Package 'ars'

December 17, 2015

Title Adaptive rejection sampler

Version 1.0

Description User inputs log of a density function. Returns n points using adaptive rejection sampling, as described by Gilks et al. (1992)

Depends R (>= 3.2.2)

Author Zhenyuan Liu, Thibault Doutre, Hao Lyu, Paul Cho

Maintainer Paul Cho <calpcho@berkeley.edu>

License GPL (>=2)

LazyData true

RoxygenNote 5.0.1

Imports pracma

Suggests testthat

NeedsCompilation no

R topics documented:

`abscissae` *Abscissae function*

Description

Looks for suitable x1 and xk if they are not given, this can ensure the adaptive algorithm is not biased as well as avoiding numerical issues. It can then generate the initial grid, the default number of nodes is 5. Depending on the domain, different algorithms are used, but they are all adaptive. The relaxation factor and minimum step can be defined as well.

Usage

```
abscissae(h,domain,x1 = NULL,xk =NULL,x0=0,nmesh=5,min_step=0.001,relax_factor=5)
```

Arguments

`h`	The log of the density function
`domain`	A vector of length 2 giving the domain (left bound, right bound) of the density function
`x1`	The leftmost node
`xk`	The rightmost node
`x0`	An initial point for searching for suitable x1 and xk. If provided by the user based on the knowledge of the density function, then it would be faster to find x1 and xk
`nmesh`	The number of nodes in the initial grid
`min_step`	The minimum step allowed in the adaptive-step search algorithm, if the variance is extremely small, then the used can decrease the min_step, yet it would be slower
`relax_factor`	The relaxation factor in the adaptive-step search algorithm

`ars` *Adaptive rejection sampler The main ars() function*

Description

Adaptive rejection sampler The main ars() function

Usage

```
ars(h, n, domain, x1 = NULL, xk = NULL, x0 = 0, nmesh = 5,
  x_start = 0, max_x = 10000, min_step = 0.01, relax_factor = 10)
```

Arguments

h	The log of the density function
n	The number of points to be sampled
domain	A vector of length 2 giving the domain (left bound, right bound) of the density function
x1	The leftmost node of the abscissae, if not provided, the abscissae() will try to find one which has a suitable slope, leading to a unbiased and numerical-issues friendly sampling process
xk	The the rightmost node of the abscissae, if not provided, the abscissae() will try to find one which has a suitable slope, leading to a unbiased and numerical-issues friendly sampling process
x0	An initial point for searching for suitable x1 and xk. If provided by the user based on the knowledge of the density function, then it would be faster to find x1 and xk. default = 0
nmesh	The number of nodes in the initial grid. default = 5
x_start	If x1 and xk are not provided, ars will call search() to adaptively search for an appropriate starting point to pass to abscissae() as x0. default = 0
max_x	The maximum search limit for x. default is set to 10000. If the mode is shifted beyond this limit, the user should define this parameter.
min_step	The minimum size of steps to take while searching for appropriate x1 to xk. default = .01
relax_factor	The relaxation factor in the adaptive-step search algorithm. default = 10

Value

A vector with n sampled points from density of interest.

Examples

```
samples <- ars( function(x) log( dnorm(x, 0, 1) ), 10000, c(-Inf, Inf))
```

---

envelop *Envelope function*

Description

Calculates the upper hull of h. The returned value is a matrix with each row storing (h'(x), h(x_i),x_i,xmin,xmax) for each node. In this way, each upper hull is uniquely defined, as well as its corresponding domain.

Usage

```
envelop(h,x,domain)
```

Arguments

h	The log of the density function
x	The nodes, found by abscissae()
domain	A vector of length 2 giving the domain (left bound, right bound) of the density function

envelop_density *Envelope density*

Description

Returns a normalized density function using the upper hull. A matrix with each row (h'(x_i), h(x_i),x_i,xmin,xmax,cumulative probability, normalized cumulative probability) at each node is returned. With this infomation we can sample with sk.

Usage

```
envelop_density(h,x,domain)
```

Arguments

h	The log of the density function
x	The nodes, found by abscissae()
domain	A vector of length 2 giving the domain (left bound, right bound) of the density function

log_concavity *Log concavity test*

Description

Checks that at each node, the upper hull and lower hull would bound the h function. The slope of the envelop lines and squeezing lines are examined.

Usage

```
log_concavity(u,l)
```

Arguments

u	the upper hull calculated using envelop(h,x,domain)
l	the lower hull calculated using squeezing(h,x)

rejection_test *Rejection test*

Description

Performs the rejection test for a newly sampled x if it fails the squeezing test.

Usage

```
rejection_test(x_sampled,u,h)
```

Arguments

x_sampled	a newly sampled x, from sample_one_point()
u	the upper hull calculated using envelop(h,x,domain)
h	the log density function

sample_one_point *Sampling one point*

Description

Can sample one point from the given piece-wise exponential density using the upper hull. The Inverse-CDF method is used here. A random number is generated, then it's used with the help of the calculated inverse CDF function of the piece-wise exponential density to generate an x.

Usage

```
sample_one_point(s)
```

Arguments

s	A matrix with each row (h'(x), h(x_i),x_i,xmin,xmax,cumulative probability, normalized cumulative probability) at each node. This can be calculated using envelop_density(h,x,domain)

search *Search function*

Description

Searches for a x0 such that h(x0) is finite and the gradient of h at x0 is attained. This is particular useful when the domain is unbounded, the mode is far away from 0 (yet unknown) and the variance is very small, i.e. the density function is very narrow. In such cases, it's even difficult to make a guess for an initial value to find x1 and xk.

Usage

```
search(h,domain,x_start,max_x,min_step,relax_factor)
```

Arguments

h	The log of the density function
domain	A vector of length 2 giving the domain (left bound, right bound) of the density function
x_start	The starting x value for the search
max_x	The maximum search limit for x, if the mode is shifted beyond this limit, the user should define this parameter
min_step	The minimum step allowed in the adaptive-step search algorithm, if the variance is extremely small, then the used can decrease the min_step, yet it would be slower
relax_factor	The relaxation factor in the adaptive-step search algorithm

squeezing *Squeezing function*

Description

Calculates the lower hull of h. The returned value is a matrix with each row storing (slope, h(x_i),x_i) for each node. In this way, each lower hull is uniquely defined. Note that the number of squeezing lines is that of the envelop lines plus one.

Usage

```
squeezing(h,x)
```

Arguments

h	The log of the density function
x	The nodes, found by abscissae()

squeezing_test *Squeezing test*

Description

Performs the squeezing test for a newly sampled x

Usage

```
squeezing_test(x_sampled,l,u)
```

Arguments

x_sampled	a newly sampled x, from sample_one_point()
l	the lower hull calculated using squeezing(h,x)
u	the upper hull calculated using envelop(h,x,domain)

update_grid *Updating step*

Description

Updates the grid if both h and the gradient of h is evaluated at a sampled point.#' @usage envelop_density(h,x,domain)

Usage

```
update_grid(x, x_add)
```

Arguments

x	The abscissae in the previous step
x_add	An x value to be added to the abscissae