Package 'logcondiscr'

July 3, 2015

Type Package

Title Estimate a Log-Concave Probability Mass Function from Discrete i.i.d. Observations

Version 1.0.6

Date 2015-07-03

Author Kaspar Rufibach <kaspar.rufibach@gmail.com> and Fadoua Balabdaoui <fadoua@ceremade.dauphine.fr> and Hanna Jankowski <hkj@mathstat.yorku.ca> and Kathrin Weyermann <kathrin.weyermann@bkw-fmb.ch>

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

Depends Matrix, mvtnorm, cobs

Imports stats

Description Given independent and identically distributed observations X(1), ..., X(n), allows to compute the maximum likelihood estimator (MLE) of probability mass function (pmf) under the assumption that it is log-concave, see Weyermann (2007) and Balabdaoui, Jankowski, Rufibach, and Pavlides (2012). The main functions of the package are 'logConDiscrMLE' that allows computation of the log-concave MLE, 'logConDiscrCI' that computes pointwise confidence bands for the MLE, and 'kInflatedLogConDiscr' that computes a mixture of a log-concave PMF and a point mass at k.

License GPL (>= 2)

URL http://www.kasparrufibach.ch ,

http://www.ceremade.dauphine.fr/~fadoua ,

http://www.math.yorku.ca/~hkj

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-03 12:51:05

R topics documented:

`logcondiscr-package` *Estimate a Log-Concave Probability Mass Function from Discrete i.i.d. Observations*

Description

Implements the maximum likelihood estimator (MLE) for a probability mass function (PMF) under the assumption of log-concavity from i.i.d. data.

Details

Package:	logcondiscr
Type:	Package
Version:	1.0.6
Date:	2015-07-03
License:	GPL (>=2)
LazyLoad:	yes

The main functions in the package are:

`logConDiscrMLE`: Compute the maximum likelihood estimator (MLE) of a log-concave PMF from i.i.d. data. The constrained log-likelihood function is maximized using an active set algorithm as initially described in Weyermann (2007).

`logConDiscrCI`: Compute the maximum likelihood estimator (MLE) of a log-concave PMF from i.i.d. data and corresponding, asymptotically valid, pointwise confidence bands as developed in Balabdaoui et al (2012).

`kInflatedLogConDiscr`: Compute an estimate of a mixture of a log-concave PMF that is inflated at k, from i.i.d. data, using an EM algorithm.

Author(s)

Kaspar Rufibach (maintainer) <kaspar.rufibach@gmail.com>
http://www.kasparrufibach.ch
Fadoua Balabdaoui <fadoua@ceremade.dauphine.fr>
http://www.ceremade.dauphine.fr/~fadoua
Hanna Jankowski <hkj@mathstat.yorku.ca>
http://www.math.yorku.ca/~hkj
Kathrin Weyermann

References

Balabdaoui, F., Jankowski, H., Rufibach, K., and Pavlides, M. (2013). Maximum likelihood estimation and confidence bands for a discrete log-concave distribution. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **75**(4), 769–790.

Weyermann, K. (2007). An Active Set Algorithm for Log-Concave Discrete Distributions. *MSc thesis, University of Bern* (Supervisor: Lutz Duembgen).

See Also

Functions to estimate the log-concave MLE for a univariate continuous distribution are provided in the package **logcondens** and for observations in more than one dimension in **LogConDEAD**.

Examples

```
## see the help files for the abovementioned functions for examples
```

`internal` *Functions for estimation of a log-concave probability mass function via maximum likelihood*

Description

Internal functions for the estimation of a log-concave probability mass function. These functions are not intended to be called by the user directly.

`Direction` Compute vector that points in direction of $\max L(\psi)$ via Newton step.

`dMLE` Compute the vector ψ s.t. the log-likelihood function L, as implemented in `LikFunk`, is maximized over all PMFs (under no additional restrictions, though).

`GradientL` Gradient of `LikFunk`.

`HesseL` Hesse matrix of `LikFunk`.

`J00` Function introduced in Section 2.3 in Weyermann (2007), defined as

$$J^{\delta_k}(\psi_k, \psi_{k+1}) := \sum_{j=0}^{\delta_k} \exp\Big((1 - j/\delta_k)\psi_k + (j/\delta_k)\psi_{k+1}\Big).$$

This function is used to compute the value of the log-likelihood in `LikFunk`.

`J10` Derivative of $J^{\delta_k}(\psi_k, \psi_{k+1})$ w.r.t to the first argument.

`J11` Derivative of $J^{\delta_k}(\psi_k, \psi_{k+1})$ w.r.t to both arguments.

`J20` Second derivative of $J^{\delta_k}(\psi_k, \psi_{k+1})$ w.r.t to the first argument.

`LikFunk` The log-likelihood function for the discrete log-concave MLE.

`LocalCoarsen` Auxiliary function.

`LocalConcavity` Auxiliary function.

`LocalExtend` Auxiliary function.

`LocalMLE` Auxiliary function.

`LocalNormalize` Auxiliary function.

`StepSize` Auxiliary function.

Author(s)

Kaspar Rufibach (maintainer) <kaspar.rufibach@gmail.com>
http://www.kasparrufibach.ch
Fadoua Balabdaoui <fadoua@ceremade.dauphine.fr>
http://www.ceremade.dauphine.fr/~fadoua
Hanna Jankowski <hkj@mathstat.yorku.ca>
http://www.math.yorku.ca/~hkj
Kathrin Weyermann

References

Balabdaoui, F., Jankowski, H., Rufibach, K., and Pavlides, M. (2013). Maximum likelihood estimation and confidence bands for a discrete log-concave distribution. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **75**(4), 769–790.

Weyermann, K. (2007). An Active Set Algorithm for Log-Concave Discrete Distributions. *MSc thesis, University of Bern* (Supervisor: Lutz Duembgen).

See Also

All these functions are used by the function `logConDiscrMLE`.

`kInflatedLogConDiscr` *Compute a mixture of a point mass at 0 and a log-concave probability mass function from i.i.d. data*

Description

Using an EM algorithm, compute an estimate of a mixture of a point mass at k and a log-concave probability mass function from discrete i.i.d. data.

Usage

```
kInflatedLogConDiscr(x, k = 0, prec1 = 1e-10, prec2 = 1e-15,
    itermax = 200, output = TRUE, theta0 = 0.5, p0 = NA)
```

Arguments

`x`	Vector of observations.
`k`	Point at which inflation should be assumed. Must be in $x_1, x_1+1, \ldots, x_{n-1}, x_n$.
`prec1`	Precision for stopping criterion.
`prec2`	Precision to remove ends of support in case weights $<$`prec2`.

itermax	Maximal number of iterations of EM algorithm.
output	Logical, if TRUE, progress of EM algorithm is shown.
theta0	Optional initialization for θ_0, the point mass at k.
p0	Optional initialization for the PMF.

Details

Given a vector of observations $\boldsymbol{x}_n = (x_1, \ldots, x_n)$ from a discrete PMF with a (potential) point mass at k (typically $k = 0$), kInflatedLogConDiscr computes a pmf that is a mixture between a point mass at k and a log-concave PMF on x. To accomplish this, an EM algorithm is used.

Value

A list containing the following elementes:

z	The support.
f	The estimated k-inflated log-concave PMF.
E(L)	The value of the expected composite log-likelihood at the maximum.
loglik	The value of the composite log-likelihood at the maximum.
theta	The estimated weight at k.
logconc.pmf	The log-concave part of the mixture.
logconc.z	The support of logconc.pmf.

Author(s)

Kaspar Rufibach (maintainer) <kaspar.rufibach@gmail.com>
http://www.kasparrufibach.ch
Fadoua Balabdaoui <fadoua@ceremade.dauphine.fr>
http://www.ceremade.dauphine.fr/~fadoua
Hanna Jankowski <hkj@mathstat.yorku.ca>
http://www.math.yorku.ca/~hkj
Kathrin Weyermann

References

Balabdaoui, F., Jankowski, H., Rufibach, K., and Pavlides, M. (2013). Maximum likelihood estimation and confidence bands for a discrete log-concave distribution. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **75**(4), 769–790.

Weyermann, K. (2007). An Active Set Algorithm for Log-Concave Discrete Distributions. *MSc thesis, University of Bern* (Supervisor: Lutz Duembgen).

Examples

```
## ------------------------------------------------
## generate zero-inflated negative binomial sample
## ------------------------------------------------
```

```
set.seed(2011)
n <- 100
theta <- 0.05
r <- 6
p <- 0.3
x <- rnbinom(n, r, p)

## inflate at 0
x <- ifelse(runif(n) <= theta, 0, x)

## estimate log-concave MLE
fit1 <- logConDiscrMLE(x, w = NA, psi_o = NA, prec = 1e-05, output = TRUE)

## estimate zero-inflated log-concave MLE
fit2 <- kInflatedLogConDiscr(x, k = 0)

## plot the results
par(mfrow = c(1, 1), las = 1)
plot(fit1$x, exp(fit1$psi), type = "b", col = 2, xlim = range(x), xlab = "x",
    ylim = c(0, max(exp(fit1$psi), fit2$f)), ylab = "PMF",
    main = "Estimate MLE from a zero-inflated negative-binomial", pch = 19)
lines(fit2$z, fit2$f, type = "b", col = 4, pch = 15)

## add the true PMF we sampled from
z <- fit2$z
f.true <- theta * c(1, rep(0, length(z) - 1)) + (1 - theta) * dnbinom(z, r, p)
lines(z, f.true, col = 6, type = "b", pch = 17)

legend("topright", c("log-concave MLE", "zero-inflated log-concave MLE",
    "true PMF"), col = c(2, 4, 6), lty = c(1, 1, 1), pch = c(19, 15, 17),
    bty = "n")

## Not run:
## ------------------------------------------------
## generate seven-inflated negative binomial sample
## ------------------------------------------------
theta <- 0.05
r <- 4
p <- 0.3
n <- 10000
x <- rnbinom(n, r, p)
x <- ifelse(runif(n) <= theta, 7, x)
x <- c(x, rep(7, 10))

## compute different estimates
zero.mle <- kInflatedLogConDiscr(x, k = 7)
mle <- logConDiscrMLE(x, output = FALSE)
f.mle <- exp(mle$psiSupp)
z<-zero.mle$z
f1 <- theta * c(rep(0, 7 - min(x)), 1, rep(0, max(x) - 7))
f2 <- (1 - theta) * dnbinom(z, r, p)
f.true <- f1 + f2
true <- dnbinom(z, r, p)
```

```
f.fit <- zero.mle$f
xx <- sort(unique(x))
emp <- rep(0, length(z))
emp[xx - min(x) + 1] <- as.vector(table(x) / n)

## plot results
plot(z, f.true, type = "l", ylim = c(0, max(emp)), xlab = "x",
    ylab = "PMF", main = "Illustration k-inflated estimator")
points(z, true, type = "l", lty = 2)
points(z, f.fit, type = "l", col = "red")
points(z, zero.mle$logconc.pmf, type = "l", col = "red", lty = 2)
points(min(x):max(x), f.mle, type = "l", col = "green")
points(z, emp, type = "l", col = "purple")
points(z, emp, col = "purple")
legend("topright", inset = 0.05, c("true", "true less seven", "seven-inflated",
    "recovered", "logconc", "empirical"), lty=c(1, 2, 1, 2, 1, 1), col = c("black",
    "black", "red", "red", "green", "purple"))

## zoom in near mode
subs <- 4:12
plot(z[subs], f.true[subs], type = "l", ylim = c(0, max(emp)), xlab = "x",
    ylab = "PMF", main = "Illustration k-inflated estimator")
points(z[subs], true[subs], type = "l", lty = 2)
points(z[subs], f.fit[subs], type = "l", col = "red")
points(z[subs], zero.mle$logconc.pmf[subs], type = "l", col = "red", lty = 2)
points((min(x):max(x))[subs], f.mle[subs], type = "l", col = "green")
points(z[subs], emp[subs], type = "l", col = "purple")
points(z[subs], emp[subs], col = "purple")

## End(Not run)
```

`logConDiscrCI` *Compute pointwise confidence bands for the log-concave MLE of a PMF*

Description

Compute pointwise confidence bands for the log-concave maximum likelihood estimate of a log-concave probability mass function based on the limiting theory developed in Balabdaoui et al (2012).

Usage

```
logConDiscrCI(dat, conf.level = 0.95, type = c("MLE", "all")[1],
    B = 1000, output = TRUE, seed = 2011)
```

Arguments

`dat`	Data to compute MLE and confidence band for.
`conf.level`	The confidence level to be used.

type	To compute confidence bands one theoretically needs to know the knots of the true PMF. For type MLE the knots of the MLE will be used instead and for type all all observations will be considered knots. The latter is conservative and gives pointwise confidence intervals that are based on standard errors from a Normal approximation (the latter comes from the asymptotic theory in Balabdaoui et al, 2011).
B	Number of samples to be drawn to compute resampling quantiles.
output	If TRUE, progress of computations is output.
seed	Optional seed.

Details

The pointwise confidence bands are based on the limiting theory in Balabdaoui et al (2011).

Value

A list with the following components:

MLE	The estimated MLE (simply the output list of the function logConDiscrMLE applied to dat).
emp	A dataframe containing two columns: the unique sorted observations and the empirical PMF.
CIs	The computed confidence intervals for each $x \in \{$min(dat), ..., max(dat)$\}$.

Note

Values outside $[0, 1]$ will be clipped. As a consequence, coverage may be higher than $1 - \alpha$.

Author(s)

Kaspar Rufibach (maintainer) <kaspar.rufibach@gmail.com>
http://www.kasparrufibach.ch
Fadoua Balabdaoui <fadoua@ceremade.dauphine.fr>
http://www.ceremade.dauphine.fr/~fadoua
Hanna Jankowski <hkj@mathstat.yorku.ca>
http://www.math.yorku.ca/~hkj
Kathrin Weyermann

References

Balabdaoui, F., Jankowski, H., Rufibach, K., and Pavlides, M. (2013). Maximum likelihood estimation and confidence bands for a discrete log-concave distribution. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **75**(4), 769–790.

Weyermann, K. (2007). An Active Set Algorithm for Log-Concave Discrete Distributions. *MSc thesis, University of Bern* (Supervisor: Lutz Duembgen).

Examples

```
# ------------------------------------------------------------
# compute MLE and confidence bands for a random sample from a
# Poisson distribution
# ------------------------------------------------------------
set.seed(2011)
x <- sort(rpois(n = 100, lambda = 2))
mle <- logConDiscrMLE(x)
psi <- mle$psi

# compute confidence bands
CIs <- logConDiscrCI(x, type = "MLE", output = TRUE, seed = 20062011)$CIs

# plot estimated PMF and log of estimate
par(mfrow = c(1, 2), las = 1)
true <- dpois(0:20, lambda = 2)
plot(mle$x, exp(psi), type = "b", col = 2, xlim = c(min(x), max(x) + 1),
    xlab = "x", ylim = c(0, max(exp(psi), true, CIs[, 3])), ylab = "PMF",
    main = "Estimate MLE from a Poisson", pch = 19)
legend("topright", c("truth", "MLE", "confidence bands"), col = c(4, 2, 2),
    lty = c(1, 1, 2), pch = c(0, 19, NA), bty = "n")

# add true PMF
lines(0:20, true, type = "l", col = 4)

# add confidence bands
matlines(CIs[, 1], CIs[, 2:3], type = "l", lty = 2, col = 2)

# log-density
plot(mle$x, psi, type = "p", col = 2, xlim = c(min(x), max(x) + 1),
    xlab = "x", ylab = "PMF", main = "Estimate MLE from a Poisson",
    pch = 19, ylim = c(-6, log(max(exp(psi), true, CIs[, 3]))))
lines(0:20, log(true), type = "l", col = 4)

# add confidence bands
matlines(CIs[, 1], log(CIs[, 2:3]), type = "l", lty = 2, col = 2)


# ------------------------------------------------------------
# compute confidence bands when only estimate (not original data)
# are available (as a an example we simply use the estimator from
# above)
# ------------------------------------------------------------
x.est <- 0:6
est <- c(0.09, 0.30, 0.30, 0.19, 0.09, 0.02, 0.01)

# generate original data (up to given precision)
x <- rep(0:6, times = 100 * est)
```

---

logConDiscrMLE *Compute log-concave probability mass function from i.i.d. data*

Description

Compute the maximum likelihood estimate of a log-concave probability mass function from discrete i.i.d. data.

Usage

```
logConDiscrMLE(x, w = NA, psi_o = NA, prec = 1e-05, output = TRUE)
```

Arguments

x	Vector of observations. If `w = NA`, then weights will be generated for each non-unique observation of `x`.
w	If `w = NA`, weights will be generated from `x`. If `w != NA`, then it is assumed that `x` and `w` are of equal length and the elements in `w` correspond to the weights in `x`.
psi_o	Optional start vector.
prec	Precision for stopping criterion.
output	Logical, if `TRUE`, progress of the active set algorithm is shown.

Details

Given a vector of observations $\boldsymbol{x}_n = (x_1, \ldots, x_n)$ from a discrete PMF, `logConDiscrMLE` computes a function $\widehat{p}_k$ on $\{x_1, \ldots, x_n\}$ with knots only in $\{x_1, \ldots, x_n\}$ such that

$$L(\boldsymbol{p}) = \sum_{i=1}^{n} w_i \log(p_i)$$

is maximal over all log-concave PMFs $\{p_k\}, k = 1, \ldots, n$, where w_i is the frequency of the observation x_i. To accomplish this, an active set algorithm is used.

Value

A list containing the following elementes:

x	Vector of unique observations, sorted.
w	Generated weights.
psi	The estimated log-density on the grid of unique, sorted observations.
L	The value of the log-likelihood at the maximum.
IsKnot	Binary vector where `isKnot_k = 1` if ψ has a knot at x_k.
xSupp	The full support $\{x_1, x_1 + 1, \ldots, x_m - 1, x_m\}$.
psiSupp	ψ interpolated on `xSupp`.

Author(s)

Kaspar Rufibach (maintainer) <kaspar.rufibach@gmail.com>
http://www.kasparrufibach.ch
Fadoua Balabdaoui <fadoua@ceremade.dauphine.fr>
http://www.ceremade.dauphine.fr/~fadoua
Hanna Jankowski <hkj@mathstat.yorku.ca>
http://www.math.yorku.ca/~hkj
Kathrin Weyermann

References

Balabdaoui, F., Jankowski, H., Rufibach, K., and Pavlides, M. (2013). Maximum likelihood estimation and confidence bands for a discrete log-concave distribution. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **75**(4), 769–790.

Weyermann, K. (2007). An Active Set Algorithm for Log-Concave Discrete Distributions. *MSc thesis, University of Bern* (Supervisor: Lutz Duembgen).

Examples

```
# ------------------------------------------------------------
# compute MLE for a random sample from a Poisson distribution
# ------------------------------------------------------------
x <- sort(rpois(n = 100, lambda = 2))
mle <- logConDiscrMLE(x)
psi <- mle$psi

# plot estimated PMF and log of estimate
par(mfrow = c(1, 2), las = 1)
true <- dpois(0:20, lambda = 2)
plot(mle$x, exp(psi), type = "p", col = 2, xlim = range(x), xlab = "x",
    ylim = c(0, max(exp(psi), true)), ylab = "PMF",
    main = "Estimate MLE from a Poisson", pch = 19)
legend("topright", c("truth", "MLE"), col = c(4, 2), lty = c(1, 0),
    pch = c(0, 19), bty = "n")

# add true PMF
lines(0:20, true, type = "l", col = 4)

# log-density
plot(mle$x, psi, type = "p", col = 2, xlim = range(x), xlab = "x",
    ylab = "PMF", main = "Estimate MLE from a Poisson", pch = 19)
lines(0:20, log(true), type = "l", col = 4)

# use a priori specified weights: mle = mle2
mle2 <- logConDiscrMLE(x = unique(x), w = table(x))


## ------------------------------------------------------------
## Illustrate the limit process: the code below can be used to
## to reproduce the limit process figure in Balabdaoui et al (2011)
```

```
## ------------------------------------------------------------
a <- 1
b <- 7
c <- 8
d <- 11
e <- 2
n <- 10 ^ 2

## support
x <- seq(a, d, by = 1)

## true density
dens <- dTriangular(a, b, c, d, e)
logdens <- log(dens)
rand <- rTriangular(n, a, b, c, d, e)$rand

## empirical
emp <- table(rand) / n
x.emp <- names(table(rand))

## log-concave MLE
mle <- logConDiscrMLE(rand, output = FALSE)

## plot log PMF and PMF
par(mfrow = c(1, 2))
plot(x, logdens, type = "l", col = 1, pch = 19, main = "log-density",
    xlim = c(a, d), ylim = range(range(log(emp), logdens)))
lines(x, logdens, type = "l", col = 1, lwd = 0.1)
points(x.emp, log(emp), col = 4, pch = 19)
points(mle$x, mle$psi, col = 6, pch = 19)
abline(v = mle$x[mle$isKnot == 1], lty = 3, col = 3)

plot(x, dens, type = "l", col = 1, pch = 19, main = "density",
    xlim = c(a, d), ylim = c(0, max(dens, emp)))
lines(x, dens, type = "l", col = 1, lwd = 0.1)
points(x.emp, emp, col = 4, pch = 19)
points(mle$x, exp(mle$psi), col = 6, pch = 19)
legend("topleft", c("truth", "MLE", "knots of the MLE", "empirical"),
    col = c(1, 6, 3, 4), pch = c(NA, 19, NA, 19), lty = c(1, NA, 3, NA),
    bg = "white", bty = "n")
abline(v = mle$x[mle$isKnot == 1], lty = 3, col = 3)


## ------------------------------------------------------------
## Now compute and plot Y(x) and H(x)
## ------------------------------------------------------------
xla <- paste("x = {r = ", a, ", ..., s - 1 = ", b - 1, "}", sep = "")
par(mfcol = c(2, 2), oma = rep(0, 4), mar = c(4.5, 4.5, 2, 1), las = 1)
plot(x, logdens, type = "b", col = 2, pch = 19, main = "log of
    normalized triangular pmf", xlim = c(a, d), xaxt = "n", xlab = "x",
    ylab = "log of normalized pmf")
axis(1, at = c(a, b, d), labels = paste(c("a = ", "b = ", "d = "),
    c(a, b, d), sep = ""))
```

```
## compute H(x)
r <- a
s <- b
ind <- r:(s - 1)
px <- dens
p_rs <- px[ind]
m <- s - r

## ------------------------------------------------------------
## generate one observation from the distribution of U(F(x)) - U(F(x - 1))
## ------------------------------------------------------------
sigma <- diag(m) * 0
for (i in 1:m){
    for (j in 1:m){
        sigma[i, j] <- p_rs[i] * (i == j) - p_rs[i] * p_rs[j]
    }
}

set.seed(23041977)
cx <- rep(NA, d - a + 1)
cx[ind] <- rmvnorm(1, mean = rep(0, m), sigma = sigma, method =
    c("eigen", "svd", "chol")[3])
Ux <- rep(NA, length(x))
Ux[ind] <- cx[ind]

X <- x[ind]
Y <- Ux[ind] / p_rs
W <- p_rs

## concave regression using 'cobs'
Res <- conreg(x = X, y = Y, w = W, verbose = TRUE)
g <- Res$yf
gKnots <- Res$iKnots
plot(X, Y, main = expression("The concave function g* that
    minimizes "*Phi*"(g)"), xaxt = "n", ylab = "g*", ylim =
    range(c(Y, g)), xlab = xla, type = "n")
axis(1, at = 0:100, labels = 0:100); abline(v = x[gKnots],
    lty = 2, col = grey(0.75))
lines(X, g, lwd = 2, col = 3, type = "b", pch = 1)
lines(X, Y, lwd = 1, col = 2, type = "p", pch = 19)
legend("bottomright", c("values of cx / px", "minimizer g*"),
    lty = c(NA, 1), pch = c(19, 1), col = 2:3, bty = "n",
    lwd = c(NA, 2))

## compute H(x) for x = r, ..., s - 1 and plot it
gstar <- rep(NA, length(x))
gstar[ind] <- g
xs <- r:(s - 1)
Hs <- rep(0, length(xs))
for (i in 2:length(xs)){
    for (ks in r:(xs[i] - 1)){
        js <- r:ks
```

```
        Hs[i] <- Hs[i] + sum(gstar[js] * px[js])
    }
}

## check
(Hs[3:length(Hs)] - 2 * Hs[2:(length(Hs) - 1)] + Hs[1:(length(Hs) - 2)]) / p_rs[2:(length(Hs) - 1)]
gstar

## ------------------------------------------------------------
## plot the product of g* and px (the limit of the PMF)
## ------------------------------------------------------------
plot(x[ind], gstar[ind] * p_rs, main = expression("g"^"*"* " * p"),
    xaxt = "n", pch = 19, col = 2, ylab = "g*", type = "b", xlab = xla)
axis(1, at = 0:100, labels = 0:100); abline(v = x[gKnots], lty = 2,
    col = grey(0.75))

## compute Y(x) for x = r, ..., s - 1 and plot it
Ys <- rep(0, length(xs))
for (i in 2:length(xs)){
    for (ks in r:(xs[i] - 1)){
        js <- r:ks
        Ys[i] <- Ys[i] + sum(cx[js])
    }
}

## plot the two processes
plot(xs, Ys, type = "n", col = 2, xaxt = "n", lwd = 2, main = "The
    processes H(x) and Y(x)", ylab = "H and Y", xlab = xla)
axis(1, at = 0:100, labels = 0:100); abline(v = x[gKnots], lty = 2,
    col = grey(0.75))
lines(xs, Hs, col = 2, lwd = 1, type = "b")
lines(xs, Ys, col = 3, lwd = 2, type = "l", lty = 2)
legend("topleft", c("H(x)", "Y(x)"), col = 2:3, lty = c(1, 2), pch = 1,
    bty = "n", lwd = c(1, 2))
```

`Triangular` *Functions to compute a and simulate from a triangular probability mass function*

Description

In Balabdaoui et al (2012) the triangular density, defined as

$$p_x^{a,b,c,d,e} = c(x-a)/(b-a)$$

for $x \in \{a, \ldots, c\}$ and

$$p_x^{a,b,c,d,e} = (e-c)(x-b)/(d-b) + c$$

for $x \in \{c, \ldots, d\}$, was used to illustrate the limit process of the log-concave MLE. In order to provide the code to generate the limit process figure in Balabdaoui et al (2012, see the example in logConDiscrMLE for the code to generate that figure) the functions dTriangular and rTriangular are included in this package. Note that rTriangular uses the rejection sampling algorithm in Devroye (1987) which was specifically developed to generate random numbers from a log-concave PMF.

Usage

```
dTriangular(a, b, c, d, e)
rTriangular(n, a, b, c, d, e)
```

Arguments

a	Left endpoint of triangular pmf.
b	Mode of triangular pmf.
c	Height at mode.
d	Left endpoint.
e	Height at left endpoint.
n	Number of random numbers to generate.

Value

dTriangular returns a vector containing the value of the PMF at all values in $x \in \{a, \ldots, d\}$. rTriangular returns a list containing the elements:

rand	Vector with generated random numbers of length n.
x	Vector $(a, \ldots, d)$.
dens	Value of the pmf at x.

Note

This function is used to generate the plot of the limit process in the help file for the function logConDiscrMLE.

Author(s)

Kaspar Rufibach (maintainer) <kaspar.rufibach@gmail.com>
http://www.kasparrufibach.ch
Fadoua Balabdaoui <fadoua@ceremade.dauphine.fr>
http://www.ceremade.dauphine.fr/~fadoua
Hanna Jankowski <hkj@mathstat.yorku.ca>
http://www.math.yorku.ca/~hkj
Kathrin Weyermann

References

Balabdaoui, F., Jankowski, H., Rufibach, K., and Pavlides, M. (2013). Maximum likelihood estimation and confidence bands for a discrete log-concave distribution. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **75**(4), 769–790.

Devroye, L. (1987). A simple generator for discrete log-concave distributions. *Computing*, **39**, 87-91.

Examples

```
## ------------------------------------------------------------
## compute values of triangular density and simulate from it
## ------------------------------------------------------------
a <- 1
b <- 7
c <- 8
d <- 11
e <- 2
n <- 10 ^ 2

## support
x <- seq(a, d, by = 1)

## true density
dens <- dTriangular(a, b, c, d, e)
logdens <- log(dens)
rand <- rTriangular(n, a, b, c, d, e)$rand

## does the same as rTriangular()
rand2 <- sample(x = a:d, size = n, prob = dens, replace = TRUE)
```

Index