Package 'ars'

December 16, 2015

Title A package for stat243 final project

Description An adaptive-rejection sampler for a log-concave function f(x) with domain D.

Version 0.1

Author Sici Huang, Cheng Ju, Fengshi Niu, Yun Zhou (in alphabetical order).

Maintainer Cheng Ju <jucheng1992@gmail.com>

Depends R (>= 3.2.0), numDeriv, dplyr, testthat

License GPL-3

NeedsCompilation no

R topics documented:

`abscissaeSummary` *abscissaeSummary*

Description

Given a function and a sequence of points, function computes the value of the given function and its derivative at each point.

Usage

```
abscissaeSummary(x, h)
```

Arguments

x	a numeric vector composed of the abscissae
h	the original function in log scale

Value

a numeric matrix, with the three columns being the x's, h(x)'s, and derivatives of the h(x)'s respectively. If h(x) is Inf, the corresponding x and h'(x) will be dropped.

Examples

```
x <- seq(-2, 2, length.out = 10)
abscissaeSummary(x, function(x) dnorm(x))
```

---

ars *A function that returns samples from unnormalized density based on reject sampling*

Description

A function that returns samples from unnormalized density based on reject sampling

Usage

```
ars(n, g, k = 30, left = -Inf, right = Inf, stepsize = 10)
```

Arguments

n	a numeric value as sample size.
g	a log-concave function.
k	a numeric value as the max size of initial abscissae.
left	a numeric value as the left bound of input h; default is -Inf.
right	a numeric value as the right bound of input h; default is Inf.
stepsize	an integer that indicates the number of samples generated in the first iteration of sampling; it increases along iterations at a predetermined rate 1.1.

Value

a numeric vector of length n that was sampled from the normalized density function d.

Examples

```
ars(n = 1000, g = dnorm)
```

computeZ *computeZ*

Description

A function that computes the intersections of the tangent lines, provided the abscissae matrix

Usage

```
computeZ(abscissae_summary)
```

Arguments

abscissae_summary	the abscissae matrix with the three columns being the x's, h(x)'s, and derivatives of the h(x)'s respectively. There should be no Inf's in the third column

Value

a numeric vector consists of the x-coordinates of the intersection points (i.e. z in paper by Gilks et al.)

---

findInitAbsc *A function that returns a matrix with the three columns being the abscissae x1,...,xk and their corresponding h(x)'s and h'(x)'s respectively.*

Description

A function that returns a matrix with the three columns being the abscissae x1,...,xk and their corresponding h(x)'s and h'(x)'s respectively.

Usage

```
findInitAbsc(g, k, left = -Inf, right = Inf, c = 3)
```

Arguments

g	a log-concave function.
k	a numeric value that indicates the max size of the abscissae.
left	a numeric value that indicates the left bound for domain of function g; default value is -Inf
right	a numeric value that indicates the right bound for domain of function g; default value is Inf
c	a numeric value by which we shift the finite bound to find initial value for the optim function; default value is 3

Value

mat the matrix of interest

Examples

```
findInitAbsc(function(x){
   return(dnorm(x, mean=3, sd=2))
 },6)

 # Chi-square distribution
 g <- function(x){
   return(dchisq(x, 10, ncp=3))
 }
 findInitAbsc(g,4,3)

 # Uniform distribution
 g <- function(x){
   return(dunif(x, min=2, max=5))
 }
 findInitAbsc(g,6,2,5)

 # Piecewise constant distribution
 h <- function(x){
   a <- sapply(x,function(x){if(x<1) return (x)
     else if(x>=1 & x <=2) return (1)
     else if(x>2) return (-x+3)})
   return (a)
 }
 g <- function(x){
   return(exp(h(x)))
 }
 findInitAbsc(g,6)
```

`findMode` *A function that returns the mode of a function*

Description

A function that returns the mode of a function

Usage

```
findMode(g, left = -Inf, right = Inf, c = 3)
```

Arguments

`g`	a log-concave function.
`left`	a numeric value that indicates the left bound for domain of function g; default value is -Inf
`right`	a numeric value that indicates the right bound for domain of function g; default value is Inf
`c`	a numeric value by which we shift the finite bound to find initial value for the optim function; default value is 3

Value

mode a numeric value that is the mode of the function h=log(g)

Examples

```
# Normal distribution
 g <- function(x){
   return(dnorm(x, mean=3, sd=2))
 }
 findMode(g)

 # Chi-square distribution
 g <- function(x){
   return(dchisq(x, 10, ncp=3))
 }
 findMode(g,3)

 # Uniform distribution
 g <- function(x){
   return(dunif(x, min=2, max=5))
 }
 findMode(g,2,5)

 # Piecewise constant distribution
 h <- function(x){
   a <- sapply(x,function(x){if(x<1) return (x)
     else if(x>=1 & x <=2) return (1)
     else if(x>2) return (-x+3)})
   return (a)
 }
 g <- function(x){
   return(exp(h(x)))
 }
 findMode(g)
```

`lowerHull` *A function that, given a vector x, returns the corresponding values of the upper hull function*

Description

A function that, given a vector x, returns the corresponding values of the upper hull function

Usage

```
lowerHull(x, abscissae_summary)
```

Arguments

`x`	a numeric vector consists of points on the x-axis
`abscissae_summary`	a matrix with the three columns being x, h(x), and h'(x) at the support points respectively

Value

U: a numeric vector that contains the corresponding upper hull values, with same length as x

Examples

```
h <- function(x){
  return(log(dnorm(x)))
}

xSupport <- seq(-3, 3, length.out = 10)

abscissae_summary = abscissaeSummary(xSupport,h)

x <- seq(-3, 3, length.out = 10)
upperHull(x,abscissae_summary)
lowerHull(x,abscissae_summary)

x <- seq(-3, 3, length.out = 30)
upperHull(x,abscissae_summary)
lowerHull(x,abscissae_summary)
```

`sampleUpper` *sampleUpper*

Description

Sample from upper hull based on inverse CDF

Usage

```
sampleUpper(n, CDFInverse)
```

Arguments

`n`	a positive integer as the number of samples
`CDFInverse`	an inverse CDF function of the upper hull

Value

a numeric vector sampled from upper hull, which has size n.

`upperCDF` *upperCDF*

Description

Given a numeric vector and the abscissae matrix, function returns the CDF function of the upper hull.

Usage

```
upperCDF(x, abscissae_summary, xlow = -Inf, xhigh = Inf)
```

Arguments

x	a numeric vector that takes values in [0,1] as the input for the CDF
abscissae_summary	a numeric matrix comprised of abscissae together with h(x) and h'(x)
xlow	a numeric value as the left bound of the support of g
xhigh	a numeric value as the right bound of the support of g

Value

A numeric vector consists of the values of CDF for the upper hull at each abscissa.

---

upperCDFInverse *upperCDFInverse*

Description

Given a numeric vector and the abscissae matrix, function returns the inverse CDF function of the upper hull.

Usage

```
upperCDFInverse(x, abscissae_summary, xlow = -Inf, xhigh = Inf)
```

Arguments

x	a numeric vector that takes values in [0,1] as the input for the CDF
abscissae_summary	a numeric matrix comprised of abscissae together with h(x) and h'(x)
xlow	a numeric value as the left bound of the support of g
xhigh	a numeric value as the right bound of the support of g

Value

A numeric vector consists of the values of the inverse CDF for the upper hull at each abscissa.

---

upperHull *A function that, given a vector x, returns the corresponding function values from the upper hull*

Description

A function that, given a vector x, returns the corresponding function values from the upper hull

Usage

```
upperHull(x, abscissae_summary)
```

Arguments

x	a numeric vector consists of points on the x-axis
abscissae_summary	a matrix with the three columns being x, h(x), and h'(x) at the support points respectively

Value

U: a numeric vector that contains the corresponding upper hull values, with same length as x

Examples

```
# Test1: log of normal distribution
h <- function(x){
  return(log(dnorm(x)))
}

xSupport <- seq(-3, 3, length.out = 10)

abscissae_summary = abscissaeSummary(xSupport,h)

x <- seq(-3, 3, length.out = 10)
upperHull(x,abscissae_summary)
lowerHull(x,abscissae_summary)

x <- seq(-3, 3, length.out = 30)
upperHull(x,abscissae_summary)
lowerHull(x,abscissae_summary)

h <- function(x){
 return(log(dchisq(x,df = 2)))
}

xSupport <- seq(-3, 3, length.out = 10)
abscissae_summary = abscissaeSummary(xSupport,h)
x <- seq(-3, 3, length.out = 10)
upperHull(x,abscissae_summary)
lowerHull(x,abscissae_summary)

# Test2: log of uniform distribution
h <- function(x){
 return(log(dunif(x,0,1)))
}

xSupport <- seq(-3, 3, length.out = 100)
abscissae_summary = abscissaeSummary(xSupport,h)
x <- seq(-3, 3, length.out = 10)
x <- seq(0.2, 0.9, length.out = 10)
upperHull(x,abscissae_summary)
lowerHull(x,abscissae_summary)

# Test3: log of piecewise constant distribution
h <- function(x){
 a= sapply(x,function(x){if(x<1) return (x)
 else if(x>=1 & x <=2) return (1)
 else if(x>2) return (-x+3)})
```

```
 return (a)
}

xSupport <- seq(-3, 3, length.out = 15)
abscissae_summary = abscissaeSummary(xSupport,h)
x <- seq(-3, 3, length.out = 10)
x <- seq(0.2, 0.9, length.out = 10)
upperHull(x,abscissae_summary)
lowerHull(x,abscissae_summary)
```

Index