Package 'multIntTestFunc'

September 7, 2024

Type Package

Title Provides Test Functions for Multivariate Integration

Version 0.3.0

Maintainer Klaus Herrmann <klaus.herrmann@usherbrooke.ca>

Description Provides implementations of functions that can be used to test multivariate integration routines. The package covers six different integration domains (unit hypercube, unit ball, unit sphere, standard simplex, non-negative real numbers and R^n). For each domain several functions with different properties (smooth, non-differentiable, ...) are available. The functions are available in all dimensions n >= 1. For each function the exact value of the integral is known and implemented to allow testing the accuracy of multivariate integration routines. Details on the available test functions can be found at on the development website.

License MIT + file LICENSE

URL https://github.com/KlausHerrmann/multIntTestFunc

Imports methods, mvtnorm, pracma, stats

Suggests knitr, rmarkdown, statmod, testthat

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.1.9000

Collate 'AllGeneric.R' 'Pn_lognormalDensity.R' 'Pn_logtDensity.R' 'Rn_Gauss.R' 'Rn_floorNorm.R' 'Rn_normalDensity.R' 'Rn_tDensity.R' 'domainChecks.R' 'misc.R' 'multIntTestFunc.R' 'standardSimplex_Dirichlet.R' 'standardSimplex_exp_sum.R' 'unitBall_normGauss.R' 'unitBall_polynomial.R' 'unitCube_BFN1.R' 'unitCube_BFN2.R' 'unitCube_BFN3.R' 'unitCube_BFN4.R' 'unitCube_Genz1.R' 'unitCube_cos2.R' 'unitCube_floor.R' 'unitCube_max.R' 'unitSphere_innerProduct1.R' 'unitSphere_polynomial.R'

NeedsCompilation no

Author Klaus Herrmann [aut, cre] (<https://orcid.org/0000-0002-8044-5717>)

Repository CRAN

Date/Publication 2024-09-07 13:40:02 UTC

Contents

checkClosedUnitBall *Domain check for closed unit ball* $\{\vec{x} \in R\hat{}n : \|\vec{x}\|\_2 \leq 1\}$

Description

The function checks if a point (one row in the input argument) is inside the closed unit ball $\{\vec{x} \in R^n : \|\vec{x}\|_2 \leq 1\}$ or not. If the input matrix contains entries that are not numeric, i.e., not representing real numbers, the function throws an error. The dimension n is automatically inferred from the input matrix and is equal to the number of columns.

Usage

```
checkClosedUnitBall(x)
```

Arguments

x	Matrix with numeric entries. Each row represents one point

Value

Vector where each element (TRUE or FALSE) indicates if a point is in the closed unit ball

Author(s)

Klaus Herrmann

Examples

```
x <- matrix(rnorm(30),10,3)
checkClosedUnitBall(x)
```

checkClosedUnitCube *Domain check for closed unit hypercube* $[0,1]\hat{}n$

Description

The function checks if a point (one row in the input argument) is inside the closed unit hypercube $[0,1]^n$ or not. If the input matrix contains entries that are not numeric, i.e., not representing real numbers, the function throws an error. The dimension n is automatically inferred from the input matrix and is equal to the number of columns.

Usage

```
checkClosedUnitCube(x)
```

Arguments

x	Matrix with numeric entries. Each row represents one point

Value

Vector where each element (TRUE or FALSE) indicates if a point is in the unit hypercube

Author(s)

Klaus Herrmann

Examples

```
x <- matrix(rnorm(30),10,3)
checkClosedUnitCube(x)
```

---

checkPos *Domain check for* $[0,\infty)\hat{}n$

Description

The function checks if a point (one row in the input argument) is inside $[0,\infty)^n = \times_{i=1}^{n}[0,\infty)$ or not. In this case the return values are all TRUE. If the input matrix contains entries that are not numeric, i.e., not representing real numbers, the function throws an error. The dimension n is automatically inferred from the input matrix and is equal to the number of columns.

Usage

```
checkPos(x)
```

Arguments

x	Matrix with numeric entries. Each row represents one point

Value

Vector where each element (TRUE or FALSE) indicates if a point is in $[0,\infty)^n$

Author(s)

Klaus Herrmann

Examples

```
x <- matrix(rexp(30,rate=1),10,3)
checkPos(x)
```

checkRn *Domain check for $R\hat{}n$*

Description

The function checks if a point (one row in the input argument) is inside the n-dimensional Euclidean space $R^n = \times_{i=1}^{n} R$ or not. In this case the return values are all TRUE. If the input matrix contains entries that are not numeric, i.e., not representing real numbers, the function throws an error. The dimension n is automatically inferred from the input matrix and is equal to the number of columns.

Usage

```
checkRn(x)
```

Arguments

x	Matrix with numeric entries. Each row represents one point

Value

Vector where each element (TRUE or FALSE) indicates if a point is in R^n

Author(s)

Klaus Herrmann

Examples

```
x <- matrix(rnorm(30),10,3)
checkRn(x)
```

checkStandardSimplex *Domain check for standard simplex $\{\vec{x} \in R\hat{}n : x\_i \geq 0, \|\vec{x}\|\_1 \leq 1\}$*

Description

The function checks if a point (one row in the input argument) is inside the standard simplex $\{\vec{x} \in R^n : x_i \geq, \|\vec{x}\|_1 \leq 1\}$ or not. If the input matrix contains entries that are not numeric, i.e., not representing real numbers, the function throws an error. The dimension n is automatically inferred from the input matrix and is equal to the number of columns.

Usage

```
checkStandardSimplex(x)
```

Arguments

x	Matrix with numeric entries. Each row represents one point

Value

Vector where each element (TRUE or FALSE) indicates if a point is in the standard simplex

Author(s)

Klaus Herrmann

Examples

```
x <- matrix(rnorm(30),10,3)
checkStandardSimplex(x)
```

---

checkUnitSphere *Domain check for unit sphere* $\{\vec{x} \in R\hat{}n : \|\vec{x}\|\_2 = 1\}$

Description

The function checks if a point (one row in the input argument) is inside the unit sphere $\{\vec{x} \in R^n : \|\vec{x}\|_2 = 1\}$ or not. If the input matrix contains entries that are not numeric, i.e., not representing real numbers, the function throws an error. The dimension n is automatically inferred from the input matrix and is equal to the number of columns. The function allows for an additional parameter $\varepsilon \geq 0$ to test $\{\vec{x} \in R^n : 1 - \varepsilon \leq \|\vec{x}\|_2 \leq 1 + \varepsilon\}$. WARNING: Due to floating point arithmetic the default value of $\varepsilon = 0$ will not work properly in most cases.

Usage

```
checkUnitSphere(x, eps = 0)
```

Arguments

x	Matrix with numeric entries. Each row represents one point
eps	Non-negative numeric that allows to test points with an additional tolerance

Value

Vector where each element (TRUE or FALSE) indicates if a point is in the unit sphere

Author(s)

Klaus Herrmann

Examples

```
x <- matrix(rnorm(30),10,3)
checkUnitSphere(x,eps=0.001)
```

domainCheck *Check if node points are in the domain of a test function instance*

Description

domainCheck is a generic function that allows to test if a collection of evaluation points are inside the integration domain associated to the test function instance or not.

Usage

```
domainCheck(object, x)

## S4 method for signature 'Pn_lognormalDensity,matrix'
domainCheck(object, x)

## S4 method for signature 'Pn_logtDensity,matrix'
domainCheck(object, x)

## S4 method for signature 'Rn_Gauss,matrix'
domainCheck(object, x)

## S4 method for signature 'Rn_floorNorm,matrix'
domainCheck(object, x)

## S4 method for signature 'Rn_normalDensity,matrix'
domainCheck(object, x)

## S4 method for signature 'Rn_tDensity,matrix'
domainCheck(object, x)

## S4 method for signature 'standardSimplex_Dirichlet,matrix'
domainCheck(object, x)

## S4 method for signature 'standardSimplex_exp_sum,matrix'
domainCheck(object, x)

## S4 method for signature 'unitBall_normGauss,matrix'
domainCheck(object, x)

## S4 method for signature 'unitBall_polynomial,matrix'
domainCheck(object, x)

## S4 method for signature 'unitCube_BFN1,matrix'
domainCheck(object, x)

## S4 method for signature 'unitCube_BFN2,matrix'
domainCheck(object, x)
```

```
## S4 method for signature 'unitCube_BFN3,matrix'
domainCheck(object, x)

## S4 method for signature 'unitCube_BFN4,matrix'
domainCheck(object, x)

## S4 method for signature 'unitCube_Genz1,matrix'
domainCheck(object, x)

## S4 method for signature 'unitCube_cos2,matrix'
domainCheck(object, x)

## S4 method for signature 'unitCube_floor,matrix'
domainCheck(object, x)

## S4 method for signature 'unitCube_max,matrix'
domainCheck(object, x)

## S4 method for signature 'unitSphere_innerProduct1,matrix'
domainCheck(object, x)

## S4 method for signature 'unitSphere_polynomial,matrix'
domainCheck(object, x)
```

Arguments

object	Test function that gets evaluated
x	Matrix where each row represents one evaluation point

Value

Vector where each element (TRUE or FALSE) indicates if a point (row in the input matrix) is in the integration domain

Author(s)

Klaus Herrmann

---

domainCheckP *Check if node points are in the domain of a test function instance ("overload" of domainCheck with additional parameter)*

Description

domainCheckP is a generic function that allows to test if a collection of evaluation points are inside the integration domain associated to the test function instance or not. This "overload" of domainCheck allows to pass a list of additional parameters.

Usage

```
domainCheckP(object, x, param)

## S4 method for signature 'unitSphere_innerProduct1,matrix,list'
domainCheckP(object, x, param)

## S4 method for signature 'unitSphere_polynomial,matrix,list'
domainCheckP(object, x, param)
```

Arguments

object	Test function that gets evaluated
x	Matrix where each row represents one evaluation point
param	List of additional parameters

Value

Vector where each element (TRUE or FALSE) indicates if a point (row in the input matrix) is in the integration domain

Author(s)

Klaus Herrmann

---

evaluate *Evaluate test function instance for a set of node points*

Description

evaluate is a generic function that evaluates the test function instance for a collection of evaluation points represented by a matrix. Each row is one evaluation point.

Usage

```
evaluate(object, x)

## S4 method for signature 'Pn_lognormalDensity,matrix'
evaluate(object, x)

## S4 method for signature 'Pn_logtDensity,ANY'
evaluate(object, x)

## S4 method for signature 'Rn_Gauss,matrix'
evaluate(object, x)

## S4 method for signature 'Rn_floorNorm,matrix'
```

```
evaluate(object, x)

## S4 method for signature 'Rn_normalDensity,matrix'
evaluate(object, x)

## S4 method for signature 'Rn_tDensity,ANY'
evaluate(object, x)

## S4 method for signature 'standardSimplex_Dirichlet,matrix'
evaluate(object, x)

## S4 method for signature 'standardSimplex_exp_sum,matrix'
evaluate(object, x)

## S4 method for signature 'unitBall_normGauss,matrix'
evaluate(object, x)

## S4 method for signature 'unitBall_polynomial,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_BFN1,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_BFN2,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_BFN3,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_BFN4,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_Genz1,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_cos2,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_floor,matrix'
evaluate(object, x)

## S4 method for signature 'unitCube_max,matrix'
evaluate(object, x)

## S4 method for signature 'unitSphere_innerProduct1,matrix'
evaluate(object, x)

## S4 method for signature 'unitSphere_polynomial,matrix'
```

```
evaluate(object, x)
```

Arguments

object	Test function that gets evaluated
x	Matrix where each row represents one evaluation point

Value

Vector where each element is an evaluation of the test function for a node point (row in x)

Author(s)

Klaus Herrmann

---

exactIntegral *Get exact integral for test function instance*

Description

exactIntegral is a generic function that allows to calculate the exact value of a test function instance over the associated integration domain.

Usage

```
exactIntegral(object)

## S4 method for signature 'Pn_lognormalDensity'
exactIntegral(object)

## S4 method for signature 'Pn_logtDensity'
exactIntegral(object)

## S4 method for signature 'Rn_Gauss'
exactIntegral(object)

## S4 method for signature 'Rn_floorNorm'
exactIntegral(object)

## S4 method for signature 'Rn_normalDensity'
exactIntegral(object)

## S4 method for signature 'Rn_tDensity'
exactIntegral(object)

## S4 method for signature 'standardSimplex_Dirichlet'
exactIntegral(object)
```

```
## S4 method for signature 'standardSimplex_exp_sum'
exactIntegral(object)

## S4 method for signature 'unitBall_normGauss'
exactIntegral(object)

## S4 method for signature 'unitBall_polynomial'
exactIntegral(object)

## S4 method for signature 'unitCube_BFN1'
exactIntegral(object)

## S4 method for signature 'unitCube_BFN2'
exactIntegral(object)

## S4 method for signature 'unitCube_BFN3'
exactIntegral(object)

## S4 method for signature 'unitCube_BFN4'
exactIntegral(object)

## S4 method for signature 'unitCube_Genz1'
exactIntegral(object)

## S4 method for signature 'unitCube_cos2'
exactIntegral(object)

## S4 method for signature 'unitCube_floor'
exactIntegral(object)

## S4 method for signature 'unitCube_max'
exactIntegral(object)

## S4 method for signature 'unitSphere_innerProduct1'
exactIntegral(object)

## S4 method for signature 'unitSphere_polynomial'
exactIntegral(object)
```

Arguments

`object`	The test function that gets evaluated

Value

Numeric value of the integral of the test function

Author(s)

Klaus Herrmann

`getIntegrationDomain` *Get description of integration domain for test function instance*

Description

getIntegrationDomain is a generic function that returns a description of the integration domain associate to the test function instance.

Usage

```
getIntegrationDomain(object)

## S4 method for signature 'Pn_lognormalDensity'
getIntegrationDomain(object)

## S4 method for signature 'Pn_logtDensity'
getIntegrationDomain(object)

## S4 method for signature 'Rn_Gauss'
getIntegrationDomain(object)

## S4 method for signature 'Rn_floorNorm'
getIntegrationDomain(object)

## S4 method for signature 'Rn_normalDensity'
getIntegrationDomain(object)

## S4 method for signature 'Rn_tDensity'
getIntegrationDomain(object)

## S4 method for signature 'standardSimplex_Dirichlet'
getIntegrationDomain(object)

## S4 method for signature 'standardSimplex_exp_sum'
getIntegrationDomain(object)

## S4 method for signature 'unitBall_normGauss'
getIntegrationDomain(object)

## S4 method for signature 'unitBall_polynomial'
getIntegrationDomain(object)

## S4 method for signature 'unitCube_BFN1'
```

```
getIntegrationDomain(object)

## S4 method for signature 'unitCube_BFN2'
getIntegrationDomain(object)

## S4 method for signature 'unitCube_BFN3'
getIntegrationDomain(object)

## S4 method for signature 'unitCube_BFN4'
getIntegrationDomain(object)

## S4 method for signature 'unitCube_Genz1'
getIntegrationDomain(object)

## S4 method for signature 'unitCube_cos2'
getIntegrationDomain(object)

## S4 method for signature 'unitCube_floor'
getIntegrationDomain(object)

## S4 method for signature 'unitCube_max'
getIntegrationDomain(object)

## S4 method for signature 'unitSphere_innerProduct1'
getIntegrationDomain(object)

## S4 method for signature 'unitSphere_polynomial'
getIntegrationDomain(object)
```

Arguments

`object`	Test function for which the description is returned

Value

Description of the integration domain of the function

Author(s)

Klaus Herrmann

`getReferences` *Get references for test function instance*

Description

getReferences is a generic function that returns a vector of references associated to the test function instance.

Usage

```
getReferences(object)

## S4 method for signature 'Pn_lognormalDensity'
getReferences(object)

## S4 method for signature 'Pn_logtDensity'
getReferences(object)

## S4 method for signature 'Rn_Gauss'
getReferences(object)

## S4 method for signature 'Rn_floorNorm'
getReferences(object)

## S4 method for signature 'Rn_normalDensity'
getReferences(object)

## S4 method for signature 'Rn_tDensity'
getReferences(object)

## S4 method for signature 'standardSimplex_Dirichlet'
getReferences(object)

## S4 method for signature 'standardSimplex_exp_sum'
getReferences(object)

## S4 method for signature 'unitBall_normGauss'
getReferences(object)

## S4 method for signature 'unitBall_polynomial'
getReferences(object)

## S4 method for signature 'unitCube_BFN1'
getReferences(object)

## S4 method for signature 'unitCube_BFN2'
getReferences(object)

## S4 method for signature 'unitCube_BFN3'
getReferences(object)

## S4 method for signature 'unitCube_BFN4'
getReferences(object)

## S4 method for signature 'unitCube_Genz1'
getReferences(object)
```

```
## S4 method for signature 'unitCube_cos2'
getReferences(object)

## S4 method for signature 'unitCube_floor'
getReferences(object)

## S4 method for signature 'unitCube_max'
getReferences(object)

## S4 method for signature 'unitSphere_innerProduct1'
getReferences(object)

## S4 method for signature 'unitSphere_polynomial'
getReferences(object)
```

Arguments

object	Test function for which the references are returned

Value

Vector with references for the specific function

Author(s)

Klaus Herrmann

---

getTags *Get tags for test function instance*

Description

getTags is a generic function that returns a vector of tags associated to the test function instance.

Usage

```
getTags(object)

## S4 method for signature 'Pn_lognormalDensity'
getTags(object)

## S4 method for signature 'Pn_logtDensity'
getTags(object)

## S4 method for signature 'Rn_Gauss'
getTags(object)

## S4 method for signature 'Rn_floorNorm'
```

```
getTags(object)

## S4 method for signature 'Rn_normalDensity'
getTags(object)

## S4 method for signature 'Rn_tDensity'
getTags(object)

## S4 method for signature 'standardSimplex_Dirichlet'
getTags(object)

## S4 method for signature 'standardSimplex_exp_sum'
getTags(object)

## S4 method for signature 'unitBall_normGauss'
getTags(object)

## S4 method for signature 'unitBall_polynomial'
getTags(object)

## S4 method for signature 'unitCube_BFN1'
getTags(object)

## S4 method for signature 'unitCube_BFN2'
getTags(object)

## S4 method for signature 'unitCube_BFN3'
getTags(object)

## S4 method for signature 'unitCube_BFN4'
getTags(object)

## S4 method for signature 'unitCube_Genz1'
getTags(object)

## S4 method for signature 'unitCube_cos2'
getTags(object)

## S4 method for signature 'unitCube_floor'
getTags(object)

## S4 method for signature 'unitCube_max'
getTags(object)

## S4 method for signature 'unitSphere_innerProduct1'
getTags(object)

## S4 method for signature 'unitSphere_polynomial'
```

```
getTags(object)
```

Arguments

object	Test function for which the tags are returned

Value

Vector with tags related to the function

Author(s)

Klaus Herrmann

---

multIntTestFunc — *multIntTestFunc: A package to define test functions for multivariate numerical integration.*

Description

The multIntTestFunc package provides multivariate test functions to test numerical integration routines. The functions are available in all dimensions n>=1 and the exact value of the integral is known.

multIntTestFunc functions

The multIntTestFunc functions are S4 classes that are instantiated with the dimension and additional parameters if necessary. They implement methods to evaluate the function for given evaluation points (arranged in a matrix) and to evaluate the exact value of the integral of the function over the respective integration domain.

---

pIntRule — *Product rule for numerical quadrature from univariate nodes and weights*

Description

The function allows to build a multivariate quadrature rule from univariate ones. The multivariate node points are all possible combinations of the univariate node points, and the final weights are the product of the respective univariate weights.

Usage

```
pIntRule(x, dim = NULL)
```

Arguments

x	Either a list with two elements $nodes and $weights representing a one dimensional quadrature formula which are then used for all dimensions, or a list where each element is a itself a list with two elements $nodes and $weights. In this case the respective quadrature rule is used for each dimension.
dim	An integer that defines the dimension of the output quadrature formula. Default is NULL. If dim is NULL then x has to be a list of quadrature rules (list of lists) and the dimensions is automatically generated. If dim is a positive integer value the same quadrature rule is used in all dimensions.

Value

A list with a matrix of multivariate node points (each row is one point) and a vector of corresponding weights

Author(s)

Klaus Herrmann

Examples

```
require(statmod)
herm <- gauss.quad(2,"hermite")
lag <- gauss.quad(3,"laguerre")
qRule1 <- pIntRule(herm,2)
qRule2 <- pIntRule(list(herm,lag))
```

Pn_lognormalDensity-class

An S4 class to represent the function $\frac{1}{(\prod\_i{=}1\hat{}nx\_i)\sqrt{(2\pi)\hat{}n\det(\Sigma)}}\exp(-((\ln(\vec{x}) - \vec{\mu})\hat{}T\Sigma\hat{}{-}1(\ln(\vec{x}) - \vec{\mu}))/2)$ *on* $[0,\infty)\hat{}n$

Description

Implementation of the function

$$f\colon R^n \to [0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \frac{1}{(\prod_{i=1}^{n} x_i)\sqrt{(2\pi)^n \det(\Sigma)}} \exp(-((\ln(\vec{x})-\vec{\mu})^T \Sigma^{-1} (\ln(\vec{x})-\vec{\mu}))/2),$$

where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $[0,\infty)^n = \times_{i=1}^{n}[0,\infty)$. In this case the integral is know to be

$$\int_{R^n} f(\vec{x}) d\vec{x} = 1.$$

Details

The instance needs to be created with three parameters representing the dimension n, the location vector $\vec{\mu}$ and the variance-covariance matrix Σ which needs to be symmetric positive definite.

Slots

`dim` An integer that captures the dimension

`mean` A vector of size dim with real entries.

`sigma` A matrix of size dim x dim that is symmetric positive definite.

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("Pn_lognormalDensity",dim=n,mean=rep(0,n),sigma=diag(n))
```

`Pn_logtDensity-class` *An S4 class to represent the function* $(\prod\_i = 1\hat{}nx\_i\hat{}-1)\frac{\Gamma[(\nu+n)/2]}{\Gamma(\nu/2)\nu\hat{}n/2\pi\hat{}n/2|\Sigma|\hat{}1/2}\left[1+\frac{1}{\nu}(\log(\vec{x})-\vec{\delta})\hat{}T\Sigma\hat{}-1(\log(\vec{x})-\vec{\delta})\right]\hat{}-(\nu+$ *on* $[0,\infty)\hat{}n$

Description

Implementation of the function

$$f\colon [0,\infty)^n \to (0,\infty),\ \vec{x} \mapsto f(\vec{x}) = (\prod_{i=1}^{n} x_i^{-1})\frac{\Gamma\left[(\nu+n)/2\right]}{\Gamma(\nu/2)\nu^{n/2}\pi^{n/2}\left|\Sigma\right|^{1/2}}\left[1+\frac{1}{\nu}(\log(\vec{x})-\vec{\delta})^T\Sigma^{-1}(\log(\vec{x})-\vec{\delta})\right]^{-(\nu+n)/2},$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $[0,\infty)^n = \times_{i=1}^{n}[0,\infty)$. In this case the integral is know to be

$$\int_{[0,\infty)^n} f(\vec{x})d\vec{x} = 1.$$

Details

The instance needs to be created with four parameters representing the dimension n, the location vector $\vec{\delta}$, the variance-covariance matrix Σ which needs to be symmetric positive definite and the degrees of freedom parameter ν.

Slots

`dim` An integer that captures the dimension

`delta` A vector of size dim with real entries.

`sigma` A matrix of size dim x dim that is symmetric positive definite.

`df` A positive numerical value representing the degrees of freedom.

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("Pn_logtDensity",dim=n,delta=rep(0,n),sigma=diag(n),df=3)
```

Rn_floorNorm-class *An S4 class to represent the function* $\frac{\Gamma(n/2+1)}{\pi\hat{}n/2(1+\lfloor\|\vec{x}\|\_2\hat{}n\rfloor)\hat{}s}$ *on* $R\hat{}n$

Description

Implementation of the function

$$f\colon R^n \to [0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \frac{\Gamma(n/2+1)}{\pi^{n/2}(1+\lfloor\|\vec{x}\|_2^n\rfloor)^s},$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $R^n = \times_{i=1}^n R$ and $s > 1$ is a parameter. In this case the integral is know to be

$$\int_{R^n} f(\vec{x})d\vec{x} = \zeta(s),$$

where $\zeta(s)$ is the Riemann zeta function.

Details

The instance needs to be created with two parameters representing n and s.

Slots

`dim` An integer that captures the dimension

`s` A numeric value bigger than 1 representing a power

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("Rn_floorNorm",dim=n,s=2)
```

`Rn_Gauss-class` *An S4 class to represent the function* $\exp(-\vec{x}\cdot\vec{x})$ *on* $R\hat{}n$

Description

Implementation of the function

$$f\colon R^n \to (0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \exp(-\vec{x}\cdot\vec{x}) = \exp(-\sum_{i=1}^{n} x_i^2),$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $R^n = \times_{i=1}^{n} R$. In this case the integral is know to be

$$\int_{R^n} f(\vec{x})d\vec{x} = \pi^{n/2}.$$

Details

The instance needs to be created with one parameter representing n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("Rn_Gauss",dim=n)
```

`Rn_normalDensity-class`
An S4 class to represent the function $\frac{1}{\sqrt{(2\pi)\hat{}n\det(\Sigma)}}\exp(-((\vec{x}-\vec{\mu})\hat{}T\Sigma\hat{}-1(\vec{x}-\vec{\mu}))/2)$ *on* $R\hat{}n$

Description

Implementation of the function

$$f\colon R^n \to (0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \frac{1}{\sqrt{(2\pi)^n\det(\Sigma)}}\exp(-((\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu}))/2),$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $R^n = \times_{i=1}^{n} R$. In this case the integral is know to be

$$\int_{R^n} f(\vec{x})d\vec{x} = 1.$$

Details

The instance needs to be created with three parameters representing the dimension n, the location vector $\vec{\mu}$ and the variance-covariance matrix Σ which needs to be symmetric positive definite.

Slots

`dim` An integer that captures the dimension

`mean` A vector of size dim with real entries.

`sigma` A matrix of size dim x dim that is symmetric positive definite.

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("Rn_normalDensity",dim=n,mean=rep(0,n),sigma=diag(n))
```

`Rn_tDensity-class` *An S4 class to represent the function* $\frac{\Gamma[(\nu+n)/2]}{\Gamma(\nu/2)\nu\hat{}n/2\pi\hat{}n/2|\Sigma|\hat{}1/2}\left[1+\frac{1}{\nu}(\vec{x}-\vec{\delta})\hat{}T\Sigma\hat{}-1(\vec{x}-\vec{\delta})\right]\hat{}-(\nu+n)/2$ *on* $R\hat{}n$

Description

Implementation of the function

$$f\colon R^n \to (0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \frac{\Gamma\left[(\nu+n)/2\right]}{\Gamma(\nu/2)\nu^{n/2}\pi^{n/2}\left|\Sigma\right|^{1/2}}\left[1+\frac{1}{\nu}(\vec{x}-\vec{\delta})^T\Sigma^{-1}(\vec{x}-\vec{\delta})\right]^{-(\nu+n)/2},$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $R^n = \times_{i=1}^{n} R$. In this case the integral is know to be

$$\int_{R^n} f(\vec{x})d\vec{x} = 1.$$

Details

The instance needs to be created with four parameters representing the dimension n, the location vector $\vec{\delta}$, the variance-covariance matrix Σ which needs to be symmetric positive definite and the degrees of freedom parameter ν.

Slots

`dim` An integer that captures the dimension

`delta` A vector of size dim with real entries.

`sigma` A matrix of size dim x dim that is symmetric positive definite.

`df` A positive numerical value representing the degrees of freedom.

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("Rn_tDensity",dim=n,delta=rep(0,n),sigma=diag(n),df=3)
```

`standardSimplex_Dirichlet-class`

An S4 class to represent the function $\prod\_i = 1\hat{}nx\_i\hat{}v\_i - 1(1 - x\_1 - \ldots - x\_n)\hat{}v\_n + 1 - 1$ *on* T_n

---

Description

Implementation of the function

$$f\colon T_n \to (0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \prod_{i=1}^{n} x_i^{v_i-1}(1 - x_1 - \ldots - x_n)^{v_{n+1}-1},$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $T_n = \{\vec{x} \in \mathsf{R}^n : x_i \geq 0, \|\vec{x}\|_1 \leq 1\}$ and $v_i > 0$, $i = 1,\ldots,n+1$, are constants. The integral is known to be

$$\int_{T_n} f(\vec{x})d\vec{x} = \frac{\prod_{i=1}^{n+1}\Gamma(v_i)}{\Gamma(\sum_{i=1}^{n+1} v_i)},$$

where $v_i > 0$ for $i = 1,\ldots,n+1$.

Details

The instance needs to be created with two parameters representing the dimension n and the vector of positive parameters.

Slots

`dim` An integer that captures the dimension

`v` A vector of dimension $n+1$ with positive entries representing the constants

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("standardSimplex_Dirichlet",dim=n,v=c(1,2,3,4))
```

`standardSimplex_exp_sum-class` *An S4 class to represent the function* $\exp(-c(x\_1 + \ldots + x\_n))$ *on* T_n

---

Description

Implementation of the function

$$f\colon T_n \to (0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \exp(-c(x_1 + \ldots + x_n)),$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $T_n = \{\vec{x} \in \mathsf{R}^n : x_i \geq 0, \|\vec{x}\|_1 \leq 1\}$ and $c > 0$ is a constant. The integral is known to be

$$\int_{T_n} f(\vec{x}) d\vec{x} = \frac{\Gamma(n) - \Gamma(n,c)}{\Gamma(n) c^n},$$

where $\Gamma(s,x)$ is the incomplete gamma function.

Details

The instance needs to be created with two parameters representing the dimension n and the parameter $c > 0$.

Slots

`dim` An integer that captures the dimension

`coeff` A strictly positive number representing the constant

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("standardSimplex_exp_sum",dim=n,coeff=1)
```

`unitBall_normGauss-class`

An S4 class to represent the function $\frac{1}{(2\pi)\hat{}n/2}\exp(-\|\vec{x}\|\_2\hat{}2/2)$ *on* $B\hat{}n$

Description

Implementation of the function

$$f\colon B_n \to [0,\infty),\ \vec{x} \mapsto f(\vec{x}) = \frac{1}{(2\pi)^{n/2}}\exp(-\|\vec{x}\|_2^2/2) = \frac{1}{(2\pi)^{n/2}}\exp(-\frac{1}{2}\sum_{i=1}^{n} x_i^2),$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $B_n = \{\vec{x} \in R^n : \|\vec{x}\|_2 \leq 1\}$. In this case the integral is know to be

$$\int_{B_n} f(\vec{x})d\vec{x} = P[Z \leq 1] = F_{\chi^2_n}(1),$$

where Z follows a chisquare distribution with n degrees of freedom.

Details

The instance needs to be created with one parameter representing n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitBall_normGauss",dim=n)
```

`unitBall_polynomial-class`

An S4 class to represent the function $\prod\_i = 1\hat{}nx\_i\hat{}a\_i$ *on* B_n

---

Description

Implementation of the function

$$f\colon B_n \to R,\ \vec{x} \mapsto f(\vec{x}) = \prod_{i=1}^{n} x_i^{a_i},$$

where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $B_n = \{\vec{x} \in R^n : \|\vec{x}\|_2 \leq 1\}$ and $a_i \in \{0, 1, 2, 3, \ldots\}$, $i = 1, \ldots, n$, are parameters. If at least one of the coefficients a_i is odd, i.e., $a_i \in \{1, 3, 5, 7, \ldots\}$ for at leas one $i = 1, \ldots, n$, the integral is zero, otherwise the integral is known to be

$$\int_{B_n} f(\vec{x}) d\vec{x} = 2 \frac{\prod_{i=1}^{n} \Gamma(b_i)}{\Gamma(\sum_{i=1}^{n} b_i)(n + \sum_{i=1}^{n} a_i)},$$

where $b_i = (a_i + 1)/2$.

Details

The instance needs to be created with two parameters representing the dimension n and a n-dimensional vector of integers (including 0) representing the exponents.

Slots

`dim` An integer that captures the dimension

`expo` An vector that captures the exponents

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitBall_polynomial",dim=n,expo=c(1,2,3))
```

`unitCube_BFN1-class` *An S4 class to represent the function* $\prod \hat{}n\_i = 1|4x\_i-2|$ *on* $[0,1]\hat{}n$

Description

Implementation of the function

$$f\colon [0,1]^n \to (-\infty, \infty),\ \vec{x} \mapsto f(\vec{x}) = \prod_{i=1}^{n} |4x_i - 2|$$

, where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $C_n = [0, 1]^n$. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = 1.$$

Details

The instance needs to be created with one parameter representing the dimension n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitCube_BFN1",dim=n)
```

`unitCube_BFN2-class` *An S4 class to represent the function* $\prod \hat{} n\_i = 1 i \cos(i x\_i)$ *on* $[0,1] \hat{} n$

Description

Implementation of the function

$$f\colon [0,1]^n \to (-\infty, \infty),\ \vec{x} \mapsto f(\vec{x}) = \prod_{i=1}^{n} i \cos(i x_i)$$

, where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $C_n = [0,1]^n$. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = \prod_{i=1}^{n} \sin(i).$$

Details

The instance needs to be created with one parameter representing the dimension n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitCube_BFN2",dim=n)
```

`unitCube_BFN3-class` *An S4 class to represent the function* $\prod \hat{} n\_i = 1T\_\nu(i)(2x\_i - 1)$ *on* $[0,1]\hat{} n$

Description

Implementation of the function

$$f\colon [0,1]^n \to (-\infty, \infty),\ \vec{x} \mapsto f(\vec{x}) = \prod_{i=1}^{n} T_{\nu(i)}(2x_i - 1)$$

, where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $C_n = [0,1]^n$ and T_k is the Chebyshev polynomial of degree k and $\nu(i) = (i \mod 4) + 1$. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = 0.$$

Details

The instance needs to be created with one parameter representing the dimension n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitCube_BFN3",dim=n)
```

`unitCube_BFN4-class` *An S4 class to represent the function* $\sum \_i = 1\hat{} n(-1)\hat{} i \prod \_j = 1\hat{} i x\_j$ *on* $[0,1]\hat{} n$

Description

Implementation of the function

$$f\colon [0,1]^n \to (-\infty, \infty),\ \vec{x} \mapsto f(\vec{x}) = \sum_{i=1}^{n} (-1)^i \prod_{j=1}^{i} x_j$$

, where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $C_n = [0,1]^n$. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = -(1 - (-1/2)^n)/3.$$

Details

The instance needs to be created with one parameter representing the dimension n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitCube_BFN4",dim=n)
```

`unitCube_cos2-class` *An S4 class to represent the function* $(\cos(\vec{x} \cdot \vec{v}))\hat{}2$ *on* $[0,1]\hat{}n$

Description

Implementation of the function

$$f\colon [0,1]^n \to [0,1],\ \vec{x} \mapsto f(\vec{x}) = (\cos(\vec{x} \cdot \vec{v}))^2,$$

where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $C_n = [0,1]^n$ and $\vec{v}$ is a n-dimensional parameter vector where each entry is different from 0. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = \frac{1}{2} + \frac{1}{2}\cos(\sum_{j=1}^{n} v_j) \prod_{j=1}^{n} \frac{\sin(v_j)}{v_j}.$$

Details

The instance needs to be created with two parameters representing the dimension n and the n-dimensional parameter vector where each entry is different from 0.

Slots

`dim` An integer that captures the dimension

`coeffs` A vector of non-zero parameters

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitCube_cos2",dim=n, coeffs=c(-1,2,-2))
```

unitCube_floor-class *An S4 class to represent the function* $\lfloor x\_1 + \ldots + x\_n \rfloor$ *on* $[0,1]\hat{}n$

Description

Implementation of the function

$$f\colon [0,1]^n \to [0,n],\ \vec{x} \mapsto f(\vec{x}) = \lfloor x_1 + \ldots + x_n \rfloor,$$

where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $C_n = [0,1]^n$. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = \frac{n-1}{2}.$$

Details

The instance needs to be created with one parameter representing the dimension n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitCube_floor",dim=n)
```

unitCube_Genz1-class *An S4 class to represent the function* $\cos(2\pi u + \sum \hat{}n\_i = 1 a\_i x\_i)$ *on* $[0,1]\hat{}n$

Description

Implementation of the function

$$f\colon [0,1]^n \to (-\infty,\infty),\ \vec{x} \mapsto f(\vec{x}) = \cos\left(2\pi u + \sum_{i=1}^{n} a_i x_i\right)$$

, where $n \in \{1,2,3,\ldots\}$ is the dimension of the integration domain $C_n = [0,1]^n$. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = \frac{2^n \cos\left(2\pi u + \sum_{i=1}^{n} a_i/2\right) \prod_{i=1}^{n} \sin(a_i/2)}{\prod_{i=1}^{n} a_i}.$$

Details

The instance needs to be created with three parameter representing the dimension n, the real number u and the vector $(a_1, ..., a_n)$.

Slots

`dim` An integer that captures the dimension

`u` A real number representing a shift in the integrand

`a` A vector of real numbers, each non-zero, increasing the difficulty of the integrand with higher absolute values

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
u <- pi
a <- rep(exp(1),n)
f <- new("unitCube_Genz1",dim=n, u=u, a=a)
```

`unitCube_max-class` *An S4 class to represent the function* $\max(x\_1, \ldots, x\_n)$ *on* $[0,1]\hat{}n$

Description

Implementation of the function

$$f\colon [0,1]^n \to [0,n],\ \vec{x} \mapsto f(\vec{x}) = \max(x_1, \ldots, x_n)$$

, where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $C_n = [0,1]^n$. The integral is known to be

$$\int_{C_n} f(\vec{x}) d\vec{x} = \frac{n}{n+1}.$$

Details

The instance needs to be created with one parameter representing the dimension n.

Slots

`dim` An integer that captures the dimension

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitCube_max",dim=n)
```

---

unitSphere_innerProduct1-class

An S4 class to represent the function $(\vec{x} \cdot \vec{a})(\vec{x} \cdot \vec{b})$ *on* $S\hat{}n - 1$

Description

Implementation of the function

$$f \colon S^{n-1} \to R,\ \vec{x} \mapsto f(\vec{x}) = (\vec{x} \cdot \vec{a})(\vec{x} \cdot \vec{b}),$$

where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $S^{n-1} = \{\vec{x} \in R^n : \|\vec{x}\|_2 = 1\}$ and $\vec{a}$ and $\vec{b}$ are two n-dimensional parameter vectors. The integral is known to be

$$\int_{S^{n-1}} f(\vec{x}) d\vec{x} = \frac{2\pi^{n/2}(\vec{a} \cdot \vec{b})}{n\Gamma(n/2)},$$

where $\vec{a} \in R^n$ and $\vec{b} \in R^n$.

Details

Due to the difficulty of testing $\|\vec{x}\|_2 = 1$ in floating point arithmetic this class also implements the function "domainCheckP". This allows to pass a list with an additional non-negative parameter "eps" representing a non-negative real number ε and allows to test $1 - \varepsilon \leq \|\vec{x}\|_2 \leq 1 + \varepsilon$. See also the documentation of the function "checkUnitSphere" that is used to perform the checks.

The instance needs to be created with three parameters representing the dimension n and the two n-dimensional (real) vectors $\vec{a}$ and $\vec{b}$.

Slots

`dim` An integer that captures the dimension

`a` A n-dimensional real vector

`b` A n-dimensional real vector

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitSphere_innerProduct1",dim=n,a=c(1,2,3),b=c(-1,-2,-3))
```

`unitSphere_polynomial-class` *An S4 class to represent the function* $\prod\_i = 1\hat{}nx\_i\hat{}a\_i$ *on* $S\hat{}n - 1$

Description

Implementation of the function

$$f\colon S^{n-1} \to R,\ \vec{x} \mapsto f(\vec{x}) = \prod_{i=1}^{n} x_i^{a_i},$$

where $n \in \{1, 2, 3, \ldots\}$ is the dimension of the integration domain $S^{n-1} = \{\vec{x} \in R^n : \|\vec{x}\|_2 = 1\}$ and $a_i \in \{0, 1, 2, 3, \ldots\}$, $i = 1, \ldots, n$, are parameters. If at least one of the coefficients a_i is odd, i.e., $a_i \in \{1, 3, 5, 7, \ldots\}$ for at leas one $i = 1, \ldots, n$, the integral is zero, otherwise the integral is known to be

$$\int_{S^{n-1}} f(\vec{x}) d\vec{x} = 2\frac{\prod_{i=1}^{n} \Gamma(b_i)}{\Gamma(\sum_{i=1}^{n} b_i)},$$

where $b_i = (a_i + 1)/2$.

Details

Due to the difficulty of testing $\|\vec{x}\|_2 = 1$ in floating point arithmetic this class also implements the function "domainCheckP". This allows to pass a list with an additional non-negative parameter "eps" representing a non-negative real number ε and allows to test $1 - \varepsilon \leq \|\vec{x}\|_2 \leq 1 + \varepsilon$. See also the documentation of the function "checkUnitSphere" that is used to perform the checks.

The instance needs to be created with two parameters representing the dimension n and a n-dimensional vector of integers (including 0) representing the exponents.

Slots

`dim` An integer that captures the dimension

`expo` An vector that captures the exponents

Author(s)

Klaus Herrmann

Examples

```
n <- as.integer(3)
f <- new("unitSphere_polynomial",dim=n,expo=c(1,2,3))
```

Index