave, see Rufibach (2007) and Duembgen and Rufibach (2009). The main function of the package is 'logConDens' that allows computation of the log-concave MLE and its smoothed version. In addition, we provide functions to compute (1) the value of the density and distribution function estimates (MLE and smoothed) at a given point (2) the characterizing functions of the estimator, (3) to sample from the estimated distribution, (5) to compute a two-sample permutation test based on log-concave densities, (6) the ROC curve based on log-concave estimates within cases and controls, including confidence intervals for given values of false positive fractions (7) computation of a confidence interval for the value of the true density at a fixed point. Finally, three datasets that have been used to illustrate log-concave density estimation are made available.

License GPL (>= 2)

URL http://www.kasparrufibach.ch ,

https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-22 13:20:02 UTC

Contents

`logcon-package` *Estimate a Log-Concave Probability Density from iid Observations*

Description

The main function of this package is logConDens: compute the maximum likelihood estimator (MLE) of a log-concave density from one-dimensional i.i.d. observations as well as the kernel smoothed version derived from it. A list of additional functions that can be used to compute quantities relevant in that context can be found below.

Two algorithms are offered to compute the estimate: An active set (logConDens) and an iterative algorithm based on the pool-adjacent-violaters algorithm (icmaLogCon). The latter of these functions is only part of this package for historical reasons: it was the first method that was proposed to estimate a log-concave density, see Rufibach (2007). The more efficient way of computing the estimate is via an active set algorithm. The smoothed versions of the log-concave density and distribution function estimates discussed in Section 3 of Duembgen and Rufibach (2009) are available in the function evaluateLogConDens.

To compute a log-concave density, CDF, and survival function from interval- or right-censored observations use the package **logconcens**. A log-concave probability mass function can be computed using the package **logcondiscr**, see Balabdaoui et al (2012) for details. For the computation of a log-concave estimate in any dimension the package **LogConDEAD** can be used.

Details

Package:	logcondens
Type:	Package
Version:	2.1.8
Date:	2023-08-21
License:	GPL (>=2)

The following additional functions and datasets are provided in the package:

evaluateLogConDens Computes the value of the estimated log-density, density, and distribution function of the MLE and the smoothed estimator at a given x0.

quantilesLogConDens Computes quantiles of the estimated distribution function at a given x0.

intECDF Compute the integrated empirical distribution function of the observations at a given x0.

intF Compute the integral of the distribution function corresponding to the log-concave density estimator at a given x0.

logconTwoSample Compute a permutation test for the difference between two distribution functions.

logConROC Compute ROC curve based on log-concave density estimates within cases and controls.

confIntBootLogConROC_t0 Compute bootstrap confidence intervals at given false positive fractions (= 1 - specificity) for the ROC curve based on log-concave density estimates.

logConCI Compute a confidence interval for the value of the true density at a fixed point, based on the theory developed in Balabdaoui et al (2009). This function was contributed by Mahdis Azadbakhsh and Hanna Jankowski, see Azadbakhsh et al (2012).

isoMean Compute the weighted least squares regression under a monotonicity constraint (the Grenander estimator). This function is used as part of icmaLogCon but is also of independent interest.

The following datasets have been used in several publications to illustrate log-concave density estimation and are therefore included in this package:

`reliability` Dataset that contains the data analyzed in Duembgen and Rufibach (2009, Figure 2). See the help file for `logConDens` for the analysis of this data.

`brightstar` Dataset that contains the data analyzed in Mizera and Koenker (2009, Section 5). The sample consists of measurements of radial and rotational velocities for the stars from the Bright Star Catalog, see Hoffleit and Warren (1991).

`pancreas` Data from pancreatic cancer serum biomarker study, first published by Wieand et al (1989). Contains data on serum measurements for a cancer antigen (CA-125) and a carbohydrate antigen (CA19.9) both of which are measured on a continuous non-negative scale. The measurements were taken within a case-control study on 90 cases with pancreatic cancer and 51 controls who did not have cancer but pancreatitis.

A `print` and `summary` for objects of class `dlc`, generated by `logConDens`, are available as well.

Author(s)

Kaspar Rufibach (maintainer), <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

Kaspar Rufibach gratefully acknowledges support by the Swiss National Science Foundation SNF, http://www.snf.ch.

MatLab code with an implementation of the active set algorithm is available on request from Lutz Duembgen.

References

Azadbakhsh, M., Jankowski, H. and Gao, X. (2014). Computing confidence intervals for log-concave densities. *Comput. Statist. Data Anal.*, **75**, 248–264.

Balabdaoui, F., Jankowski, H., Rufibach, K. and Pavlides, M. (2012). Asymptotics of the discrete log-concave maximum likelihood estimator and related applications. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **75(4)**, 769–790.

Baladbaoui, F., Rufibach, K. and Wellner, J. (2009). Limit distribution theory for maximum likelihood estimation of a log-concave density. *Ann. Statist.*, **37(3)**, 1299–1331.

Duembgen, L., Huesler, A. and Rufibach, K. (2010). Active set and EM algorithms for log-concave densities based on complete and censored data. Technical report 61, IMSV, Univ. of Bern, available at https://arxiv.org/abs/0707.4643.

Duembgen, L. and Rufibach, K. (2009). Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011). logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Hoffleit, D., Warren, W.H. (1991). *The Bright Star Catalog.* Yale University Observatory, New Heaven.

Mizera, I., Koenker, R. (2010). Quasi-concave density estimation. *Ann. Statist.*, **38(5)**, 2998–3027.

Rufibach K. (2006). *Log-concave Density Estimation and Bump Hunting for i.i.d. Observations.* PhD Thesis, University of Bern, Switzerland and Georg-August University of Goettingen, Germany, 2006.
Available at https://slsp-ube.primo.exlibrisgroup.com/permalink/41SLSP_UBE/17e6d97/alma99116730175505511.

Rufibach, K. (2007). Computing maximum likelihood estimators of a log-concave density function. *J. Stat. Comput. Simul.* **77**, 561–574.

Rufibach, K. (2012). A smooth ROC curve estimator based on log-concave density estimates. *Int. J. Biostat.*, **8**(1), 1–29.

Examples

```
## estimate gamma density
set.seed(1977)
x <- rgamma(100, 2, 1)
res <- logConDens(x, smoothed = FALSE, print = TRUE)
summary(res)

## compare performance to ICMA
res2 <- icmaLogCon(x, T1 = 2000, robustif = TRUE, print = TRUE)

res$L
res2$L

## plot resulting functions
par(mfrow = c(2, 2), mar = c(3, 2, 1, 2))
plot(res, which = "density")
plot(res, which = "log-density")
plot(res, which = "CDF")
xli <- range(res$x) + 0.1 * c(-1, 1) * diff(range(res$x))
plot(res$x, res$H, type = 'l', xlim = xli, yli = c(min(res$H) * 1.1, 0))
segments(res$knots, 0, res$knots, min(res$H) * 1.1, lty = 2)
rug(res$x); abline(h = 0, lty = 2)

## compute function values at an arbitrary point
x0 <- (res$x[50] + res$x[51]) / 2
evaluateLogConDens(x0, res)

## compute 0.5 quantile of Fhat
quantilesLogConDens(0.5, res)
```

`activeSetLogCon` *Computes a Log-Concave Probability Density Estimate via an Active Set Algorithm*

Description

Given a vector of observations $\boldsymbol{x}_n = (x_1, \ldots, x_n)$ with not necessarily equal entries, `activeSetLogCon` first computes vectors $\boldsymbol{x}_m = (x_1, \ldots, x_m)$ and $\boldsymbol{w} = (w_1, \ldots, w_m)$ where w_i is the weight of each

x_i s.t. $\sum_{i=1}^m w_i = 1$. Then, `activeSetLogCon` computes a concave, piecewise linear function $\widehat{\phi}_m$ on $[x_1, x_m]$ with knots only in $\{x_1, \ldots, x_m\}$ such that

$$L(\phi) = \sum_{i=1}^{m} w_i \phi(x_i) - \int_{-\infty}^{\infty} \exp(\phi(t)) dt$$

is maximal. To accomplish this, an active set algorithm is used.

Usage

```
activeSetLogCon(x, xgrid = NULL, print = FALSE, w = NA)
```

Arguments

`x`	Vector of independent and identically distributed numbers, not necessarily unique.
`xgrid`	Governs the generation of weights for observations. See `preProcess` for details.
`print`	`print = TRUE` outputs the log-likelihood in every loop, `print = FALSE` does not. Make sure to tell `R` to output (press CTRL+W).
`w`	Optional vector of weights. If weights are provided, i.e. if `w != NA`, then `xgrid` is ignored.

Value

`xn`	Vector with initial observations $x_1, \ldots, x_n$.
`x`	Vector of observations $x_1, \ldots, x_m$ that was used to estimate the density.
`w`	The vector of weights that had been used. Depends on the chosen setting for `xgrid`.
`phi`	Vector with entries $\widehat{\phi}_m(x_i)$.
`IsKnot`	Vector with entries $\text{IsKnot}_i = 1\{\widehat{\phi}_m$ has a kink at $x_i\}$.
`L`	The value $L(\widehat{\boldsymbol{\phi}}_m)$ of the log-likelihood-function L at the maximum $\widehat{\boldsymbol{\phi}}_m$.
`Fhat`	A vector $(\widehat{F}_{m,i})_{i=1}^m$ of the same size as $\boldsymbol{x}$ with entries $$\widehat{F}_{m,i} = \int_{x_1}^{x_i} \exp(\widehat{\phi}_m(t)) dt.$$
`H`	Vector $(H_1, \ldots, H_m)'$ where H_i is the derivative of $$t \to L(\phi + t\Delta_i)$$ at zero and $\Delta_i(x) = \min(x - x_i, 0)$.
`n`	Number of initial observations.
`m`	Number of unique observations.
`knots`	Observations that correspond to the knots.
`mode`	Mode of the estimated density $\hat{f}_m$.
`sig`	The standard deviation of the initial sample $x_1, \ldots, x_n$.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L, Huesler, A. and Rufibach, K. (2010) Active set and EM algorithms for log-concave densities based on complete and censored data. Technical report 61, IMSV, Univ. of Bern, available at https://arxiv.org/abs/0707.4643.

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

See Also

activeSetLogCon can be used to estimate a log-concave density. However, to generate an object of class dlc that allows application of summary and plot we recommend to use logConDens.

The following functions are used by activeSetLogCon:

J00, J10, J11, J20, Local_LL, Local_LL_all, LocalCoarsen, LocalConvexity, LocalExtend, LocalF, LocalMLE, LocalNormalize, MLE

Log concave density estimation via an iterative convex minorant algorithm can be performed using icmaLogCon.

Examples

```
## estimate gamma density
set.seed(1977)
n <- 200
x <- rgamma(n, 2, 1)
res <- activeSetLogCon(x, w = rep(1 / n, n), print = FALSE)

## plot resulting functions
par(mfrow = c(2, 2), mar = c(3, 2, 1, 2))
plot(res$x, exp(res$phi), type = 'l'); rug(x)
plot(res$x, res$phi, type = 'l'); rug(x)
plot(res$x, res$Fhat, type = 'l'); rug(x)
plot(res$x, res$H, type = 'l'); rug(x)

## compute and plot function values at an arbitrary point
x0 <- (res$x[100] + res$x[101]) / 2
Fx0 <- evaluateLogConDens(x0, res, which = 3)[, "CDF"]
plot(res$x, res$Fhat, type = 'l'); rug(res$x)
abline(v = x0, lty = 3); abline(h = Fx0, lty = 3)

## compute and plot 0.9-quantile of Fhat
```

```
q <- quantilesLogConDens(0.9, res)[2]
plot(res$x, res$Fhat, type = 'l'); rug(res$x)
abline(h = 0.9, lty = 3); abline(v = q, lty = 3)
```

---

activeSetRoutines *Auxiliary Numerical Routines for the Function activeSetLogCon*

Description

Functions that are used by activeSetLogCon.

Usage

```
LocalCoarsen(x, w, IsKnot)
LocalConvexity(x, phi)
LocalExtend(x, IsKnot, x2, phi2)
LocalF(x, phi)
LocalNormalize(x, phi)
LocalMLE(x, w, IsKnot, phi_o, prec)
LocalVariance(x, w = NULL, phi)
```

Arguments

x	Vector of independent and identically distributed numbers, with strictly increasing entries.
w	Optional vector of nonnegative weights corresponding to $\boldsymbol{x}$.
IsKnot	Vector with entries $\text{IsKnot}_i = 1\{\phi \text{ has a kink at } x_i\}$.
phi	Vector with entries $\phi(x_i)$.
x2	Vector of same type as $\boldsymbol{x}$.
phi2	Vector of same type as $\boldsymbol{\phi}$.
phi_o	Optional starting vector.
prec	Threshold for the directional derivative during Newton-Raphson procedure.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

See Also

All the above functions are used by activeSetLogCon to estimate a log-concave probability density.

Log concave density estimation via an iterative convex minorant algorithm can be performed using icmaLogCon.

brightstar *Bright star dataset used to illustrate log-concave density estimation*

Description

Dataset that contains the data analyzed in Mizera and Koenker (2009, Section 5). The sample consists of measurements of radial and rotational velocities for the stars from the Bright Star Catalog, see Hoffleit and Warren (1991).

Usage

```
data(brightstar)
```

Format

A data frame with 9092 rows on the following 2 variables.

`nr` Location of measurements.

`rad` Measurements of radial velocities.

`rot` Measurements of rotational velocities.

References

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Hoffleit, D., Warren, W.H. (1991). *The Bright Star Catalog.* Yale University Observatory, New Heaven.

Mizera, I., Koenker, R. (2010). Quasi-concave density estimation. *Ann. Statist.*, **38(5)**, 2998–3027.

Examples

```
# ---- load rotational velocity data ----
data(brightstar)

# ---- compute and plot log-concave estimate ----
# See also Figure 3 in Koenker & Mizera (2009)
x0 <- sort(brightstar[, 3])
res <- logConDens(x0, print = FALSE, smoothed = FALSE)
plot(res, which = "density")
```

`confIntBootLogConROC_t0` *Function to compute a bootstrap confidence interval for the ROC curve at a given t, based on the log-concave ROC curve*

Description

This function computes a bootstrap confidence interval for the ROC curve at a given value false negative fraction (1 - specificity) t. The ROC curve estimate is based on log-concave densities, as discussed in Rufibach (2011).

Usage

```
confIntBootLogConROC_t0(controls, cases, grid = c(0.2, 0.8), conf.level = 0.95,
M = 1000, smooth = TRUE, output = TRUE)
```

Arguments

`cases`	Values of the continuous variable for the cases.
`controls`	Values of the continuous variable for the controls.
`grid`	Values of 1 - specificity where confidence intervals should be computed at (may be a vector).
`conf.level`	Confidence level of confidence interval.
`M`	Number of bootstrap replicates.
`smooth`	`Logical`. Compute confidence interval also for ROC curve estimate based on smoothed log-concave densities.
`output`	`Logical`. Show progress of computations?

Value

A list containing the following elements:

`qs`	`data.frame` with the columns `t` (false positive fractions where confidence interval is computed at) and the confidence intervals for the ROC curve at `grid`, based on the log-concave density estimate.
`boot.mat`	Bootstrap samples for the ROC curve based on the log-concave density estimate.
`qs.smooth`	If `smooth = TRUE`, same as `qs` but for the ROC curve based on the smooth log-concave density estimate.
`boot.mat.smooth`	If `smooth = TRUE`, bootstrap samples for the ROC curve based on the smoothed log-concave density estimate.

Note

The confidence intervals are only valid if observations are *independent*, i.e. eacht patient only contributes one measurement, e.g.

Author(s)

Kaspar Rufibach (maintainer)
<kaspar.rufibach@gmail.com>
http://www.kasparrufibach.ch.

References

The reference for computation of these bootstrap confidence intervals is:

Rufibach, K. (2012). A smooth ROC curve estimator based on log-concave density estimates. *Int. J. Biostat.*, **8**(1), 1–29.

The bootstrap competitor based on the empirical ROC curve is described in:

Zhou, X.H. and Qin, G. (2005). Improved confidence intervals for the sensitivity at a fixed level of specificity of a continuous-scale diagnostic test. *Statist. Med.*, **24**, 465–477.

See Also

The ROC curve based on log-concave density estimates can be computed using `logConROC`. In the example below we analyze the `pancreas` data.

Examples

```
## Not run:
## ROC curve for pancreas data
data(pancreas)
status <- factor(pancreas[, "status"], levels = 0:1, labels = c("healthy", "diseased"))
var <- log(pancreas[, "ca199"])
cases <- var[status == "diseased"]
controls <- var[status == "healthy"]

## compute confidence intervals
res <- confIntBootLogConROC_t0(controls, cases, grid = c(0.2, 0.8), conf.level = 0.95,
    M = 1000, smooth = TRUE, output = TRUE)
res

## End(Not run)
```

---

evaluateLogConDens *Evaluates the Log-Density MLE and Smoothed Estimator at Arbitrary Real Numbers xs*

Description

Based on a `"dlc"` object generated by `logConDens`, this function computes the values of

$$\widehat{\phi}_m(t)$$

$$\widehat{f}_m(t) = \exp(\widehat{\phi}_m(t))$$

$$\widehat{F}_m(t) = \int_{x_1}^{t} \exp(\widehat{\phi}_m(x))dx$$

$$\widehat{f}^*_m(t) = \exp(\widehat{\phi}^*_m(t))$$

$$\widehat{F}^*_m(t) = \int_{x_1}^{t} \exp(\widehat{\phi}^*_m(x))dx$$

at all real number t in `xs`. The exact formula for $\widehat{F}_m$ and $t \in [x_j, x_{j+1}]$ is

$$\widehat{F}_m(t) = \widehat{F}_m(x_j) + (x_{j+1} - x_j)J\Big(\widehat{\phi}_j, \widehat{\phi}_{j+1}, \frac{t - x_j}{x_{j+1} - x_j}\Big)$$

for the function J introduced in `Jfunctions`. Closed formulas can also be given for $\widehat{f}^*_m(t)$ and $\widehat{F}^*_m(t)$.

Usage

```
evaluateLogConDens(xs, res, which = 1:5, gam = NULL, print = FALSE)
```

Arguments

`xs`	Vector of real numbers where the functions should be evaluated at.
`res`	An object of class `"dlc"`, usually a result of a call to `logConDens`.
`which`	A (sub-)vector of `1:5` specifying which of the above quantities should be computed.
`gam`	Only necessary if `smoothed = TRUE`. The standard deviation of the normal kernel. If equal to `NULL`, `gam` is chosen such that the variances of the original sample $x_1, \ldots, x_n$ and $\widehat{f}^*_n$ coincide. See `logConDens` for details.
`print`	Progress in computation of smooth estimates is shown.

Value

Matrix with rows $(x_{0,i}, \widehat{\phi}_m(x_{0,i}), \widehat{f}_m(x_{0,i}), \widehat{F}_m(x_{0,i}), \widehat{f}^*_m(x_{0,i}), \widehat{F}^*_m(x_{0,i}))$ where $x_{0,i}$ is the i-th entry of `xs`.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

Examples

```
## estimate gamma density
set.seed(1977)
x <- rgamma(200, 2, 1)
res <- logConDens(x, smoothed = TRUE, print = FALSE)

## compute function values at an arbitrary point
xs <- (res$x[100] + res$x[101]) / 2
evaluateLogConDens(xs, res)

## only compute function values for non-smooth estimates
evaluateLogConDens(xs, res, which = 1:3)
```

`icmaLogCon` *Computes a Log-Concave Probability Density Estimate via an Iterative Convex Minorant Algorithm*

Description

Given a vector of observations $\boldsymbol{x}_n = (x_1, \ldots, x_n)$ with not necessarily equal entries, `activeSetLogCon` first computes vectors $\boldsymbol{x}_m = (x_1, \ldots, x_m)$ and $\boldsymbol{w} = (w_1, \ldots, w_m)$ where w_i is the weight of each x_i s.t. $\sum_{i=1}^m w_i = 1$. Then, `activeSetLogCon` computes a concave, piecewise linear function $\widehat{\phi}_m$ on $[x_1, x_m]$ with knots only in $\{x_1, \ldots, x_m\}$ such that

$$L(\phi) = \sum_{i=1}^{m} w_i \phi(x_i) - \int_{-\infty}^{\infty} \exp(\phi(t)) dt$$

is maximal. In order to be able to apply the pool - adjacent - violaters algorithm, computations are performed in the parametrization

$$\boldsymbol{\eta}(\boldsymbol{\phi}) = \Big(\phi_1, \Big(\eta_1 + \sum_{j=2}^{i} (x_i - x_{i-1})\eta_i\Big)_{i=2}^{m}\Big).$$

To find the maximum of L, a variant of the iterative convex minorant using the pool - adjacent - violaters algorithm is used.

Usage

```
icmaLogCon(x, xgrid = NULL, eps = 10^-8, T1 = 2000,
    robustif = TRUE, print = FALSE)
```

Arguments

x	Vector of independent and identically distributed numbers, not necessarily equal.
xgrid	Governs the generation of weights for observations. See preProcess for details.
eps	An arbitrary real number, typically small. Iterations are halted if the directional derivative of $\boldsymbol{\eta} \to L(\boldsymbol{\eta})$ in the direction of the new candidate is $\leq \varepsilon$.
T1	Maximal number of iterations to perform.
robustif	robustif = TRUE performs the robustification and Hermite interpolation procedure detailed in Rufibach (2006, 2007), robustif = FALSE does not. In the latter case, convergence of the algorithm is no longer guaranteed.
print	print = TRUE outputs log-likelihood in every loop, print = FALSE does not. Make sure to tell R to output (press CTRL+W).

Value

x	Vector of observations $x_1, \ldots, x_m$ that was used to estimate the density.
w	The vector of weights that had been used. Depends on the chosen setting for xgrid.
f	Vector with entries $\widehat{f}_m(x_i)$.
xn	Vector with initial observations $x_1, \ldots, x_n$.
Loglik	The value $L(\widehat{\phi}_m)$ of the log-likelihood-function L at the maximum $\widehat{\phi}_m$.
Iterations	Number of iterations performed.
sig	The standard deviation of the initial sample $x_1, \ldots, x_n$.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Rufibach K. (2006) *Log-concave Density Estimation and Bump Hunting for i.i.d. Observations.* PhD Thesis, University of Bern, Switzerland and Georg-August University of Goettingen, Germany, 2006.
Available at https://slsp-ube.primo.exlibrisgroup.com/permalink/41SLSP_UBE/17e6d97/alma99116730175505511.

Rufibach, K. (2007) Computing maximum likelihood estimators of a log-concave density function. *J. Stat. Comput. Simul.* **77**, 561–574.

See Also

icmaLogCon can be used to estimate a log-concave density. However, to generate an object of class dlc that allows application of summary and plot one has to use logConDens.

The following functions are used by icmaLogCon:

phieta, etaphi, Lhat_eta, quadDeriv, robust, isoMean.

Examples

```
set.seed(1977)
x <- rgamma(200, 2, 1)
## Not run:
res <- icmaLogCon(x, T1 = 2000, robustif = TRUE, print = TRUE)

## plot resulting functions
par(mfrow = c(2, 1), mar = c(3, 2, 1, 2))
plot(x, exp(res$phi), type = 'l'); rug(x)
plot(x, res$phi, type = 'l'); rug(x)

## End(Not run)
```

`intECDF` *Computes the Integrated Empirical Distribution Function at Arbitrary Real Numbers in s*

Description

Computes the value of

$$\bar{I}(t) = \int_{x_1}^{t} \bar{F}(r) d\, r$$

where $\bar{F}$ is the empirical distribution function of $x_1, \ldots, x_m$, at all real numbers t in the vector $\boldsymbol{s}$. Note that t (so all elements in $\boldsymbol{s}$) must lie in $[x_1, x_m]$. The exact formula for $\bar{I}(t)$ is

$$\bar{I}(t) = \Big(\sum_{i=2}^{i_0} (x_i - x_{i-1}) \frac{i-1}{n} \Big) + (t - x_{i_0}) \frac{i_0 - 1}{n}$$

where $i_0 = \max_{i=1,\ldots,m} \{x_i \leq t\}$.

Usage

```
intECDF(s, x)
```

Arguments

`s`	Vector of real numbers in $[x_1, x_m]$ where $\bar{I}$ should be evaluated at.
`x`	Vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ of original observations.

Value

Vector of the same length as $\boldsymbol{s}$, containing the values of $\bar{I}$ at the elements of $\boldsymbol{s}$.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Rufibach K. (2006) *Log-concave Density Estimation and Bump Hunting for i.i.d. Observations.* PhD Thesis, University of Bern, Switzerland and Georg-August University of Goettingen, Germany, 2006.
Available at https://slsp-ube.primo.exlibrisgroup.com/permalink/41SLSP_UBE/17e6d97/alma99116730175505511.

See Also

This function together with `intF` can be used to check the characterization of the log-concave density estimator in terms of distribution functions, see Rufibach (2006) and Duembgen and Rufibach (2009).

Examples

```
# for an example see the function intF.
```

`intF` *Computes the Integral of the estimated CDF at Arbitrary Real Numbers in s*

Description

Based on an object of class `dlc` as output by the function `logConDens`, this function gives values of

$$\widehat{I}(t) = \int_{x_1}^{t} \widehat{F}(r)dr$$

at all numbers in $\boldsymbol{s}$. Note that t (so all elements in $\boldsymbol{s}$) must lie in $[x_1, x_m]$. The exact formula for $\widehat{I}(t)$ is

$$\widehat{I}(t) = \Big(\sum_{i=1}^{i_0} \widehat{I}_i(x_{i+1})\Big) + \widehat{I}_{i_0}(t)$$

where $i_0 = \min\{m-1\,,\ \{i\ :\ x_i \le t\}\}$ and

$$I_j(x) = \int_{x_j}^{x} \widehat{F}(r)dr = (x-x_j)\widehat{F}(x_j) + \Delta x_{j+1}\Big(\frac{\Delta x_{j+1}}{\Delta\widehat{\phi}_{j+1}} J\Big(\widehat{\phi}_j, \widehat{\phi}_{j+1}, \frac{x-x_j}{\Delta x_{j+1}}\Big) - \frac{\widehat{f}(x_j)(x-x_j)}{\Delta\widehat{\phi}_{j+1}}\Big)$$

for $x \in [x_j, x_{j+1}]$, $j = 1, \ldots, m-1$, $\Delta v_{i+1} = v_{i+1} - v_i$ for any vector $\boldsymbol{v}$ and the function J introduced in `Jfunctions`.

Usage

```
intF(s, res)
```

Arguments

`s`	Vector of real numbers where the functions should be evaluated at.
`res`	An object of class `"dlc"`, usually a result of a call to `logConDens`.

Value

Vector of the same length as $\boldsymbol{s}$, containing the values of $\widehat{I}$ at the elements of $\boldsymbol{s}$.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Rufibach K. (2006) *Log-concave Density Estimation and Bump Hunting for i.i.d. Observations.* PhD Thesis, University of Bern, Switzerland and Georg-August University of Goettingen, Germany, 2006.
Available at https://slsp-ube.primo.exlibrisgroup.com/permalink/41SLSP_UBE/17e6d97/alma99116730175505511.

See Also

This function uses the output of `activeSetLogCon`. The function `intECDF` is similar, but based on the empirical distribution function.

Examples

```
## estimate gamma density
set.seed(1977)
x <- rgamma(200, 2, 1)
res <- logConDens(x, smoothed = FALSE, print = FALSE)

## compute and plot the process D(t) in Duembgen and Rufibach (2009)
s <- seq(min(res$x), max(res$x), by = 10 ^ -3)
D1 <- intF(s, res)
D2 <- intECDF(s, res$xn)
par(mfrow = c(2, 1))
plot(res$x, res$phi, type = 'l'); rug(res$x)
plot(s, D1 - D2, type = 'l'); abline(h = 0, lty = 2)
```

`isoMean` *Pool-Adjacent Violaters Algorithm: Least Square Fit under Monotonicity Constraint*

Description

Fits a vector $\widehat{\boldsymbol{g}}$ with nondecreasing components to the data vector $\boldsymbol{y}$ such that

$$\sum_{i=1}^{n}(y_i - \widehat{g}_i)^2$$

is minimal (pool - adjacent - violators algorithm). In case a weight vector with positive entries (and the same size as $\boldsymbol{y}$) is provided, the function produces an isotonic vector minimizing

$$\sum_{i=1}^{n} w_i(y_i - \widehat{g}_i)^2.$$

Usage

```
isoMean(y, w)
```

Arguments

`y`	Vector $(y_1, \ldots, y_n)$ of data points.
`w`	Arbitrary vector $(w_1, \ldots, w_n)$ of weights.

Value

Returns vector $\widehat{\boldsymbol{g}}$.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

Examples

```
## simple regression model
n <- 50
x <- sort(runif(n, 0, 1))
y <- x ^ 2 + rnorm(n, 0, 0.2)
s <- seq(0, 1, by = 0.01)
plot(s, s ^ 2, col = 2, type = 'l', xlim = range(c(0, 1, x)),
    ylim = range(c(0, 1 , y))); rug(x)

## plot pava result
lines(x, isoMean(y, rep(1 / n, n)), type = 's')
```

`Jfunctions` *Numerical Routine J and Some Derivatives*

Description

J00 represents the function $J(x, y, v)$, where for real numbers x, y and $v \in [0, 1]$,

$$J(x, y, v) = \int_0^v \exp((1-t)x + ty)dt = \frac{\exp(x + v(y-x)) - \exp(x)}{y - x}.$$

The functions Jab give the respective derivatives J_{ab} for $v = 1$, i.e.

$$J_{ab}(x, y) = \frac{\partial^{a+b}}{\partial x^a \partial y^b} J(x, y).$$

Specifically,

$$J_{10}(x, y) = \frac{\exp(y) - \exp(x) - (y-x)\exp(x)}{(y-x)^2};$$

$$J_{11}(x, y) = \frac{(y-x)(\exp(x) + \exp(y)) + 2(\exp(y) - \exp(x))}{(y-x)^3};$$

$$J_{20}(x, y) = 2\frac{\exp(y) - \exp(x) - (y-x)\exp(x) - (y-x)^2\exp(x)}{(y-x)^3}.$$

Usage

```
J00(x, y, v)
J10(x, y)
J11(x, y)
J20(x, y)
```

Arguments

x	Vector of length d with real entries.
y	Vector of length d with real entries.
v	Number in $[0, 1]^d$.

Value

Value of the respective function.

Note

Taylor approximations are used if $y - x$ is small. We refer to Duembgen et al (2011, Section 6) for details.

These functions are not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L, Huesler, A. and Rufibach, K. (2010) Active set and EM algorithms for log-concave densities based on complete and censored data. Technical report 61, IMSV, Univ. of Bern, available at https://arxiv.org/abs/0707.4643.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Lhat_eta *Value of the Log-Likelihood Function L, where Input is in Eta-Parametrization*

Description

Gives the value of

$$L(\phi) = \sum_{i=1}^{m} w_i \phi(x_i) - \int_{x_1}^{x_m} \exp(\phi(t))dt$$

where ϕ is parametrized via

$$\boldsymbol{\eta}(\boldsymbol{\phi}) = \Big(\phi_1, \Big(\eta_1 + \sum_{j=2}^{i}(x_i - x_{i-1})\eta_i\Big)_{i=2}^{m}\Big).$$

Usage

```
Lhat_eta(x, w, eta)
```

Arguments

x	Vector of independent and identically distributed numbers, with strictly increasing entries.
w	Optional vector of nonnegative weights corresponding to $\boldsymbol{x}_m$.
eta	Some vector $\boldsymbol{\eta}$ of the same length as $\boldsymbol{x}$ and $\boldsymbol{w}$.

Value

Value $L(\boldsymbol{\phi}) = L(\boldsymbol{\phi}(\boldsymbol{\eta}))$ of the log-likelihood function is returned.

Note

This function is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

`Local_LL` *Value of the Log-Likelihood Function L, where Input is in Phi-Parametrization*

Description

Gives the value of

$$L(\phi) = \sum_{i=1}^{m} w_i \phi(x_i) - \int_{x_1}^{x_m} \exp(\phi(t))dt.$$

Usage

```
Local_LL(x, w, phi)
```

Arguments

`x`	Vector of independent and identically distributed numbers, with strictly increasing entries.
`w`	Optional vector of nonnegative weights corresponding to $\boldsymbol{x}_m$.
`phi`	Some vector $\boldsymbol{\phi}$ of the same length as $\boldsymbol{x}$ and $\boldsymbol{w}$.

Value

Value $L = L(\boldsymbol{\phi})$ of the log-likelihood function is returned.

Note

This function is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

`Local_LL_all` *Log-likelihood, New Candidate and Directional Derivative for L*

Description

Computes the value of the log-likelihood function

$$L(\phi) = \sum_{i=1}^{m} w_i \phi(x_i) - \int_{x_1}^{x_m} \exp(\phi(t))dt,$$

a new candidate for ϕ via the Newton method as well as the directional derivative of $\boldsymbol{\phi} \to L(\boldsymbol{\phi})$ into that direction.

Usage

```
Local_LL_all(x, w, phi)
```

Arguments

`x`	Vector of independent and identically distributed numbers, with strictly increasing entries.
`w`	Optional vector of nonnegative weights corresponding to $\boldsymbol{x}_m$.
`phi`	Some vector $\boldsymbol{\phi}$ of the same length as $\boldsymbol{x}$ and $\boldsymbol{w}$.

Value

`ll`	Value $L(\phi)$ of the log-likelihood function at ϕ.
`phi_new`	New candidate for ϕ via the Newton-method, using the complete Hessian matrix.
`dirderiv`	Directional derivative of $\phi \to L(\phi)$ into the direction ϕ_{new}.

Note

This function is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

logConCI *Compute pointwise confidence interval for a density assuming log-concavity*

Description

Compute approximate confidence interval for the true log-concave density, on a grid of points. Two main approaches are implemented: In the first, the confidence interval at a fixed point is based on the pointwise asymptotic theory for the log-concave maximum likelihood estimator (MLE) developed in Balabdaoui, Rufibach, and Wellner (2009). In the second, the confidence interval is estimated via the boostrap.

Usage

```
logConCI(res, xx0, conf.level = c(0.8, 0.9, 0.95, 0.99)[3],
    type = c("DR", "ks", "nrd", "ECDFboot", "NPMLboot")[2],
    htype = c("hscv", "hlscv", "hpi", "hns")[4], BB = 500)
```

Arguments

res	An object of class dlc, usually a result of a call to logConDens.
xx0	Vector of grid points at which to calculate the confidence interval.
conf.level	Confidence level for the confidence interval(s). The default is 95%.
type	Vector of strings indicating type of confidence interval to compute. When type = ks is chosen, then htype should also be specified. The default is type = ks.
htype	Vector of strings indicating bandwidth selection method if type = ks. The default is htype = hns.
BB	number of iterations in the bootstrap if type = NPMLboot or type = ECDFboot. The default is BB = 500.

Details

In Balabdaoui et al. (2009) it is shown that (if the true density is strictly log-concave) the limiting distribution of the MLE of a log-concave density $\widehat{f}_n$ at a point x is

$$n^{2/5}(\widehat{f}_n(x) - f(x)) \to c_2(x)\bar{C}(0).$$

The nuisance parameter $c_2(x)$ depends on the true density f and the second derivative of its logarithm. The limiting process $\bar{C}(0)$ is found as the second derivative at zero of a particular operator (called the "envelope") of an integrated Brownian motion plus t^4.

Three of the confidence intervals are based on inverting the above limit using estimated quantiles of $\bar{C}(0)$, and estimating the nuisance parameter $c_2(x)$. The options for the function logConCI provide different ways to estimate this nuisance parameter. If type = "DR", $c_2(x)$ is estimated using derivatives of the smoothed MLE as calculated by the function logConDens (this method does not perform well in simulations and is therefore not recommended). If type="ks", $c_2(x)$ is estimated

using kernel density estimates of the true density and its first and second derivatives. This is done using the R package **ks**, and, with this option, a bandwidth selection method htype must also be chosen. The choices in htype correspond to the various options for bandwidth selection available in **ks**. If type = "nrd", the second derivative of the logarithm of the true density in $c_2(x)$ is estimated assuming a normal reference distribution.

Two of the confidence intervals are based on the bootstrap. For type = "ECDFboot" confidence intervals based on re-sampling from the empirical cumulative distribution function are computed. For type = "NPMLboot" confidence intervals based on re-sampling from the nonparametric maximum likelihood estimate of log-concave density are computed. Bootstrap confidence intervals take a few minutes to compute!

The default option is type = "ks" with htype = "hns". Currently available confidence levels are 80%, 90%, 95% and 99%, with a default of 95%.

Azadbakhsh et al. (2014) provides an empirical study of the relative performance of the various approaches available in this function.

Value

The function returns a list containing the following elements:

fhat	MLE evaluated at grid points.
up_DR	Upper confidence interval limit when type = DR.
lo_DR	Lower confidence interval limit when type = DR.
up_ks_hscv	Upper confidence interval limit when type = ks and htype = hscv.
lo_ks_hscv	Lower confidence interval limit when type = ks and htype = hscv.
up_ks_hlscv	Upper confidence interval limit when type = ks and htype = hlscv.
lo_ks_hlscv	Lower confidence interval limit when type = ks and htype = hlscv.
up_ks_hpi	Upper confidence interval limit when type = ks and htype = hpi.
lo_ks_hpi	Lower confidence interval limit when type = ks and htype = hpi.
up_ks_hns	Upper confidence interval limit when type = ks and htype = hns.
lo_ks_hns	Lower confidence interval limit when type = ks and htype = hns.
up_nrd	Upper confidence interval limit when type = nrd.
lo_nrd	Lower confidence interval limit when type = nrd.
up_npml	Upper confidence interval limit when type = NPMLboot.
lo_npml	Lower confidence interval limit when boot = NPMLboot.
up_ecdf	Upper confidence interval limit when boot = ECDFboot.
lo_ecdf	Lower confidence interval limit when boot = ECDFboot.

Author(s)

Mahdis Azadbakhsh

Hanna Jankowski, <hkj@yorku.ca>

References

Azadbakhsh, M., Jankowski, H. and Gao, X. (2014). Computing confidence intervals for log-concave densities. *Comput. Statist. Data Anal.*, **75**, 248–264.

Baladbaoui, F., Rufibach, K. and Wellner, J. (2009) Limit distribution theory for maximum likelihood estimation of a log-concave density. *Ann. Statist.*, **37(3)**, 1299–1331.

Tarn Duong (2012). ks: Kernel smoothing. R package version 1.8.10. https://CRAN.R-project.org/package=ks

Examples

```
## Not run:
## =====================================================
## Confidence intervals at a fixed point for the density
## =====================================================
data(reliability)
x.rel <- sort(reliability)

# calculate 95
grid <- seq(min(x.rel), max(x.rel), length.out = 200)
res <- logConDens(x.rel)
ci  <- logConCI(res, grid, type = c("nrd", "ECDFboot"))

par(las = 1, mar = c(2.5, 3.5, 0.5, 0.5))
hist(x.rel, n = 25, col = gray(0.9), main = "", freq = FALSE,
    xlab = "", ylab = "", ylim = c(0, 0.0065), border = gray(0.5))
lines(grid, ci$fhat, col = "black", lwd = 2)
lines(grid, ci$lo_nrd, col = "red", lwd = 2, lty = 2)
lines(grid, ci$up_nrd, col = "red", lwd = 2, lty = 2)
lines(grid, ci$lo_ecdf, col = "blue", lwd = 2, lty = 2)
lines(grid, ci$up_ecdf, col = "blue", lwd = 2, lty = 2)
legend("topleft", col = c("black", "blue", "red"), lwd = 2, lty = c(1, 2, 2), legend =
c("log-concave NPMLE", "CI for type = nrd", "CI for type = ECDFboot"), bty = "n")

## End(Not run)
```

---

logConCIfunctions *Functions that are used by logConCI*

Description

Functions that are used by logConCI and are not intended to be called by the user.

Author(s)

Mahdis Azadbakhsh

Hanna Jankowski, <hkj@yorku.ca>

References

Azadbakhsh, M., Jankowski, H. and Gao, X. (2014). Computing confidence intervals for log-concave densities. *Comput. Statist. Data Anal.*, **75**, 248–264.

Baladbaoui, F., Rufibach, K. and Wellner, J. (2009). Limit distribution theory for maximum likelihood estimation of a log-concave density. *Ann. Statist.*, **37(3)**, 1299–1331.

Tarn Duong (2012). ks: Kernel smoothing. R package version 1.8.10. https://CRAN.R-project.org/package=ks

---

`logConDens` *Compute log-concave density estimator and related quantities*

Description

Compute the log-concave and smoothed log-concave density estimator.

Usage

```
logConDens(x, xgrid = NULL, smoothed = TRUE, print = FALSE,
    gam = NULL, xs = NULL)
```

Arguments

`x`	Vector of independent and identically distributed numbers, not necessarily unique.
`xgrid`	Governs the generation of weights for observations. See `preProcess` for details.
`smoothed`	If `TRUE`, the smoothed version of the log-concave density estimator is also computed.
`print`	`print = TRUE` outputs the log-likelihood in every loop, `print = FALSE` does not. Make sure to tell R to output (press CTRL+W).
`gam`	Only necessary if `smoothed = TRUE`. The standard deviation of the normal kernel. If equal to `NULL`, `gam` is chosen such that the variances of the original sample $x_1, \ldots, x_n$ and $\widehat{f}_n^*$ coincide.
`xs`	Only necessary if `smoothed = TRUE`. Either provide a vector of support points where the smoothed estimator should be computed at, or leave as `NULL`. Then, a sufficiently width equidistant grid of points will be used.

Details

See `activeSetLogCon` for details on the computations.

Value

logConDens returns an object of class "dlc", a list containing the following components: xn, x, w, phi, IsKnot, L, Fhat, H, n, m, knots, mode, and sig as generated by activeSetLogCon. If smoothed = TRUE, then the returned object additionally contains f.smoothed, F.smoothed, gam, and xs as generated by evaluateLogConDens. Finally, the entry smoothed of type "logical" returnes the value of smoothed.

The methods summary.dlc and plot.dlc are used to obtain a summary and generate plots of the estimated density.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L, Huesler, A. and Rufibach, K. (2010). Active set and EM algorithms for log-concave densities based on complete and censored data. Technical report 61, IMSV, Univ. of Bern, available at https://arxiv.org/abs/0707.4643.

Duembgen, L. and Rufibach, K. (2009). Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011). logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Examples

```
## ====================================================
## Illustrate on simulated data
## ====================================================

## Set parameters
n <- 50
x <- rnorm(n)

res <- logConDens(x, smoothed = TRUE, print = FALSE, gam = NULL,
    xs = NULL)
summary(res)
plot(res, which = "density", legend.pos = "topright")
plot(res, which = "log-density")
plot(res, which = "CDF")

## Compute slopes and intercepts of the linear functions that
## compose phi
slopes <- diff(res$phi) / diff(res$x)
intercepts <- -slopes * res$x[-n] + res$phi[-n]
```

```
## ================================================
## Illustrate method on reliability data
## Reproduce Fig. 2 in Duembgen & Rufibach (2009)
## ================================================

## Set parameters
data(reliability)
x <- reliability
n <- length(x)
res <- logConDens(x, smooth = TRUE, print = TRUE)
phi <- res$phi
f <- exp(phi)

## smoothed log-concave PDF
f.smoothed <- res$f.smoothed
xs <- res$xs

## compute kernel density
sig <- sd(x)
h <- sig / sqrt(n)
f.kernel <- rep(NA, length(xs))
for (i in 1:length(xs)){
    xi <- xs[i]
    f.kernel[i] <- mean(dnorm(xi, mean = x, sd = h))
}

## compute normal density
mu <- mean(x)
f.normal <- dnorm(xs, mean = mu, sd = sig)

## ================================================
## Plot resulting densities, i.e. reproduce Fig. 2
## in Duembgen and Rufibach (2009)
## ================================================
plot(0, 0, type = 'n', xlim = range(xs), ylim = c(0, 6.5 * 10^-3))
rug(res$x)
lines(res$x, f, col = 2)
lines(xs, f.normal, col = 3)
lines(xs, f.kernel, col = 4)
lines(xs, f.smoothed, lwd = 3, col = 5)
legend("topleft", c("log-concave", "normal", "kernel",
    "log-concave smoothed"), lty = 1, col = 2:5, bty = "n")


## ================================================
## Plot log-densities
## ================================================
plot(0, 0, type = 'n', xlim = range(xs), ylim = c(-20, -5))
legend("bottomright", c("log-concave", "normal", "kernel",
    "log-concave smoothed"), lty = 1, col = 2:5, bty = "n")
rug(res$x)
lines(res$x, phi, col = 2)
lines(xs, log(f.normal), col = 3)
```

```
lines(xs, log(f.kernel), col = 4)
lines(xs, log(f.smoothed), lwd = 3, col = 5)


## ================================================
## Confidence intervals at a fixed point for the density
## see help file for logConCI()
## ================================================
```

`logConROC` *Compute ROC curve based on log-concave estimates for the constituent distributions*

Description

The receiver operating characteristic (ROC) curve for two constituent distributions F and G is defined as

$$R(t; F, G) = 1 - G(F^{-1}(1 - t))$$

for $t \in [0, 1]$. It is typically used to assess the performance of a diagnostic test used to discriminate between healthy and diseased individuals based on a continuous variable.

Usage

```
logConROC(cases, controls, grid, smooth = TRUE)
```

Arguments

`cases`	A vector of measurements for the cases.
`controls`	A vector of measurements for the controls.
`grid`	A vector specifying the grid where the ROC curve is computed on.
`smooth`	Logical, indicating whether ROC curve and AUC should also be computed based on the smoothed log-concave density estimator.

Details

In Rufibach (2011) it was shown that the ROC curve based on log-concave density estimates exhibit nice properties for finite sample sizes as well as asymptotically. Its performance is typically much better than that of the empirical ROC curve and only, if at all, sligthly worse compared to the binormal model when in fact the underlying densities are normal. However, log-concavity encompasses many parametric densities, so this new model is much more flexible than the binormal one, at little efficiency sacrifice.

Value

m	Number of control measurements.
n	Number of case measurements.
fROC	Estimated ROC curve based on the log-concave density estimate.
fROC.smooth	Estimated ROC curve based on the smoothed log-concave density estimate.
res0	dlc object as a result of a call to logConDens for the data of the controls.
res1	dlc object as a result of a call to logConDens for the data of the cases.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

References

Duembgen, L. and Rufibach, K. (2009). Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011). logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Rufibach, K. (2012). A smooth ROC curve estimator based on log-concave density estimates. *Int. J. Biostat.*, **8**(1), 1–29.

See Also

Confidence intervals at given false-positive fractions for the ROC curve based on log-concave densities can be computed using confIntBootLogConROC_t0. For the computation of the AUC the function ROCx is used. In the example below we analyze the pancreas data.

Examples

```
## ROC curve for pancreas data
data(pancreas)
status <- factor(pancreas[, "status"], levels = 0:1,
    labels = c("healthy", "diseased"))
var <- log(pancreas[, "ca199"])
cases <- var[status == "diseased"]
controls <- var[status == "healthy"]

## compute and plot empirical ROC curve
## code modified from https://stat.ethz.ch/pipermail/r-help/2008-October/178531.html
xx <- c(-Inf, sort(unique(c(cases, controls))), Inf)
sens <- sapply(xx, function(x){mean(cases >= x)})
spec <- sapply(xx, function(x){mean(controls < x)})

## compute log-concave ROC curve
grid <- seq(0, 1, by = 1 / 500)
roc.logcon <- logConROC(cases, controls, grid)
```

```
## plot
plot(0, 0, xlim = c(0, 1), ylim = c(0, 1), type = 'l',
    main = "ROC curves for pancreas data", xlab = "1 - specificity",
    ylab = "sensitivity", pty = 's')
legend("bottomright", c("empirical ROC", "log-concave ROC", "smooth log-concave ROC"),
    lty = c(1, 1, 2), lwd = 2, col = 2:4, bty = "n")
segments(0, 0, 1, 1, col = 1)
lines(1 - spec, sens, type = 'l', col = 2, lwd = 2)
lines(grid, roc.logcon$fROC, col = 3, lwd = 2)
lines(grid, roc.logcon$fROC.smooth, col = 4, lwd = 2, lty = 2)

## Not run:
## bootstrap confidence intervals at 1 - specificity = 0.2 and 0.8:
res <- confIntBootLogConROC_t0(controls, cases, grid = c(0.2, 0.8), conf.level = 0.95,
    M = 1000, smooth = TRUE, output = TRUE)
res

## End(Not run)
```

`logconTwoSample` *Compute p-values for two-sample test based on log-concave CDF estimates*

Description

Compute p-values for a test for the null hypothesis of equal CDFs of two samples. The test statistic is reminiscient of Kolmogorv-Smirnov's, but instead of computing it for the empirical CDFs, this function computes it based on log-concave estimates for the CDFs.

Usage

```
logconTwoSample(x, y, which = c("MLE", "smooth"), M = 999,
    n.grid = 500, display = TRUE, seed0 = 1977)
```

Arguments

`x`	First data sample.
`y`	Second data sample.
`which`	Indicate for which type of estimate the test statistic should be computed.
`M`	Number of permutations.
`n.grid`	Number of grid points in computation of maximal difference between smoothed log-concave CDFs. See `maxDiffCDF` for details.
`display`	If `TRUE` progress of computations is shown.
`seed0`	Set seed to reproduce results.

Details

Given two i.i.d. samples $x_1, \ldots, x_{n_1}$ and $y_1, \ldots, y_{n_2}$ this function computes a permutation test p-value that provides evidence against the null hypothesis

$$H_0 : F_1 = F_2$$

where F_1, F_2 are the CDFs of the samples, respectively. A test either based on the log-concave MLE or on its smoothed version (see Duembgen and Rufibach, 2009, Section 3) are provided. Note that computation of the smoothed version takes considerably more time.

Value

`p.value`	A two dimensional vector containing the p-values.
`test.stat.orig`	The test statistics for the original samples.
`test.stats`	A $M \times 2$ matrix containing the test statistics for all the permutations.

Warning

Note that the algorithm that finds the maximal difference for the smoothed estimate is of approximative nature only. It may fail for very large sample sizes.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Examples

```
## Not run:
n1 <- 30
n2 <- 25
x <- rgamma(n1, 2, 1)
y <- rgamma(n2, 2, 1) + 1
twosample <- logconTwoSample(x, y, which = c("MLE", "smooth")[1], M = 999)

## End(Not run)
```

`maxDiffCDF` *Compute maximal difference between CDFs corresponding to log-concave estimates*

Description

Compute the maximal difference between two estimated log-concave distribution functions, either the MLEs or the smoothed versions. This function is used to set up a two-sample permutation test that assesses the null hypothesis of equality of distribution functions.

Usage

```
maxDiffCDF(res1, res2, which = c("MLE", "smooth"), n.grid = 500)
```

Arguments

`res1`	An object of class `"dlc"`, usually a result of a call to `logConDens` for the first sample.
`res2`	An object of class `"dlc"`, usually a result of a call to `logConDens` for the second sample.
`which`	Indicate for which type of estimate the maximal difference should be computed.
`n.grid`	Number of grid points used to find zeros of $\hat{f}^*_{n_1} - \hat{f}^*_{n_2}$ for the smooth estimate.

Details

Given two i.i.d. samples $x_1, \ldots, x_{n_1}$ and $y_1, \ldots, y_{n_2}$ this function computes the maxima of

$$D_1(t) = \hat{F}_{n_1}(t) - \hat{F}_{n_2}(t)$$

and

$$D_2(t) = \hat{F}^*_{n_1}(t) - \hat{F}^*_{n_2}(t).$$

Value

`test.stat`	A two-dimensional vector containing the above maxima.
`location`	A two-dimensional vector where the maxima occur.

Warning

Note that the algorithm that finds the maximal difference for the smoothed estimate is of approximative nature only. It may fail for very large sample sizes.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Examples

```
n1 <- 100
n2 <- 120
x <- sort(rgamma(n1, 2, 1))
y <- sort(rgamma(n2, 2, 1))
res1 <- logConDens(x, smoothed = TRUE)
res2 <- logConDens(y, smoothed = TRUE)
d <- maxDiffCDF(res1, res2, n.grid = 200)

## log-concave estimate
xs <- seq(min(res1$xs, res2$xs), max(res1$xs, res2$xs), length = 200)
F1 <- matrix(NA, nrow = length(xs), ncol = 1); F2 <- F1
for (i in 1:length(xs)){
    F1[i] <- evaluateLogConDens(xs[i], res1, which = 3)[, "CDF"]
    F2[i] <- evaluateLogConDens(xs[i], res2, which = 3)[, "CDF"]
    }
par(mfrow = c(1, 2))
plot(xs, abs(F1 - F2), type = "l")
abline(v = d$location[1], lty = 2, col = 3)
abline(h = d$test.stat[1], lty = 2, col = 3)

## smooth estimate
xs <- seq(min(res1$xs, res2$xs), max(res1$xs, res2$xs), length = 200)
F1smooth <- matrix(NA, nrow = length(xs), ncol = 2); F2smooth <- F1smooth
for (i in 1:length(xs)){
    F1smooth[i, ] <- evaluateLogConDens(xs[i], res1, which = 4:5)[,
        c("smooth.density", "smooth.CDF")]
    F2smooth[i, ] <- evaluateLogConDens(xs[i], res2, which = 4:5)[,
        c("smooth.density", "smooth.CDF")]
    }
plot(xs, abs(F1smooth[, 2] - F2smooth[, 2]), type = "l")
abline(h = 0)
abline(v = d$location[2], lty = 2, col = c(3, 4))
abline(h = d$test.stat[2], lty = 2, col = c(3, 4))
```

MLE *Unconstrained piecewise linear MLE*

Description

Given a vector of observations $\boldsymbol{x} = (x_1, \ldots, x_m)$ with pairwise distinct entries and a vector of weights $\boldsymbol{w} = (w_1, \ldots, w_m)$ s.t. $\sum_{i=1}^m w_i = 1$, this function computes a function $\widehat{\phi}_{MLE}$ (represented by the vector $(\widehat{\phi}_{MLE}(x_i))_{i=1}^m$) supported by $[x_1, x_m]$ such that

$$L(\phi) = \sum_{i=1}^{m} w_i \phi(x_i) - \sum_{j=1}^{m-1} (x_{j+1} - x_j) J(\phi_j, \phi_{j+1})$$

is maximal over all continuous, piecewise linear functions with knots in $\{x_1, \ldots, x_m\}$

Usage

```
MLE(x, w = NA, phi_o = NA, prec = 1e-7, print = FALSE)
```

Arguments

x	Vector of independent and identically distributed numbers, with strictly increasing entries.
w	Optional vector of nonnegative weights corresponding to $\boldsymbol{x}_m$.
phi_o	Optional starting vector.
prec	Threshold for the directional derivative during the Newton-Raphson procedure.
print	print = TRUE outputs log-likelihood in every loop, print = FALSE does not. Make sure to tell R to output (press CTRL+W).

Value

phi	Resulting column vector $(\widehat{\phi}_{MLE}(x_i))_{i=1}^m$.
L	Value $L(\widehat{\phi}_{MLE})$ of the log-likelihood at $\widehat{\phi}_{MLE}$.
Fhat	Vector of the same length as $\boldsymbol{x}$ with entries $\widehat{F}_{MLE,1} = 0$ and $$\widehat{F}_{MLE,k} = \sum_{j=1}^{k-1} (x_{j+1} - x_j) J(\phi_j, \phi_{j+1})$$ for $k \geq 2$.

Note

This function is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

---

`pancreas` *Data from pancreatic cancer serum biomarker study*

Description

First published by Wieand et al (1989), this dataset contains data on serum measurements for a cancer antigen (CA-125) and a carbohydrate antigen (CA19.9) both of which are measured on a continuous non-negative scale. The measurements were taken within a case-control study on 90 cases with pancreatic cancer and 51 controls who did not have cancer but pancreatitis. The primary question of the study was which one of the two biomarkers best distinguishes cases from controls.

Usage

```
data(pancreas)
```

Format

A data frame with 141 observations on the following 3 variables.

`ca199` CA19.9 measurements.

`ca125` CA125 measurements.

`status` Patient status, 0 for controls and 1 for cases.

Source

The data was downloaded from http://labs.fhcrc.org/pepe/book/ on February 2, 2011.

References

Wieand, S., Gail, M. H., James, B. R., and James, K.L. (1989). A family of nonparametric statistics for comparing diagnostic markers with paired or unpaired data. *Biometrika*, **76**, 585–592.

Pepe, M.S. (2003) *The statistical evaluation of medical tests for classification and prediction*. Oxford: Oxford University Press (Section 1.3.3).

See Also

This data is analyzed in the help file for the function `logConROC`.

plot.dlc *Standard plots for a dlc object*

Description

plot method for class "dlc". Three plots (selectable by which) are currently available: a plot of the estimated density, the estimated log-density, or the distribution function corresponding to the estimated log-concave density. By default, a plot of the density estimate is provided. If smoothed = TRUE, the smoothed version of the log-concave density estimate (saved in x) is added to the density and log-density plot. For the CDF, the smoothed version is not contained by default in a dlc object and needs to be computed when asked to be plotted.

Usage

```
## S3 method for class 'dlc'
plot(x, which = c("density", "log-density", "CDF"),
    add.title = TRUE, legend.pos = "topright", ...)
```

Arguments

x	An object of class "dlc", usually a result of a call to logConDens.
which	One of "density", "log-density", or "CDF".
add.title	Logical, if TRUE adds a standard title to the plot.
legend.pos	Placement of the legend. One of "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"; or "none" for not displaying a legend. See legend for details.
...	Further arguments.

Details

See activeSetLogCon and evaluateLogConDens for details on the computations.

Value

Chosen plot is generated.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L, Huesler, A. and Rufibach, K. (2010). Active set and EM algorithms for log-concave densities based on complete and censored data. Technical report 61, IMSV, Univ. of Bern, available at https://arxiv.org/abs/0707.4643.

Duembgen, L. and Rufibach, K. (2009). Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Examples

```
## See help file of function "logConDens".
```

`preProcess` *Compute a weighted sample from initial observations*

Description

Generates weights from initial sample.

Usage

```
preProcess(x, xgrid = NULL)
```

Arguments

`x`	Vector of independent and identically distributed numbers, not necessarily unique.
`xgrid`	Parameter that governs the generation of weights: If `xgrid = NULL` a new sample of unique observations is generated with corresponding vector of weights. If `xgrid` is a positive number, observations are binned in a grid with grid length `xgrid`. Finally, an entire vector specifying a user-defined grid can be supplied.

Value

`x`	Vector of unique and sorted observations deduced from the input `x` according to the specification given by `xgrid`.
`w`	Vector of corresponding weights, normalized to sum to one.
`sig`	Standard deviation of the inputed observations. This quantity is needed when computing the smoothed log-concave density estimator via `evaluateLogConDens`.
`n`	Number of initial observations.

Note

This function is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

---

`Q00` *Numerical Routine Q*

Description

This function is used in the computation of $\widehat{f}^*$ and $\widehat{F}^*$.

Usage

```
Q00(x, a, u, v, gamma, QFhat = FALSE)
```

Arguments

`x`	Number at which to compute q and/or Q.
`a`	Vector of length m with real entries.
`u`	Vector of length m with real entries.
`v`	Vector of length m with real entries.
`gamma`	Real number. Standard deviation to be used.
`QFhat`	Logical. Should Q be computed?

Value

The vector(s) q and/or Q.

Note

Taylor approximation is used if a is small. In addition, as described in Duembgen and Rufibach (2011) at the end of Appendix C, in extreme situations, e.g. when data sets contain extreme spacings, numerical problems may occur in the computation of the function q_γ (eq. (7) in Duembgen and Rufibach, 2011). For it may happen that the exponent is rather large while the difference of Gaussian CDFs is very small. To moderate these problems, we are using the following bounds:

$$\exp(-m^2/2)\big(\Phi(\delta) - \Phi(-\delta)\big) \;\le\; \Phi(b) - \Phi(a) \;\le\; \exp(-m^2/2)\cosh(m\delta)\big(\Phi(\delta) - \Phi(-\delta)\big)$$

for arbitrary numbers $a < b$ and $m := (a+b)/2$, $\delta := (b-a)/2$.

However, the function `Q00` is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. https://www.jstatsoft.org/v39/i06

`qloglin` *Quantile Function In a Simple Log-Linear model*

Description

Suppose the random variable X has density function

$$g_\theta(x) = \frac{\theta \exp(\theta x)}{\exp(\theta) - 1}$$

for an arbitrary real number θ and $x \in [0, 1]$. The function `qloglin` is simply the quantile function

$$G_\theta^{-1}(u) = \theta^{-1} \log\Big(1 + (e^\theta - 1)u\Big)$$

in this model, for $u \in [0, 1]$. This quantile function is used for the computation of quantiles of $\widehat{F}_m$ in `quantilesLogConDens`.

Usage

```
qloglin(u, t)
```

Arguments

`u`	Vector in $[0, 1]^d$ where quantiles are to be computed at.
`t`	Parameter θ.

Value

`z`	Vector containing the quantiles $G_n^{-1}(u_i)$ for $i = 1, \ldots, d$.

Note

Taylor approximation is used if θ is small.

This function is not intended to be called by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

---

`quadDeriv` *Gradient and Diagonal of Hesse Matrix of Quadratic Approximation to Log-Likelihood Function L*

Description

Computes gradient and diagonal of the Hesse matrix w.r.t. to η of a quadratic approximation to the reparametrized original log-likelihood function

$$L(\phi) = \sum_{i=1}^{m} w_i \phi(x_i) - \int_{-\infty}^{\infty} \exp(\phi(t))dt.$$

where L is parametrized via

$$\boldsymbol{\eta}(\boldsymbol{\phi}) = \left(\phi_1, \left(\eta_1 + \sum_{j=2}^{i}(x_i - x_{i-1})\eta_i\right)_{i=2}^{m}\right).$$

$\boldsymbol{\phi}$: vector $(\phi(x_i))_{i=1}^m$ representing concave, piecewise linear function ϕ,
$\boldsymbol{\eta}$: vector representing successive slopes of ϕ.

Usage

```
quadDeriv(dx, w, eta)
```

Arguments

`dx`	Vector $(0, x_i - x_{i-1})_{i=2}^m$.
`w`	Vector of weights as in `activeSetLogCon`.
`eta`	Vector $\boldsymbol{\eta}$.

Value

$m \times 2$ matrix. First column contains gradient and second column diagonal of Hesse matrix.

Note

This function is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

See Also

quadDeriv is used by the function icmaLogCon.

---

quantilesLogConDens *Function to compute Quantiles of Fhat*

Description

Function to compute p_0-quantile of

$$\widehat{F}_m(t) = \int_{x_1}^{t} \widehat{f}_m(t)dt,$$

where $\widehat{f}_m$ is the log-concave density estimator, typically computed via logConDens and p_0 runs through the vector ps. The formula to compute a quantile at $u \in [\widehat{F}_m(x_j), \widehat{F}_m(x_{j+1})]$ for $j = 1, \ldots, n-1$ is:

$$\widehat{F}_m^{-1}(u) = x_j + (x_{j+1} - x_j) G^{-1}_{(x_{j+1}-x_j)(\widehat{\phi}_{j+1}-\widehat{\phi}_j)}\Big(\frac{u - \widehat{F}_m(x_j)}{\widehat{F}_m(x_{j+1}) - \widehat{F}_m(x_j)}\Big),$$

where G_θ^{-1} is described in qloglin.

Usage

```
quantilesLogConDens(ps, res)
```

Arguments

ps	Vector of real numbers where quantiles should be computed.
res	An object of class "dlc", usually a result of a call to logConDens.

Value

Returns a data.frame with row $(p_{0,i}, q_{0,i})$ where $q_{0,i} = \inf_x\{\widehat{F}_m(x) \geq p_{0,i}\}$ and $p_{0,i}$ runs through ps.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

Examples

```
## estimate gamma density
set.seed(1977)
x <- rgamma(200, 2, 1)
res <- logConDens(x, smoothed = FALSE, print = FALSE)

## compute 0.95 quantile of Fhat
q <- quantilesLogConDens(0.95, res)[, "quantile"]
plot(res, which = "CDF", legend.pos = "none")
abline(h = 0.95, lty = 3); abline(v = q, lty = 3)
```

---

reliability *Reliability dataset used to illustrate log-concave density estimation*

Description

Dataset that contains the data analyzed in Duembgen and Rufibach (2009, Figure 2).

Usage

```
data(reliability)
```

Format

A vector containing the 786 observations analyzed in Duembgen and Rufibach (2009, Figure 2).

Source

The data was taken from Duembgen and Rufibach (2009).

References

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

See Also

See the example in logConDens for the analysis of this data.

`reparametrizations` *Changes Between Parametrizations*

Description

Given a vector $(\phi_1, \ldots, \phi_m)$ representing the values of a piecewise linear concave function at $x_1, \ldots, x_m$, `etaphi` returns a column vector with the entries

$$\boldsymbol{\eta} = \Big(\phi_1, \Big(\eta_1 + \sum_{j=2}^{m}(x_i - x_{i-1})\eta_i\Big)_{i=2}^{m}\Big).$$

The function `phieta` returns a vector with the entries

$$\boldsymbol{\phi} = \Big(\eta_1, \Big(\frac{\phi_i - \phi_{i-1}}{x_i - x_{i-1}}\Big)_{i=2}^{m}\Big).$$

Usage

```
etaphi(x, eta)
phieta(x, phi)
```

Arguments

`x`	Vector of independent and identically distributed numbers, with strictly increasing entries.
`eta`	Vector with entries $\eta_i = \eta(x_i)$.
`phi`	Vector with entries $\phi_i = \phi(x_i)$.

Note

These functions are not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

`rlogcon` *Generate random sample from the log-concave and the smoothed log-concave density estimator*

Description

Generate a random sample from a distribution with density $\hat{f}_n$ and $\hat{f}_n^*$, as described in Duembgen and Rufibach (2009, Section 3).

Usage

```
rlogcon(n, x0)
```

Arguments

`n`	Size of random sample to be generated.
`x0`	Sorted vector of independent and identically distributed numbers, not necessarily unique.

Value

`X`	Random sample from $\hat{f}_n$.
`X_star`	Random sample from $\hat{f}_n^*$.
`U`	Uniform random sample of size `n` used in the generation of `X`.
`Z`	Normal random sample of size `n` used in the generation of `X_star`.
`f`	Computed log-concave density estimator.
`f.smoothed`	List containing smoothed log-concave density estimator, as output by `evaluateLogConDens`.
`x`	Vector of distinct observations generated from `x0`.
`w`	Weights corresponding to `x`.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L. and Rufibach, K. (2009) Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Examples

```
## ================================================
## Generate random samples as described in Section 3 of
## Duembgen and Rufibach (2009)
## ================================================
x0 <- rnorm(111)
n <- 22
random <- rlogcon(n, x0)

## sample of size n from the log-concave density estimator
random$X

## sample of size n from the smoothed log-concave density estimator
random$X_star
```

`robust` *Robustification and Hermite Interpolation for ICMA*

Description

Performs robustification and Hermite interpolation in the iterative convex minorant algorithm as described in Rufibach (2006, 2007).

Usage

```
robust(x, w, eta, etanew, grad)
```

Arguments

`x`	Vector of independent and identically distributed numbers, with strictly increasing entries.
`w`	Optional vector of nonnegative weights corresponding to $\boldsymbol{x}_m$.
`eta`	Current candidate vector.
`etanew`	New candidate vector.
`grad`	Gradient of L at current candidate vector η.

Value

Returns a (possibly) new vector η on the segment

$$(1 - t_0)\eta + t_0\eta_{new}$$

such that the log-likelihood of this new η is strictly greater than that of the initial η and t_0 is chosen according to the Hermite interpolation procedure described in Rufibach (2006, 2007).

Note

This function is not intended to be invoked by the end user.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Rufibach K. (2006) *Log-concave Density Estimation and Bump Hunting for i.i.d. Observations.* PhD Thesis, University of Bern, Switzerland and Georg-August University of Goettingen, Germany, 2006.
Available at https://slsp-ube.primo.exlibrisgroup.com/permalink/41SLSP_UBE/17e6d97/alma99116730175505511.

Rufibach, K. (2007) Computing maximum likelihood estimators of a log-concave density function. *J. Stat. Comput. Simul.* **77**, 561–574.

ROCx — *Compute ROC curve at a given x based on log-concave estimates for the constituent distributions*

Description

Computes the value of the ROC curve at x (which may be a vector) based on log-concave density estimates of the constituent distributions.

Usage

```
ROCx(x, res0, res1, smooth = FALSE)
```

Arguments

x	Vector of numbers in $[0, 1]$ where the ROC curve should be computed at.
res0	dlc object as a result of a call to logConDens for the data of the controls.
res1	dlc object as a result of a call to logConDens for the data of the cases.
smooth	Logical. If TRUE kernel smoothed log-concave estimate is used.

Value

A real number or a vector of dimension the same as x, the value of the ROC curve at x.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

References

Rufibach, K. (2012). A smooth ROC curve estimator based on log-concave density estimates. *Int. J. Biostat.*, **8**(1), 1–29.

See Also

Used for the computation of AUC in `logConROC`.

`summary.dlc` *Summarizing log-concave density estimation*

Description

`summary` method for class `"dlc"`.

Usage

```
## S3 method for class 'dlc'
summary(object, ...)
```

Arguments

`object`	An object of class `"dlc"`, usually a result of a call to `logConDens`.
`...`	Further arguments.

Details

See `activeSetLogCon` and `evaluateLogConDens` for details on the computations.

Value

The function `summary.dlc` returns a list of summary statistics of the estimated log-concave density as well as of its smoothed version (depending on the value of `smoothed` when calling `logConDens`).

Warning

Note that the numbering of knots in the output relies on the vector of *unique* observations.

Author(s)

Kaspar Rufibach, <kaspar.rufibach@gmail.com>,
http://www.kasparrufibach.ch

Lutz Duembgen, <duembgen@stat.unibe.ch>,
https://www.imsv.unibe.ch/about_us/staff/prof_dr_duembgen_lutz/index_eng.html

References

Duembgen, L, Huesler, A. and Rufibach, K. (2010). Active set and EM algorithms for log-concave densities based on complete and censored data. Technical report 61, IMSV, Univ. of Bern, available at https://arxiv.org/abs/0707.4643.

Duembgen, L. and Rufibach, K. (2009). Maximum likelihood estimation of a log–concave density and its distribution function: basic properties and uniform consistency. *Bernoulli*, **15(1)**, 40–68.

Duembgen, L. and Rufibach, K. (2011) logcondens: Computations Related to Univariate Log-Concave Density Estimation. *Journal of Statistical Software*, **39(6)**, 1–28. doi:10.18637/jss.v039.i06

Examples

```
## See help file of function "logConDens".
```

Index