

LinePicking

Generated by Doxygen 1.8.2

Wed Aug 29 2012 17:34:39

Contents

1	LinePicking	1
2	Todo List	3
3	Module Index	5
3.1	Modules	5
4	Data Structure Index	7
4.1	Data Structures	7
5	File Index	9
5.1	File List	9
6	Module Documentation	11
6.1	LinePicking API	11
6.1.1	Detailed Description	11
6.1.2	Function Documentation	11
6.1.2.1	LinePickingCDF	11
6.1.2.2	LinePickingCheckParameters	12
6.1.2.3	LinePickingNumberOfProblems	12
6.1.2.4	LinePickingPDF	12
6.1.2.5	LinePickingSupport	12
6.1.2.6	LinePickingVar	12
7	Data Structure Documentation	13
7.1	LinePickingRec Struct Reference	13
7.1.1	Detailed Description	13
8	File Documentation	15
8.1	beta.h File Reference	15
8.1.1	Detailed Description	15
8.1.2	Function Documentation	15
8.1.2.1	beta	15
8.1.2.2	beta_inc	16

8.2	Cube.h File Reference	16
8.2.1	Detailed Description	17
8.2.2	Function Documentation	17
8.2.2.1	CubeDistanceCDF	17
8.2.2.2	CubeDistanceCheckParameters	17
8.2.2.3	CubeDistanceMean	17
8.2.2.4	CubeDistancePDF	18
8.2.2.5	CubeDistanceSupport	18
8.2.2.6	CubeDistanceVar	18
8.3	Disk.h File Reference	19
8.3.1	Detailed Description	19
8.3.2	Function Documentation	19
8.3.2.1	DiskDistanceCDF	19
8.3.2.2	DiskDistanceCheckParameters	20
8.3.2.3	DiskDistanceMean	20
8.3.2.4	DiskDistancePDF	20
8.3.2.5	DiskDistanceSupport	21
8.3.2.6	DiskDistanceVar	21
8.4	Hyperball.h File Reference	21
8.4.1	Detailed Description	22
8.4.2	Function Documentation	22
8.4.2.1	HyperballDistanceCDF	22
8.4.2.2	HyperballDistanceCheckParameters	22
8.4.2.3	HyperballDistanceMean	23
8.4.2.4	HyperballDistancePDF	23
8.4.2.5	HyperballDistanceSupport	23
8.4.2.6	HyperballDistanceVar	24
8.5	Line.h File Reference	24
8.5.1	Detailed Description	24
8.5.2	Function Documentation	25
8.5.2.1	LineDistanceCDF	25
8.5.2.2	LineDistanceCheckParameters	25
8.5.2.3	LineDistanceMean	25
8.5.2.4	LineDistancePDF	25
8.5.2.5	LineDistanceSupport	26
8.5.2.6	LineDistanceVar	26
8.6	LinePicking.h File Reference	27
8.6.1	Detailed Description	28
8.6.2	Macro Definition Documentation	28
8.6.2.1	ExpandFields	28

8.6.3	Variable Documentation	28
8.6.3.1	LinePickingFields	28
8.7	PrismGeodesic.h File Reference	28
8.7.1	Detailed Description	29
8.7.2	Function Documentation	29
8.7.2.1	PrismGeodesicDistanceCDF	29
8.7.2.2	PrismGeodesicDistanceCheckParameters	29
8.7.2.3	PrismGeodesicDistanceMean	30
8.7.2.4	PrismGeodesicDistancePDF	30
8.7.2.5	PrismGeodesicDistanceSupport	30
8.7.2.6	PrismGeodesicDistanceVar	30
8.8	Rectangle.h File Reference	31
8.8.1	Detailed Description	31
8.8.2	Function Documentation	31
8.8.2.1	RectangleDistanceCDF	31
8.8.2.2	RectangleDistanceCheckParameters	32
8.8.2.3	RectangleDistanceMean	32
8.8.2.4	RectangleDistancePDF	32
8.8.2.5	RectangleDistanceSupport	33
8.8.2.6	RectangleDistanceVar	33
8.9	Sphere.h File Reference	33
8.9.1	Detailed Description	34
8.9.2	Function Documentation	34
8.9.2.1	SphereDistanceCDF	34
8.9.2.2	SphereDistanceCheckParameters	34
8.9.2.3	SphereDistanceMean	35
8.9.2.4	SphereDistancePDF	35
8.9.2.5	SphereDistanceSupport	35
8.9.2.6	SphereDistanceVar	36
8.10	SphereGeodesic.h File Reference	36
8.10.1	Detailed Description	36
8.10.2	Function Documentation	37
8.10.2.1	SphereGeodesicDistanceCDF	37
8.10.2.2	SphereGeodesicDistanceCheckParameters	37
8.10.2.3	SphereGeodesicDistanceMean	37
8.10.2.4	SphereGeodesicDistancePDF	38
8.10.2.5	SphereGeodesicDistanceSupport	38
8.10.2.6	SphereGeodesicDistanceVar	38
8.11	Square.h File Reference	38
8.11.1	Detailed Description	39

8.11.2	Function Documentation	39
8.11.2.1	SquareDistanceCDF	39
8.11.2.2	SquareDistanceCheckParameters	39
8.11.2.3	SquareDistanceMean	40
8.11.2.4	SquareDistancePDF	40
8.11.2.5	SquareDistanceSupport	40
8.11.2.6	SquareDistanceVar	41

Index	41
--------------	-----------

Chapter 1

LinePicking

Numerical code for geometric probability problems, in particular PDFs, CDFs, means and variances for the "line picking" problem. The problem is a standard problem in stochastic geometry, where we pick lines at random from some region. The typical questions one asks are: what will the mean line length be? What will the Probability Density Function (PDF) be? This software implements the current list of known PDFs, CDFs, means and variances for such problems. It also provides solutions to some previously unsolved problems.

The library has been designed to provide a small set of entry points which are callable from R, Matlab and other C programs. Documentation for the R and Matlab bindings to this library have been provided in the a format suitable for each of their help systems.

Much of this manual is dedicated to documenting functions specific to a particular problem but users of the library only need to understand the entry points documented in the [LinePicking API](#).

A simple method for seamlessly extending the library has been provided and is also documented in the [LinePicking API](#).

Authors

Eric Parsonage eric.parsonage@adelaide.edu.au

Matthew Roughan matthew.roughan@adelaide.edu.au

Jonothan Tuke simon.tuke@adelaide.edu.au

Chapter 2

Todo List

Group [api](#)

Add some more preamble about the API.

Global [CubeDistanceCDF](#) (double a, double *b)

Write up how to derive the CDF.

Global [CubeDistanceVar](#) (double *parameters)

Try to derive a value algebraically

Global [DiskDistanceCDF](#) (double a, double *b)

Write up how to derive the CDF.

Global [DiskDistanceVar](#) (double *parameters)

Derive an algebraic expression.

Global [HyperballDistanceCDF](#) (double t, double *parameters)

Implement!

Global [PrismGeodesicDistanceCDF](#) (double t, double *parameters)

Write up the derivation.

Global [PrismGeodesicDistanceMean](#) (double *parameters)

re-derive the mean without using the three part PDF as a basis I am sure a simpler result can be found using a similar methods to that used on the rectangle

Global [PrismGeodesicDistancePDF](#) (double t, double *parameters)

Write up the derivation.

Global [PrismGeodesicDistanceVar](#) (double *parameters)

re-derive the variance without using the three part PDF as a basis I am sure a simpler result can be found using a similar methods to that used on the rectangle

Global [RectangleDistanceCDF](#) (double t, double *parameters)

Write up the derivation.

Global [SphereGeodesicDistanceCDF](#) (double s, double *parameters)

Write up the derivation.

Global [SphereGeodesicDistanceMean](#) (double *parameters)

Write up the derivation.

Global [SphereGeodesicDistancePDF](#) (double s, double *parameters)

Write up the derivation.

Global [SphereGeodesicDistanceVar](#) (double *parameters)

Write up the derivation.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

LinePicking API	11
---------------------------	----

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

LinePickingRec	13
--	----

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

beta.h	Implements the beta function and the regularized incomplete beta function	15
Cube.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points within a cube	16
Disk.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points on a disk	19
Hyperball.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points within a hyper-ball	21
Line.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points on a line	24
LinePicking.h	Exposes to matlab and R a set of functions that implement PDF, CDF, mean and variance of the distance between two random points in various geometries	27
PrismGeodesic.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface (not including the ends) of an upright prism of any cross section. The distance is measured around the surface i.e., it is a geodesic	28
Rectangle.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points on a rectangle	31
Sphere.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface of a sphere	33
SphereGeodesic.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface of a sphere. The distance is measured around the surface of the sphere i.e., it is a geodesic	36
Square.h	Functions to provide PDF, CDF, mean and variance of the distance between two random points on a square	38

Chapter 6

Module Documentation

6.1 LinePicking API

Functions

- void [LinePickingNumberOfProblems](#) (int *)
- void **LinePickingPrintAllProblems** ()
- void **LinePickingAllProblems** (char **, char **)
- void **LinePickingProblemLookup** (int *, char **, char **)
- void [LinePickingCheckParameters](#) (int *, double *, int *, int *, char **)
- void [LinePickingSupport](#) (double *, int *, double *, int *, int *, char **)
- void [LinePickingPDF](#) (double *, double *, int *, int *, double *, int *, int *, char **)
- void [LinePickingCDF](#) (double *, double *, int *, int *, double *, int *, int *, char **)
- void **LinePickingMean** (double *, int *, double *, int *, int *, char **)
- void [LinePickingVar](#) (double *, int *, double *, int *, int *, char **)

6.1.1 Detailed Description

This section documents the API exposed for the use of other programs. These are the functions called by the R and Matlab wrappers included in this package.

Todo Add some more preamble about the API.

6.1.2 Function Documentation

6.1.2.1 void [LinePickingCDF](#) (double * *t*, double * *g*, int * *N*, int * *problem*, double * *parameters*, int * *Npar*, int * *result*, char ** *error_str*)

compute distance distribution function $\int g(t) dt$ (at points *t*) between two points in a region.

t = array of points at which to calculate density *g* = array to store output problem = type of region (see [LinePicking-ProblemLookup](#)) *Npar* = number of parameters *result* = exit code 0: parameters are valid 1: unsupported problem 2: parameters out of range. 3: not enough parameters were entered. 4: other error. *error_str*: a message explaining the error

Note that *N*, *problem* and *Npar* are all passed in by reference so R can cope, and similarly, the function must return void, so we return the exit code in the last argument.

6.1.2.2 void LinePickingCheckParameters (int * *problem*, double * *parameters*, int * *Npar*, int * *result*, char ** *error_str*)

check that a problem and a set of parameters are valid

problem = type of region (see LinePickingProblemLookup) Npar = number of parameters result = exit code 0: parameters are valid 1: unsupported problem 2: parameters out of range. 3: not enough parameters were entered. 4: other error. error_str: a message explaining the error

Note that N, problem and Npar are all passed in by reference so R can cope, and similarly, the function must return void, so we return the exit code in the last argument.

6.1.2.3 void LinePickingNumberOfProblems (int * *N*)

Helper function to expose the number of currently implemented problems.

Parameters

\$N	Pointer to an integer to store the number of currently implemented problems.
-----	--

Returns

The number of currently implemented problems is returned in \$N

6.1.2.4 void LinePickingPDF (double * *t*, double * *g*, int * *N*, int * *problem*, double * *parameters*, int * *Npar*, int * *result*, char ** *error_str*)

compute distance density $g(t)$ (at points t) between two points in a region.

t = array of points at which to calculate density g = array to store output problem = type of region (see LinePickingProblemLookup) Npar = number of parameters result = exit code 0: parameters are valid 1: unsupported problem 2: parameters out of range. 3: not enough parameters were entered. 4: other error. error_str: a message explaining the error

Note that N, problem and Npar are all passed in by reference so R can cope, and similarly, the function must return void, so we return the exit code in the last argument.

6.1.2.5 void LinePickingSupport (double * *t*, int * *problem*, double * *parameters*, int * *Npar*, int * *result*, char ** *error_str*)

compute support of distance density $g(t)$ (at points t) between two points in a region.

$t = [t_{\min}, t_{\max}]$: assumes 2 spaces are allocated!!!! problem = type of region (see LinePickingProblemLookup) Npar = number of parameters result = exit code 0: parameters are valid 1: unsupported problem 2: parameters out of range. 3: not enough parameters were entered. 4: other error. error_str: a message explaining the error

Note that N, problem and Npar are all passed in by reference so R can cope, and similarly, the function must return void, so we return the exit code in the last argument.

6.1.2.6 void LinePickingVar (double * *var*, int * *problem*, double * *parameters*, int * *Npar*, int * *result*, char ** *error_str*)

compute variance between two points in a region.

t = array of points at which to calculate density var = var line length problem = type of region (see LinePickingProblemLookup) Npar = number of parameters result = exit code 0: parameters are valid 1: unsupported problem 2: parameters out of range. 3: not enough parameters were entered. 4: other error. error_str: a message explaining the error

Note that problem and Npar are all passed in by reference so R can cope, and similarly, the function must return void, so we return the exit code in the last argument.

Chapter 7

Data Structure Documentation

7.1 LinePickingRec Struct Reference

```
#include <LinePicking.h>
```

Data Fields

- double(* **PDF**)(double, double *)
- double(* **CDF**)(double, double *)
- double(* **MEAN**)(double *)
- double(* **VAR**)(double *)
- void(* **SUPPORT**)(double *, double *)
- void(* **CHECK_PAR**)(double *, int *, char *)
- int * **Npar**
- char ** **name**
- char ** **description**

7.1.1 Detailed Description

structure thingo test

The documentation for this struct was generated from the following file:

- [LinePicking.h](#)

Chapter 8

File Documentation

8.1 beta.h File Reference

Implements the beta function and the regularized incomplete beta function.

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
```

Macros

- #define **PRINT_STDOUT**(...) Rprintf(__VA_ARGS__)

Functions

- double [beta](#) (double, double)
- double [beta_inc](#) (double, double, double, int *result)

8.1.1 Detailed Description

Implements the beta function and the regularized incomplete beta function.

Author

Matthew Roughan matthew.roughan@adelaide.edu.au

Date

22/09/2012

8.1.2 Function Documentation

8.1.2.1 double beta (double x, double y)

Implements the beta function. i.e., the Euler integral of the first kind, defined by $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1} dt$.

Parameters

x	in $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1} dt$.
y	in $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1} dt$.

Returns

The result of evaluating Eulers integral of the first kind with the given parameters.

8.1.2.2 double beta_inc (double a, double b, double x, int * result)

Implements the regularized incomplete beta function. Defined as $I_x(a, b) = \frac{B(x; a, b)}{B(a, b)}$. Where $B(a, b) = \int_0^1 t^{a-1} (1-t)^{b-1} dt$ and $B(x; a, b) = \int_0^x t^{a-1} (1-t)^{b-1} dt$.

See Also

<http://doi.acm.org/10.1145/131766.131776>
http://www.boost.org/doc/libs/1_38_0/libs/math/doc/sf_and_dist/html/math-toolkit/special/sf_beta/ibeta_function.html
<http://dlmf.nist.gov/8.17>

Parameters

<i>\$a</i>	in $I_x(a, b) = \frac{B(x; a, b)}{B(a, b)}$.
<i>\$b</i>	in $I_x(a, b) = \frac{B(x; a, b)}{B(a, b)}$.
<i>\$x</i>	in $I_x(a, b) = \frac{B(x; a, b)}{B(a, b)}$.
<i>\$result</i>	is non zero if some error occured.

Returns

The result of evaluating the incomplete beta function with the given parameters or an error conditon returned in \$result

8.2 Cube.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points within a cube.

Functions

- double [CubeDistancePDF](#) (double t, double *parameters)
- double [CubeDistanceCDF](#) (double a, double *b)
- double [CubeDistanceMean](#) (double *parameters)
- double [CubeDistanceVar](#) (double *parameters)
- void [CubeDistanceSupport](#) (double *t, double *parameters)
- void [CubeDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **CubeDistanceName**
- char * **CubeDistanceDescription**
- int **CubeDistanceNpar**

8.2.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points within a cube.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.2.2 Function Documentation

8.2.2.1 double CubeDistanceCDF (double *t*, double * *parameters*)

Implements the CDF of the distance between two random points within a cube.

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au

Parameters

<i>\$t</i>	The distance to calculate the cumulative density for.
<i>\$parameters</i>	<i>\$parameters</i> [0] is the size of the cube (i.e., the length of any side).

Todo Write up how to derive the CDF.

Returns

The cumulative density at *\$t*.

8.2.2.2 void CubeDistanceCheckParameters (double * *parameters*, int * *result*, char * *error_str*)

Intended to determine if the parameters supplied are valid input to the other functions implemented in this file. However as there is only one parameter and the calling function checks that it is positive this is merely a place holder to allow for a complete implementation in geometries that have more complex relationships between parameters.

Parameters

<i>\$parameters</i>	<i>\$parameters</i> [0] is the size of the cube (i.e., the length of any side).
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by *\$result* and a message explaining the error is copied in to the location pointed to by *\$error_str*

8.2.2.3 double CubeDistanceMean (double * *parameters*)

Calculates the mean of the distance between two random points within a cube.

From Mathai, A. M.; Moschopoulos, P.; and Pederzoli, G. "Distance between Random Points in a Cube." J. Statistica 59, 61-81, 1999. but with 'corrected typos'

Parameters

<i>\$parameters</i>	\$parameters[0] is the size of the cube (i.e., the length of any side).
---------------------	---

Returns

The mean distance between two points in a unit cube.

See Also

<http://mathworld.wolfram.com/CubeLinePicking.html>

8.2.2.4 double CubeDistancePDF (double *t*, double * *parameters*)

Implements the PDF of the distance between two random points within a cube.

From Mathai, A. M.; Moschopoulos, P.; and Pederzoli, G. "Distance between Random Points in a Cube." J. Statistica 59, 61-81, 1999. but with 'corrected typos'

Parameters

<i>\$t</i>	The distance to calculate the density for.
<i>\$parameters</i>	\$parameters[0] is the size of the cube (i.e., the length of any side).

Returns

The density at *\$t*.

See Also

<http://mathworld.wolfram.com/CubeLinePicking.html>

8.2.2.5 void CubeDistanceSupport (double * *t*, double * *parameters*)

Calculates the support for the PDF and CDF of the distance between two random points within a cube.

Parameters

<i>\$t</i>	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points within a cube.
<i>\$parameters</i>	\$parameters[0] is the size of the cube. (i.e., the length of any side).

Returns

The lower end of the interval is returned in *\$t*[0] and the upper end of the interval is returned in *\$t*[1].

8.2.2.6 double CubeDistanceVar (double * *parameters*)

Calculates the variance of the distance between two random points within a cube.

This was calculated numerically.

Parameters

<i>\$parameters</i>	\$parameters[0] is the size of the cube (i.e., the length of any side).
---------------------	---

Todo Try to derive a value algebaricaly

Returns

The variance of the distances between two points in a unit cube.

8.3 Disk.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a disk.

Functions

- double [DiskDistancePDF](#) (double *t*, double **parameters*)
- double [DiskDistanceCDF](#) (double *a*, double **b*)
- double [DiskDistanceMean](#) (double **parameters*)
- double [DiskDistanceVar](#) (double **parameters*)
- void [DiskDistanceSupport](#) (double **t*, double **parameters*)
- void [DiskDistanceCheckParameters](#) (double **parameters*, int **result*, char **error_str*)

Variables

- char * **DiskDistanceName**
- char * **DiskDistanceDescription**
- int **DiskDistanceNpar**

8.3.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a disk.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.3.2 Function Documentation

8.3.2.1 double DiskDistanceCDF (double *t*, double * *parameters*)

Implements the CDF of the distance between two random points on a disk.

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au

Parameters

<i>\$t</i>	The distance to calculate the cumulative density for.
<i>\$parameters</i>	\$parameters[0] is the radius of the disk.

Todo Write up how to derive the CDF.

Returns

The cumulative density at \$t.

8.3.2.2 void DiskDistanceCheckParameters (double * parameters, int * result, char * error_str)

Intended to determine if the parameters supplied are valid input to the other functions implemented in this file. However as there is only one parameter and the calling function checks that it is positive this is merely a place holder to allow for a complete implementation in geometries that have more complex relationships between parameters.

Parameters

<i>\$parameters</i>	\$parameters[0] is the radius of the disk.
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by \$result and a message explaining the error is copied in to the location pointed to \$error_str

8.3.2.3 double DiskDistanceMean (double * parameters)

Calculates the mean of the distance between two random points on a disk.

From Tu, S.-J. and Fischbach, E. "A New Geometric Probability Technique for an {N}-Dimensional Sphere and Its Applications"

Parameters

<i>\$parameters</i>	\$parameters[0] is the radius of the disk.
---------------------	--

Returns

The mean distance between two points on a disk.

See Also

<http://mathworld.wolfram.com/BallLinePicking.html>

8.3.2.4 double DiskDistancePDF (double t, double * parameters)

Implements the PDF of the distance between two random points on a disk.

From Tu, S.-J. and Fischbach, E. "A New Geometric Probability Technique for an {N}-Dimensional Sphere and Its Applications"

Parameters

<i>\$t</i>	The distance to calculate the density for.
<i>\$parameters[0]</i>	The radius of the disk.

Returns

The density at \$t.

See Also

<http://mathworld.wolfram.com/BallLinePicking.html>

8.3.2.5 void DiskDistanceSupport (double * t, double * parameters)

Calculates the support for the PDF and CDF of the distance between two random points on a disk.

Parameters

\$t	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points on a disk.
\$parameters	\$parameters[0] is the radius of the disk.

Returns

The lower end of the interval is returned in \$t[0] and the upper end of the interval is returned in \$t[1].

8.3.2.6 double DiskDistanceVar (double * parameters)

Calculates the variance of the distances between two random points on a disk.

Currently calculated numerically.

Parameters

\$parameters	\$parameters[0] is the radius of the disk.
--------------	--

Returns

The variance of distances between two points on a disk.

See Also

<http://mathworld.wolfram.com/BallLinePicking.html>

Todo Derive an algebraic expression.

8.4 Hyperball.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points within a hyper-ball.

Functions

- double [HyperballDistancePDF](#) (double t, double *parameters)
- double [HyperballDistanceCDF](#) (double t, double *parameters)
- double [HyperballDistanceMean](#) (double *parameters)
- double [HyperballDistanceVar](#) (double *parameters)
- void [HyperballDistanceSupport](#) (double *t, double *parameters)
- void [HyperballDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **HyperballDistanceName**
- char * **HyperballDistanceDescription**
- int **HyperballDistanceNpar**

8.4.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points within a hyper-ball.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.4.2 Function Documentation

8.4.2.1 double HyperballDistanceCDF (double *a*, double * *b*)

Will implement the CDF of the distance between two random points in a hyper-ball.

Not yet implemented.

Todo Implement!

Parameters

<i>\$t</i>	The distance to calculate the cumulative density for.
<i>\$parameters</i>	<i>\$parameters</i> [0] is the dimension of the hyper-ball and <i>\$parameters</i> [1] is the radius of the hyper-ball.

Returns

The cumulative density at *\$t*, currently -1 for all *\$t*.

8.4.2.2 void HyperballDistanceCheckParameters (double * *parameters*, int * *result*, char * *error_str*)

Determines if the parameters supplied are valid input to the other functions implemented in this file.

Parameters

<i>\$parameters</i>	<i>\$parameters</i> [0] is the dimension of the hyper-ball under consideration and <i>\$parameters</i> [1] is the radius of the hyper-ball under consideration.
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by *\$result* and a message explaining the error is copied in to the location pointed to *\$error_str* The only condition this function needs to check is that the dimesnion of the hyper-ball consider is at least 1.

8.4.2.3 double HyperballDistanceMean (double * *parameters*)

Calculates the mean distance between two random points in a hyper-ball.

From Tu, S.-J. and Fischbach, E. "A New Geometric Probability Technique for an {N}-Dimensional Sphere and Its Applications"

Parameters

<i>\$parameters</i>	\$parameters[0] is the dimension of the hyper-ball and \$parameters[1] is the radius of the hyper-ball.
---------------------	---

Returns

The mean distance between two random points in a hyper-ball.

See Also

<http://arxiv.org/abs/math-ph/0004021>
<http://mathworld.wolfram.com/BallLinePicking.html>

8.4.2.4 double HyperballDistancePDF (double *t*, double * *parameters*)

Implements the PDF of the distance between two random points in a hyper-ball.

From Tu, S.-J. and Fischbach, E. "A New Geometric Probability Technique for an {N}-Dimensional Sphere and Its Applications"

Parameters

<i>\$t</i>	The distance to calculate the density for.
<i>\$parameters</i>	\$parameters[0] is the dimension of the hyper-ball and \$parameters[1] is the radius of the hyper-ball.

Returns

The density at \$t.

See Also

<http://arxiv.org/abs/math-ph/0004021>
<http://mathworld.wolfram.com/BallLinePicking.html>

8.4.2.5 void HyperballDistanceSupport (double * *t*, double * *parameters*)

Calculates the support for the PDF and CDF of the distance between two random in a hyper-ball.

Parameters

<i>\$t</i>	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points in a hyper-ball.
<i>\$parameters</i>	\$parameters[0] is the dimension of the hyper-ball and \$parameters[1] is the radius of the hyper-ball.

Returns

The lower end of the interval is returned in $t[0]$ and the upper end of the interval is returned in $t[1]$.

8.4.2.6 double HyperballDistanceVar (double * parameters)

Calculates the variance of distances between two random points in a hyper-ball.

From Tu, S.-J. and Fischbach, E. "A New Geometric Probability Technique for an $\{N\}$ -Dimensional Sphere and Its Applications"

Parameters

<i>parameters</i>	$parameters[0]$ is the dimension of the hyper-ball and $parameters[1]$ is the radius of the hyper-ball.
-------------------	---

Returns

The variance of distances between two random points in a hyper-ball.

See Also

<http://arxiv.org/abs/math-ph/0004021>
<http://mathworld.wolfram.com/BallLinePicking.html>

8.5 Line.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a line.

Functions

- double [LineDistancePDF](#) (double t, double *parameters)
- double [LineDistanceCDF](#) (double a, double *b)
- double [LineDistanceMean](#) (double *parameters)
- double [LineDistanceVar](#) (double *parameters)
- void [LineDistanceSupport](#) (double *t, double *parameters)
- void [LineDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **LineDistanceName**
- char * **LineDistanceDescription**
- int **LineDistanceNpar**

8.5.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a line.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.5.2 Function Documentation

8.5.2.1 double LineDistanceCDF (double *t*, double * *parameters*)

Implements the CDF of the distance between two random points on a line.

Parameters

<i>\$t</i>	The distance to calculate the density for.
<i>\$parameters</i>	<i>\$parameters</i> [0] is the length of the line.

Returns

The cumulative density at *\$t*.

See Also

<http://mathworld.wolfram.com/LineLinePicking.html>

8.5.2.2 void LineDistanceCheckParameters (double * *parameters*, int * *result*, char * *error_str*)

Intended to determine if the parameters supplied are valid input to the other functions implemented in this file. However as there is only one parameter and the calling function checks that it is positive this is merely a place holder to allow for a complete implementation in geometries that have more complex relationships between parameters.

Parameters

<i>\$parameters</i>	<i>parameters</i> [0] is the length of the line under consideration.
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by *\$result* and a message explaining the error is copied in to the location pointed to *\$error_str*

8.5.2.3 double LineDistanceMean (double * *parameters*)

Calculates the mean of the distance between two random points on a line.

Parameters

<i>\$parameters</i>	<i>\$parameters</i> [0] is the length of the line.
---------------------	--

Returns

The mean distance between two points on a line

See Also

<http://mathworld.wolfram.com/LineLinePicking.html>

8.5.2.4 double LineDistancePDF (double *t*, double * *parameters*)

Implements the PDF of the distance between two random points on a line.

Parameters

t	The distance to calculate the density for.
$parameters$	$parameters[0]$ is the length of the line.

Returns

The density at t .

See Also

<http://mathworld.wolfram.com/LineLinePicking.html>

8.5.2.5 void LineDistanceSupport (double * t , double * $parameters$)

Calculates the support for the PDF and CDF of the distance between two random points on a line

Parameters

t	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points on a line.
$parameters$	$parameters[0]$ is the length of the line.

Returns

The lower end of the interval is returned in $t[0]$ and the upper end of the interval is returned in $t[1]$.

8.5.2.6 double LineDistanceVar (double * $parameters$)

Calculates the variance of the distances between two random points on a line.

Currently calculated numerically.

Parameters

$parameters$	$parameters[0]$ is the length of the line.
--------------	--

Returns

The variance of distances between two points on a line

See Also

<http://mathworld.wolfram.com/BallLinePicking.html>

8.6 LinePicking.h File Reference

Exposes to matlab and R a set of functions that implement PDF, CDF, mean and variance of the distance between two random points in various geometries.

```
#include <math.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <stdio.h>
#include "Square.h"
#include "Disk.h"
#include "Hyperball.h"
#include "Rectangle.h"
#include "Line.h"
#include "Cube.h"
#include "Sphere.h"
#include "SphereGeodesic.h"
#include "PrismGeodesic.h"
#include <R.h>
```

Data Structures

- struct [LinePickingRec](#)

Macros

- #define **_LINEPICKING_H**
- #define **PRINT_STDOUT**(...) Rprintf(__VA_ARGS__)
- #define **ExpandFields**(_x)
- #define **elements**(x) (sizeof(x) / sizeof(x[0]))
- #define **NUMBER_OF_PROBLEMS** elements(LinePickingFields)

Functions

- void [LinePickingNumberOfProblems](#) (int *)
- void **LinePickingPrintAllProblems** ()
- void **LinePickingAllProblems** (char **, char **)
- void **LinePickingProblemLookup** (int *, char **, char **)
- void [LinePickingCheckParameters](#) (int *, double *, int *, int *, char **)
- void [LinePickingSupport](#) (double *, int *, double *, int *, int *, char **)
- void [LinePickingPDF](#) (double *, double *, int *, int *, double *, int *, int *, char **)
- void [LinePickingCDF](#) (double *, double *, int *, int *, double *, int *, int *, char **)
- void **LinePickingMean** (double *, int *, double *, int *, int *, char **)
- void [LinePickingVar](#) (double *, int *, double *, int *, int *, char **)

Variables

- [LinePickingRec](#) **LinePickingFields** []

8.6.1 Detailed Description

Exposes to matlab and R a set of functions that implement PDF, CDF, mean and variance of the distance between two random points in various geometries.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au
 Matthew Roughan matthew.roughan@adelaide.edu.au

Date

22/09/2012

8.6.2 Macro Definition Documentation

8.6.2.1 #define ExpandFields(_x)

Value:

```
&_x##_DistancePDF, &_x##_DistanceCDF, \
&_x##_DistanceMean, &_x##_DistanceVar, \
&_x##_DistanceSupport, &_x##_DistanceCheckParameters, \
&_x##_DistanceNpar, &_x##_DistanceName, \
&_x##_DistanceDescription
```

8.6.3 Variable Documentation

8.6.3.1 LinePickingRec LinePickingFields[]

Initial value:

```
=
{
    {ExpandFields(Square)},
    {ExpandFields(Disk)},
    {ExpandFields(Hyperball)},
    {ExpandFields(Rectangle)},
    {ExpandFields(Line)},
    {ExpandFields(Cube)},
    {ExpandFields(Sphere)},
    {ExpandFields(SphereGeodesic)},
    {ExpandFields(PrismGeodesic)}
}
```

8.7 PrismGeodesic.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface (not including the ends) of an upright prism of any cross section. The distance is measured around the surface i.e., it is a geodesic.

Functions

- double [PrismGeodesicDistancePDF](#) (double t, double *parameters)
- double [PrismGeodesicDistanceCDF](#) (double t, double *parameters)
- double [PrismGeodesicDistanceMean](#) (double *parameters)
- double [PrismGeodesicDistanceVar](#) (double *parameters)
- void [PrismGeodesicDistanceSupport](#) (double *t, double *parameters)
- void [PrismGeodesicDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **PrismGeodesicDistanceName**
- char * **PrismGeodesicDistanceDescription**
- int **PrismGeodesicDistanceNpar**

8.7.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface (not including the ends) of an upright prism of any cross section. The distance is measured around the surface i.e., it is a geodesic.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.7.2 Function Documentation

8.7.2.1 double PrismGeodesicDistanceCDF (double *w*, double * *parameters*)

Implements the CDF of the distance between two random points on the surface of an upright prism (excluding the ends). The distance is measured around the the surface of the prism (i.e., a geodesic). Derived by Eric Parsonage eric.parsonage@adelaide.edu.au

Todo Write up the derivation.

Parameters

<i>\$w</i>	The distance to calculate the cumulative density for.
<i>\$parameters</i>	<i>\$parameters</i> [0] is the length of the prism and <i>\$parameters</i> [1] is the perimeter of the prism.

Returns

The cumulative density at *\$w*.

8.7.2.2 void PrismGeodesicDistanceCheckParameters (double * *parameters*, int * *result*, char * *error_str*)

Determines if the parameters supplied are valid input to the other functions implemented in this file.

Parameters

<i>\$parameters</i>	<i>parameters</i> [0] is the length of the prism under consideration and <i>parameters</i> [1] is the perimeter of the prism under consideration.
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by *\$result* and a message explaining the error is copied in to the location pointed to *\$error_str* The only condition this function needs to check is that the length of the prism is at least as long as half the perimeter

8.7.2.3 double PrismGeodesicDistanceMean (double * *parameters*)

Calculates the mean distance between two random points on the surface of an upright prism (excluding the ends).

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au

Parameters

<i>\$parameters</i>	\$parameters[0] is the length of the prism and \$parameters[1] is the perimeter of the prism.
---------------------	---

Returns

The mean distance between two random points on the surface of an upright prism (excluding the ends).

Todo re-derive the mean without using the three part PDF as a basis I am sure a simpler result can be found using a similar methods to that used on the rectangle

8.7.2.4 double PrismGeodesicDistancePDF (double *w*, double * *parameters*)

Implements the PDF of the distance between two random points on the surface of an upright prism (excluding the ends). The distance is measured around the the surface of the prism (i.e., a geodesic). Derived by Eric Parsonage eric.parsonage@adelaide.edu.au

Todo Write up the derivation.

Parameters

<i>\$w</i>	The distance to calculate the density for.
<i>\$parameters</i>	\$parameters[0] is the length of the prism and \$parameters[1] is the perimeter of the prism.

Returns

The density at \$w.

8.7.2.5 void PrismGeodesicDistanceSupport (double * *t*, double * *parameters*)

Calculates the support for the PDF and CDF of the distance between two random points on the surface of an upright prism (excluding the ends).

Parameters

<i>\$t</i>	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points on the surface of an upright prism (excluding the ends).
<i>\$parameters</i>	\$parameters[0] is the length of the prism and \$parameters[1] is the perimeter of the prism.

Returns

The lower end of the interval is returned in *\$t*[0] and the upper end of the interval is returned in *\$t*[1].

8.7.2.6 double PrismGeodesicDistanceVar (double * *parameters*)

Calculates the variance of distances between two random points on the surface of an upright prism (excluding the ends).

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au

Parameters

<i>\$parameters</i>	\$parameters[0] is the length of the prism and \$parameters[1] is the perimeter of the prism.
---------------------	---

Returns

The variance of distances between two random points on the surface of an upright prism (excluding the ends).

Todo re-derive the variance without using the three part PDF as a basis I am sure a simpler result can be found using a similar methods to that used on the rectangle

8.8 Rectangle.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a rectangle.

Functions

- double [RectangleDistancePDF](#) (double t, double *parameters)
- double [RectangleDistanceCDF](#) (double t, double *parameters)
- double [RectangleDistanceMean](#) (double *parameters)
- double [RectangleDistanceVar](#) (double *parameters)
- void [RectangleDistanceSupport](#) (double *t, double *parameters)
- void [RectangleDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **RectangleDistanceName**
- char * **RectangleDistanceDescription**
- int **RectangleDistanceNpar**

8.8.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a rectangle.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.8.2 Function Documentation

8.8.2.1 double RectangleDistanceCDF (double w, double * parameters)

Implements the CDF of the distance between two random points on a rectangle.

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au

Todo Write up the derivation.

Parameters

<i>\$w</i>	The distance to calculate the cumulative density for.
<i>\$parameters</i>	<i>\$parameters</i> [0] is the length of one side of the rectangle and <i>\$parameters</i> [1] is the length of the other.

Returns

The cumulative density at *\$w*.

8.8.2.2 void RectangleDistanceCheckParameters (double * *parameters*, int * *result*, char * *error_str*)

Intended to determine if the parameters supplied are valid input to the other functions implemented in this file. However as the calling function checks that the supplied parameters are positive this is merely a place holder to allow for a complete implementation in geometries that have more complex relationships between parameters.

Parameters

<i>\$parameters</i>	<i>parameters</i> [0] and <i>parameters</i> [1] are the lengths of the sides of the rectangle under consideration.
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by *\$result* and a message explaining the error is copied in to the location pointed to *\$error_str*

8.8.2.3 double RectangleDistanceMean (double * *parameters*)

Calculates the mean distance between two random points on a rectangle.

From "Random Distances Within a Rectangle and Between Two Rectangles", B. Ghosh, Bulletin of the Calcutta Mathematical Society, Col.43 (1), p.17-24, 1951. "RANDOM POINTS ASSOCIATED WITH RECTANGLES", A.M. MATHAI - R MOSCHOPOULOS - G. PEDERZOLI RENDICONTI DEL CIRCOLO MATEMATICO DI PALERMO, Serie 11, Tomo XLVIII (1999), pp. 163-190

Parameters

<i>\$parameters</i>	<i>\$parameters</i> [0] is the length of one side of the rectangle and <i>\$parameters</i> [1] is the length of the other.
---------------------	--

Returns

The mean distance between two random points on a rectangle.

8.8.2.4 double RectangleDistancePDF (double *t*, double * *parameters*)

Implements the PDF of the distance between two random points on a rectangle.

From "Random Distances Within a Rectangle and Between Two Rectangles", B. Ghosh, Bulletin of the Calcutta Mathematical Society, Col.43 (1), p.17-24, 1951. "RANDOM POINTS ASSOCIATED WITH RECTANGLES", A.M. MATHAI - R MOSCHOPOULOS - G. PEDERZOLI RENDICONTI DEL CIRCOLO MATEMATICO DI PALERMO, Serie 11, Tomo XLVIII (1999), pp. 163-190

Parameters

t	The distance to calculate the density for.
$parameters$	$parameters[0]$ is the length of one side of the rectangle and $parameters[1]$ is the length of the other.

Returns

The density at t .

8.8.2.5 void RectangleDistanceSupport (double * t , double * $parameters$)

Calculates the support for the PDF and CDF of the distance between two random points on a rectangle.

Parameters

t	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points on a rectangle.
$parameters$	$parameters[0]$ is the length of one side of the rectangle and $parameters[1]$ is the length of the other.

Returns

The lower end of the interval is returned in $t[0]$ and the upper end of the interval is returned in $t[1]$.

8.8.2.6 double RectangleDistanceVar (double * $parameters$)

Calculates the variance of distances between two random points on a rectangle.

From "Random Distances Within a Rectangle and Between Two Rectangles", B. Ghosh, Bulletin of the Calcutta Mathematical Society, Col.43 (1), p.17-24, 1951. "RANDOM POINTS ASSOCIATED WITH RECTANGLES", A.M. MATHAI - R MOSCHOPOULOS - G. PEDERZOLI RENDICONTI DEL CIRCOLO MATEMATICO DI PALERMO, Serie 11, Tomo XLVIII (1999), pp. 163-190

Parameters

$parameters$	$parameters[0]$ is the length of one side of the rectangle and $parameters[1]$ is the length of the other.
--------------	--

Returns

The variance of distances between two random points on a rectangle.

8.9 Sphere.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface of a sphere.

Functions

- double [SphereDistancePDF](#) (double t , double * $parameters$)
- double [SphereDistanceCDF](#) (double a , double * b)
- double [SphereDistanceMean](#) (double * $parameters$)
- double [SphereDistanceVar](#) (double * $parameters$)

- void [SphereDistanceSupport](#) (double *t, double *parameters)
- void [SphereDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **SphereDistanceName**
- char * **SphereDistanceDescription**
- int **SphereDistanceNpar**

8.9.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface of a sphere.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.9.2 Function Documentation

8.9.2.1 double SphereDistanceCDF (double s, double * parameters)

Implements the CDF of the distance between two random points on the surface of a sphere.

From Solomon, H. Geometric Probability. Philadelphia, PA: SIAM, 1978.

Parameters

\$s	The distance to calculate the cumulative density for.
\$parameters	\$parameters[0] is the radius of the sphere.

Returns

The cumulative density at \$s.

See Also

<http://mathworld.wolfram.com/SphereLinePicking.html>

8.9.2.2 void SphereDistanceCheckParameters (double * parameters, int * result, char * error_str)

Intended to determine if the parameters supplied are valid input to the other functions implemented in this file. However as there is only one parameter and the calling function checks that it is positive this is merely a place holder to allow for a complete implementation in geometries that have more complex relationships between parameters.

Parameters

\$parameters	parameters[0] is the radius of the sphere under consideration.
\$result	Pointer to storage for an integer indicating any errors in the supplied parameters.
\$error_str	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by \$result and a message explaining the error is copied in to the location pointed to \$error_str

8.9.2.3 double SphereDistanceMean (double * parameters)

Calculates the mean of the distance between two random points on the surface of a sphere.

From Solomon, H. Geometric Probability. Philadelphia, PA: SIAM, 1978.

Parameters

\$parameters	\$parameters[0] is the radius of the sphere.
--------------	--

Returns

The mean of the distance between two random points on the surface of a sphere

See Also

<http://mathworld.wolfram.com/SphereLinePicking.html>

8.9.2.4 double SphereDistancePDF (double s, double * parameters)

Implements the PDF of the distance between two random points on the surface of a sphere.

From Solomon, H. Geometric Probability. Philadelphia, PA: SIAM, 1978.

Parameters

\$s	The distance to calculate the density for.
\$parameters	\$parameters[0] is the radius of the sphere.

Returns

The density at \$s.

See Also

<http://mathworld.wolfram.com/SphereLinePicking.html>

8.9.2.5 void SphereDistanceSupport (double * t, double * parameters)

Calculates the support for the PDF and CDF of the distance between two random points on the surface of a sphere.

Parameters

\$t	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points on the surface of a sphere.
\$parameters	\$parameters[0] is the radius of the sphere.

Returns

The lower end of the interval is returned in `$t[0]` and the upper end of the interval is returned in `$t[1]`.

8.9.2.6 double SphereDistanceVar (double * parameters)

Calculates the variance of the distances between two random points on the surface of a sphere.

From Solomon, H. Geometric Probability. Philadelphia, PA: SIAM, 1978.

Parameters

<code>\$parameters</code>	<code>\$parameters[0]</code> is the radius of the sphere.
---------------------------	---

Returns

The variance of the distances between two random points on the surface of a sphere

See Also

<http://mathworld.wolfram.com/SphereLinePicking.html>

8.10 SphereGeodesic.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface of a sphere. The distance is measured around the surface of the sphere i.e., it is a geodesic.

Functions

- double [SphereGeodesicDistancePDF](#) (double s, double *parameters)
- double [SphereGeodesicDistanceCDF](#) (double s, double *parameters)
- double [SphereGeodesicDistanceMean](#) (double *parameters)
- double [SphereGeodesicDistanceVar](#) (double *parameters)
- void [SphereGeodesicDistanceSupport](#) (double *t, double *parameters)
- void [SphereGeodesicDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **SphereGeodesicDistanceName**
- char * **SphereGeodesicDistanceDescription**
- int **SphereGeodesicDistanceNpar**

8.10.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points on the surface of a sphere. The distance is measured around the surface of the sphere i.e., it is a geodesic.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.10.2 Function Documentation

8.10.2.1 double SphereGeodesicDistanceCDF (double *s*, double * *parameters*)

Implements the CDF of the distance between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au.

Todo Write up the derivation.

Parameters

<i>\$s</i>	The distance to calculate the cumulative density for.
<i>\$parameters</i>	<i>\$parameters</i> [0] is the radius of the sphere.

Returns

The cumulative density at *\$s*.

8.10.2.2 void SphereGeodesicDistanceCheckParameters (double * *parameters*, int * *result*, char * *error_str*)

Intended to determine if the parameters supplied are valid input to the other functions implemented in this file. However as there is only one parameter and the calling function checks that it is positive this is merely a place holder to allow for a complete implementation in geometries that have more complex relationships between parameters.

Parameters

<i>\$parameters</i>	<i>parameters</i> [0] is the radius of the sphere under consideration.
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by *\$result* and a message explaining the error is copied in to the location pointed to *\$error_str*

8.10.2.3 double SphereGeodesicDistanceMean (double * *parameters*)

Calculates the mean of the distance between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au.

Todo Write up the derivation.

Parameters

<i>\$parameters</i>	<i>\$parameters</i> [0] is the radius of the sphere.
---------------------	--

Returns

The mean of the distance between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).

8.10.2.4 `double SphereGeodesicDistancePDF (double s, double * parameters)`

Implements the PDF of the distance between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au.

Todo Write up the derivation.

Parameters

<code>\$s</code>	The distance to calculate the density for.
<code>\$parameters</code>	<code>\$parameters[0]</code> is the radius of the sphere.

Returns

The density at `$s`.

8.10.2.5 `void SphereGeodesicDistanceSupport (double * t, double * parameters)`

Calculates the support for the PDF and CDF of the distance between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).

Parameters

<code>\$t</code>	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).
------------------	---

Returns

The lower end of the interval is returned in `$t[0]` and the upper end of the interval is returned in `$t[1]`.

8.10.2.6 `double SphereGeodesicDistanceVar (double * parameters)`

Calculates the variance of the distances between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).

Derived by Eric Parsonage eric.parsonage@adelaide.edu.au.

Todo Write up the derivation.

Parameters

<code>\$parameters</code>	<code>\$parameters[0]</code> is the radius of the sphere.
---------------------------	---

Returns

The mean of the distance between two random points on the surface of a sphere measured around the surface of the sphere (i.e., a geodesic).

8.11 Square.h File Reference

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a square.

Functions

- double [SquareDistancePDF](#) (double t, double *parameters)
- double [SquareDistanceCDF](#) (double a, double *b)
- double [SquareDistanceMean](#) (double *parameters)
- double [SquareDistanceVar](#) (double *parameters)
- void [SquareDistanceSupport](#) (double *t, double *parameters)
- void [SquareDistanceCheckParameters](#) (double *parameters, int *result, char *error_str)

Variables

- char * **SquareDistanceName**
- char * **SquareDistanceDescription**
- int **SquareDistanceNpar**

8.11.1 Detailed Description

Functions to provide PDF, CDF, mean and variance of the distance between two random points on a square.

Author

Eric Parsonage eric.parsonage@adelaide.edu.au

Date

22/09/2012

8.11.2 Function Documentation

8.11.2.1 double SquareDistanceCDF (double t, double * parameters)

Implements the CDF of the distance between two random points on a square.

Parameters

<i>\$t</i>	The distance to calculate the cumulative density for.
<i>\$parameters</i>	<i>\$parameters</i> [0] is the size of the square.

Returns

The cumulative density at *\$t*.

See Also

<http://mathworld.wolfram.com/SquareLinePicking.html>

8.11.2.2 void SquareDistanceCheckParameters (double * parameters, int * result, char * error_str)

Intended to determine if the parameters supplied are valid input to the other functions implemented in this file. However as there is only one parameter and the calling function checks that it is positive this is merely a place holder to allow for a complete implementation in geometries that have more complex relationships between parameters.

Parameters

<i>\$parameters</i>	parameters[0] is the length of the sides of the square under consideration.
<i>\$result</i>	Pointer to storage for an integer indicating any errors in the supplied parameters.
<i>\$error_str</i>	Pointer to storage for a message explaining any errors in the supplied parameters.

Returns

Any error conditions are indicated by placing a value other than 0 in the location pointed to by *\$result* and a message explaining the error is copied in to the location pointed to *\$error_str*

8.11.2.3 double SquareDistanceMean (double * *parameters*)

Calculates the mean of the distance between two random points on a square.

Parameters

<i>\$parameters</i>	\$parameters[0] is the size of the square.
---------------------	--

Returns

The mean distance between two random points on a square.

See Also

<http://mathworld.wolfram.com/SquareLinePicking.html>

8.11.2.4 double SquareDistancePDF (double *t*, double * *parameters*)

Implements the PDF of the distance between two random points on a square.

Parameters

<i>\$t</i>	The distance to calculate the density for.
<i>\$parameters</i>	\$parameters[0] is the size of the square.

Returns

The density at *\$t*.

See Also

<http://mathworld.wolfram.com/SquareLinePicking.html>

8.11.2.5 void SquareDistanceSupport (double * *t*, double * *parameters*)

Calculates the support for the PDF and CDF of the distance between two random points on a square.

Parameters

<i>\$t</i>	Pointer to storage for lower and upper ends of the support for the PDF and CDF of the distance between two random points on a square.
<i>\$parameters</i>	\$parameters[0] is the size of the square.

Returns

The lower end of the interval is returned in `$t[0]` and the upper end of the interval is returned in `$t[1]`.

8.11.2.6 double SquareDistanceVar (double * *parameters*)

Calculates the variance of the distances between two random points on a square.

Parameters

<i>\$parameters</i>	<code>\$parameters[0]</code> is the size of the square.
---------------------	---

Returns

The variance of the distances between two points random points on a square.

See Also

<http://mathworld.wolfram.com/SquareLinePicking.html>

Index

- beta
 - beta.h, [15](#)
- beta.h, [15](#)
 - beta, [15](#)
 - beta_inc, [16](#)
- beta_inc
 - beta.h, [16](#)
- Cube.h, [16](#)
 - CubeDistanceCDF, [17](#)
 - CubeDistanceCheckParameters, [17](#)
 - CubeDistanceMean, [17](#)
 - CubeDistancePDF, [18](#)
 - CubeDistanceSupport, [18](#)
 - CubeDistanceVar, [18](#)
- CubeDistanceCDF
 - Cube.h, [17](#)
- CubeDistanceCheckParameters
 - Cube.h, [17](#)
- CubeDistanceMean
 - Cube.h, [17](#)
- CubeDistancePDF
 - Cube.h, [18](#)
- CubeDistanceSupport
 - Cube.h, [18](#)
- CubeDistanceVar
 - Cube.h, [18](#)
- Disk.h, [19](#)
 - DiskDistanceCDF, [19](#)
 - DiskDistanceCheckParameters, [20](#)
 - DiskDistanceMean, [20](#)
 - DiskDistancePDF, [20](#)
 - DiskDistanceSupport, [21](#)
 - DiskDistanceVar, [21](#)
- DiskDistanceCDF
 - Disk.h, [19](#)
- DiskDistanceCheckParameters
 - Disk.h, [20](#)
- DiskDistanceMean
 - Disk.h, [20](#)
- DiskDistancePDF
 - Disk.h, [20](#)
- DiskDistanceSupport
 - Disk.h, [21](#)
- DiskDistanceVar
 - Disk.h, [21](#)
- ExpandFields
 - LinePicking.h, [28](#)
- Hyperball.h, [21](#)
 - HyperballDistanceCDF, [22](#)
 - HyperballDistanceCheckParameters, [22](#)
 - HyperballDistanceMean, [22](#)
 - HyperballDistancePDF, [23](#)
 - HyperballDistanceSupport, [23](#)
 - HyperballDistanceVar, [24](#)
- HyperballDistanceCDF
 - Hyperball.h, [22](#)
- HyperballDistanceCheckParameters
 - Hyperball.h, [22](#)
- HyperballDistanceMean
 - Hyperball.h, [22](#)
- HyperballDistancePDF
 - Hyperball.h, [23](#)
- HyperballDistanceSupport
 - Hyperball.h, [23](#)
- HyperballDistanceVar
 - Hyperball.h, [24](#)
- Line.h, [24](#)
 - LineDistanceCDF, [25](#)
 - LineDistanceCheckParameters, [25](#)
 - LineDistanceMean, [25](#)
 - LineDistancePDF, [25](#)
 - LineDistanceSupport, [26](#)
 - LineDistanceVar, [26](#)
- LineDistanceCDF
 - Line.h, [25](#)
- LineDistanceCheckParameters
 - Line.h, [25](#)
- LineDistanceMean
 - Line.h, [25](#)
- LineDistancePDF
 - Line.h, [25](#)
- LineDistanceSupport
 - Line.h, [26](#)
- LineDistanceVar
 - Line.h, [26](#)
- LinePicking API, [11](#)
 - LinePickingCDF, [11](#)
 - LinePickingCheckParameters, [11](#)
 - LinePickingNumberOfProblems, [12](#)
 - LinePickingPDF, [12](#)
 - LinePickingSupport, [12](#)
 - LinePickingVar, [12](#)
- LinePicking.h, [27](#)
 - ExpandFields, [28](#)
 - LinePickingFields, [28](#)
- LinePickingCDF

- LinePicking API, [11](#)
- LinePickingCheckParameters
 - LinePicking API, [11](#)
- LinePickingFields
 - LinePicking.h, [28](#)
- LinePickingNumberOfProblems
 - LinePicking API, [12](#)
- LinePickingPDF
 - LinePicking API, [12](#)
- LinePickingRec, [13](#)
- LinePickingSupport
 - LinePicking API, [12](#)
- LinePickingVar
 - LinePicking API, [12](#)
- PrismGeodesic.h, [28](#)
 - PrismGeodesicDistanceCDF, [29](#)
 - PrismGeodesicDistanceCheckParameters, [29](#)
 - PrismGeodesicDistanceMean, [30](#)
 - PrismGeodesicDistancePDF, [30](#)
 - PrismGeodesicDistanceSupport, [30](#)
 - PrismGeodesicDistanceVar, [30](#)
- PrismGeodesicDistanceCDF
 - PrismGeodesic.h, [29](#)
- PrismGeodesicDistanceCheckParameters
 - PrismGeodesic.h, [29](#)
- PrismGeodesicDistanceMean
 - PrismGeodesic.h, [30](#)
- PrismGeodesicDistancePDF
 - PrismGeodesic.h, [30](#)
- PrismGeodesicDistanceSupport
 - PrismGeodesic.h, [30](#)
- PrismGeodesicDistanceVar
 - PrismGeodesic.h, [30](#)
- Rectangle.h, [31](#)
 - RectangleDistanceCDF, [31](#)
 - RectangleDistanceCheckParameters, [32](#)
 - RectangleDistanceMean, [32](#)
 - RectangleDistancePDF, [32](#)
 - RectangleDistanceSupport, [33](#)
 - RectangleDistanceVar, [33](#)
- RectangleDistanceCDF
 - Rectangle.h, [31](#)
- RectangleDistanceCheckParameters
 - Rectangle.h, [32](#)
- RectangleDistanceMean
 - Rectangle.h, [32](#)
- RectangleDistancePDF
 - Rectangle.h, [32](#)
- RectangleDistanceSupport
 - Rectangle.h, [33](#)
- RectangleDistanceVar
 - Rectangle.h, [33](#)
- Sphere.h, [33](#)
 - SphereDistanceCDF, [34](#)
 - SphereDistanceCheckParameters, [34](#)
 - SphereDistanceMean, [35](#)
 - SphereDistancePDF, [35](#)
 - SphereDistanceSupport, [35](#)
 - SphereDistanceVar, [36](#)
- SphereDistanceCDF
 - Sphere.h, [34](#)
- SphereDistanceCheckParameters
 - Sphere.h, [34](#)
- SphereDistanceMean
 - Sphere.h, [35](#)
- SphereDistancePDF
 - Sphere.h, [35](#)
- SphereDistanceSupport
 - Sphere.h, [35](#)
- SphereDistanceVar
 - Sphere.h, [36](#)
- SphereGeodesic.h, [36](#)
 - SphereGeodesicDistanceCDF, [37](#)
 - SphereGeodesicDistanceCheckParameters, [37](#)
 - SphereGeodesicDistanceMean, [37](#)
 - SphereGeodesicDistancePDF, [37](#)
 - SphereGeodesicDistanceSupport, [38](#)
 - SphereGeodesicDistanceVar, [38](#)
- SphereGeodesicDistanceCDF
 - SphereGeodesic.h, [37](#)
- SphereGeodesicDistanceCheckParameters
 - SphereGeodesic.h, [37](#)
- SphereGeodesicDistanceMean
 - SphereGeodesic.h, [37](#)
- SphereGeodesicDistancePDF
 - SphereGeodesic.h, [37](#)
- SphereGeodesicDistanceSupport
 - SphereGeodesic.h, [38](#)
- SphereGeodesicDistanceVar
 - SphereGeodesic.h, [38](#)
- Square.h, [38](#)
 - SquareDistanceCDF, [39](#)
 - SquareDistanceCheckParameters, [39](#)
 - SquareDistanceMean, [40](#)
 - SquareDistancePDF, [40](#)
 - SquareDistanceSupport, [40](#)
 - SquareDistanceVar, [41](#)
- SquareDistanceCDF
 - Square.h, [39](#)
- SquareDistanceCheckParameters
 - Square.h, [39](#)
- SquareDistanceMean
 - Square.h, [40](#)
- SquareDistancePDF
 - Square.h, [40](#)
- SquareDistanceSupport
 - Square.h, [40](#)
- SquareDistanceVar
 - Square.h, [41](#)