Test of ARS man

November 18, 2013

`ARSpackage-package` *What the package does (short line) ~~ package title ~~*

Description

More about what it does (maybe more than one line) ~~ A concise (1-5 lines) description of the package ~~

Details

Package:	ARSpackage
Type:	Package
Version:	1.0
Date:	2013-11-18
License:	What license is it under?
Depends:	methods

~~ An overview of how to use the package, including the most important functions ~~

Author(s)

Who wrote it

Maintainer: Who to complain to <yourfault@somewhere.net> ~~ The author and/or maintainer of the package ~~

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~ ~~ <pkg> ~~

Examples

```
~~ simple examples of the most important functions ~~
```

Cadapt_reject_sample-class *Class* "Cadapt_reject_sample"

Objects from the Class

Objects can be created by calls of the form new("Cadapt_reject_sample", n, h_x, h_prime).

Slots

n: Object of class "numeric" ~~
h_x: Object of class "function" ~~
h_prime: Object of class "function" ~~

Methods

error_check signature(object = "Cadapt_reject_sample"): ...
gen_x signature(object = "Cadapt_reject_sample"): ...
initialize signature(.Object = "Cadapt_reject_sample"): ...
lower signature(object = "Cadapt_reject_sample"): ...
sample signature(object = "Cadapt_reject_sample"): ...
show signature(object = "Cadapt_reject_sample"): ...
update signature(object = "Cadapt_reject_sample"): ...
upper signature(object = "Cadapt_reject_sample"): ...

Examples

```
showClass("Cadapt_reject_sample")
```

---

a_r_s *The adapt_reject function*

Description

This calls the class Cadapt_reject_sample and it's methods.

Usage

```
a_r_s(n_samples, log_fx, log_fx_prime, ...)
```

Arguments

n_samples	Number of samples desired from distribution
log_fx	Log of function to sample from
log_fx_prime	First derivative of log of function to sample from

Value

S4 adapt_reject_sample object; a vector containing

n

Cadapt_reject_sample *The adapt_reject class*

Description

This file consists of the adapt_reject_sample class, the function that calls it and the functions it uses.

Details

It performs an adaptive rejection sampling process as proposed by Wild and Gilks in 1992.

Value

S4 `adapt_reject_sample` object; a vector containing

`n`

Slots

`n`: Variable of class `"numeric"`, n, containing the number of points to sample

`h_x`: Function of class `"function"`, containing the log(f(x)) to sample from.

`h_prime`: Function of class `"function"`, containing the first derivative log(f(x)) to sample from.

Note

1. Initialize i) x1, x2 ii) inputs: h(x) and h'(x), n (number of points to sample), optional: domain etc iii) error checks: make sure that the function is concave up and the function lies within U(x) and L(x). Check that x1 has a positive slope and X2 has a negative slope. Check that the sample size is positive and an integer. 2) Objects/methods: i) U(x) and S(x): z(x), equations for tangent lines ii) List of x points iii) list of sampled points iv) l(x) v) sample function from s(x) and uniform random number vi) update steps vii) error checking

Author(s)

J. Bladen, L. Felberg, H.W. Tsao, S. Tu

References

http://faculty.chicagobooth.edu/hedibert.lopes/teaching/ccis2010/1992GilksWild.pdf

gen_x *Cadapt_reject_sample generating first two points*

Description

Cadapt_reject_sample generating first two points

Cadapt_reject_sample initialization

Cadapt_reject_sample show

Cadapt_reject_sample error_check

Cadapt_reject_sample upper

Cadapt_reject_sample lower

Cadapt_reject_sample sample

Cadapt_reject_sample update

Arguments

object	Cadapt_reject_sample object
object	Cadapt_reject_sample object
object	Cadapt_reject_sample object
object	Cadapt_reject_sample object
object	Cadapt_reject_sample object
object	Cadapt_reject_sample object
object	Cadapt_reject_sample object
object	Cadapt_reject_sample object

error_check-methods *~~ Methods for Function* error_check *~~*

Description

~~ Methods for function error_check ~~

Methods

signature(object = "Cadapt_reject_sample")

error_check *Error checking generic*

Description

Error checking generic

Arguments

object	An object

gen_x-methods ~~ *Methods for Function* gen_x ~~

Description

~~ Methods for function gen_x ~~

Methods

```
signature(object = "Cadapt_reject_sample")
```

gen_x *Random generating first two points*

Description

Random generating first two points

Arguments

object	An object

initialize-methods ~~ *Methods for Function* initialize ~~

Description

~~ Methods for function initialize ~~

Methods

```
signature(.Object = "Cadapt_reject_sample")
```

lower-methods ~~ *Methods for Function* lower ~~

Description

~~ Methods for function lower ~~

Methods

```
signature(object = "Cadapt_reject_sample")
```

lower *Lower generic*

Description

Lower generic

Arguments

object	An object

sample-methods ~~ *Methods for Function* sample ~~

Description

~~ Methods for function sample ~~

Methods

```
signature(object = "Cadapt_reject_sample")
```

sample *Sanple generic*

Description

Sanple generic

Arguments

object	An object

show-methods ~~ *Methods for Function* show ~~

Description

~~ Methods for function show ~~

Methods

```
signature(object = "Cadapt_reject_sample")
```

update-methods ~~ *Methods for Function* update ~~

Description

~~ Methods for function update ~~

Methods

signature(object = "Cadapt_reject_sample")

update *Update generic*

Description

Update generic

Arguments

object An object

upper-methods ~~ *Methods for Function* upper ~~

Description

~~ Methods for function upper ~~

Methods

signature(object = "Cadapt_reject_sample")

upper *Upper generic*

Description

Upper generic

Arguments

object An object

validity_ars

Usage

```
validity_ars(object)
```

Arguments

```
object
```

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (object)
{
    if (is.integer(n) == FALSE) {
        stop("Input number of steps is not an integer")
    }
    if (n <= 0) {
        stop("Input number of steps is not greater than zero")
    }
  }
```

Index