Package ‘gensphere’

September 16, 2017

Type Package

Title Generalized Spherical Distributions

Version 1.1

Date 2017-09-16

Author John P Nolan

Maintainer John P Nolan <jpnolan@american.edu>

Depends R (>= 3.0), mvmesh, geometry, SphericalCubature, rgl, utils, grDevices, graphics

Description Define and compute with generalized spherical distributions - multivariate probability laws that are specified by a star shaped contour (directional behavior) and a radial component.

License GPL (>= 3)

Imports SimplicialCubature

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-16 21:19:43 UTC

R topics documented:

gensphere-package *gensphere*

Description

Define and compute with generalized spherical distributions - multivariate probability laws that are specified by a star shaped contour (directional behavior) and a radial component.

Details

This package implements some classes of generalized spherical distributions in dimensions 2, 3, and above. Functions `cfunc.new`, `cfunc.add.term`, `cfunc.finish` give a flexible way to define a range of shapes for the star-shaped contours. Then function `gensphere` defines a generalized spherical distribution using a contour function and a specification of the radial term. Function `dgensphere` is used to compute the multivariate density g(x) for X and function `rgensphere` is used to simulate a sample random vectors with the (approximate) distribution X.

A large class of distribution can be described as generalized spherical laws. In particular, all isotropic/radially symmetric distributions and all elliptically contoured distributions are generalized spherical laws. Such distributions can be represented as: $\mathbf{X} = R\mathbf{S}$ where R is a positive random variable and **S** is a random vector distributed uniformly (with respect to surface area) on the contour, see Nolan (2015).

Throughout this package, points in d-dimensional space are represented as column vectors; this is different than what base R and packages **mvmesh**, **geometry**, etc. use; but it is the same as package **SphericalCubature**, **SimplicialCubature**, and other packages.

This research was supported by an agreement with Cornell University, Operations Research & Information Engineering, under contract W911NF-12-1-0385 from the Army Research Development and Engineering Command.

Please let me know if you find any mistakes. I will try to fix bugs promptly. Constructive comments for improvements are welcome; actually implementing any suggestions will be dependent on time constraints.

Version 1.0 was released on 18 May 2016. Version 1.1 was released on 13 September 2017 and includes a new optional argument norm.const.method in the function cfunc.finish. Also changes were made to accomodate changes in package SphericalCubature.

Author(s)

John P Nolan

Maintainer: John P Nolan

References

B. C. Arnold, E. Castillo and J. M. Sarabia, Multivariate distributions defined in terms of contours, J. Stat. Planning and Inference, 138, 4158 - 4171, 2008

C. Fernandez, J. Osiewalksi and M. F. J. Steel, Modeling and Inference with v-Spherical Distributions, J. Amer. Stat. Assoc., 90, 1331-1340, 1995

J. P. Nolan, Models for generalized spherical and related distributions. arXiv preprint, Sept. 2015

See Also

cfunc.new, gensphere

---

cfunc.new *Define and evaluate a contour function*

Description

The directional part of a generalized spherical distribution is defined by a contour function, cfunc for short. These functions are used to define a contour function and then evaluate it.

Usage

```
cfunc.new(d)
cfunc.add.term(cfunc, type, k)
cfunc.finish(cfunc, nsubdiv = 2, maxEvals=100000, norm.const.method="integrate",...)
cfunc.eval(cfunc, x)
```

Arguments

d	dimension of the space
cfunc	an object of class "gensphere.contour"
type	string describing what type of term to add to the contour
k	a vector of constants used to specify a term
x	a (d x n) matrix, with columns x[,i] being points in R^d
nsubdiv	number of dyadic subdivisions of the sphere, controls the refinement of the tessellation. Typical values are 2 or 3.
maxEvals	maximum number of evaluations of the integrand function, see details section below
norm.const.method	method used to compute the norming constant. Can be "integrate" (the default, in which case the contour function is numerically integrated to get the norming constant), or "simplex.area" (in which case no integration is done; the surface area of the contour is approximated by the surface area of the tesselation). This later choice is for complex surface or higher dimensional surfaces where the numerical integration may fail or take a very long time. This allows simulation in cases where the numerical integration is not feasible.
...	optional arguments to pass to integration routine, see details section below

Details

A contour function describes the directional behavior of a generalized spherical distribution. In this package, a contour function is built by calling three functions: `cfunc.new` to start the definition of a d-dimensional contour, `cfunc.add.term` to add a new term to a contour (may be called more than once), and `cfunc.finish` to complete the defintion of a contour.

When adding a term, type is one of the literal strings "constant", "elliptical", "proj.normal", "lp.norm", "gen.lp.norm", or "cone". The vector k contains the constants necessary to specify the desired shape. k[1]=the first element of k is always a scale, it allows one to expand or contract a shape. The remaining elements of k depend on the value of 'type'.:

- "constant": k is a single number; k[1]=radius of the sphere
- "elliptical": k is a vector of length d^2+1; k[1]=scale, k[2:(d^2+1)] specify the symmetric positive definite matrix B with is used to compute the elliptical contour
- "proj.normal": k is a vector of length d+2; k[1]=scale, mu=k[2:(d+1)] is the vector pointing in the direction where the normal bump is centered, k[d+2]=scale of the isotropic normal bump
- "lp.norm": k is a vector of length 2; k[1]=scale and k[2]=p, the power used in the l_p norm
- "gen.lp.norm": k is vector of length 2+m*d for some positive integer m; k[1]=scale, k[2]=p, the power used in the l-p norm, k[3:(2+m*d)] spcifies a matrix A that is used to compute || A x ||_p
- "cone": k is a vector of length d+2, k[1]=scale, mu=k[2:(d+1)]= the center of the cone, k[d+2]=base of the cone

Note that `cfunc.finish` does a lot of calculation, and may take a while, especially in dimension d > 2. The most time consuming part is numerically integrating over the contour, a (d-1) dimensional surface in d-dimensional space and in tesselating the contour in a way that focuses on the bulges in the contour from cones and normal bumps. The integration is required to calculate the norming constant needed to compute the density. This integration is performed by using function `adaptIntegrateSphereTri` in **SphericalCubature** and is numerically challenging. In dimension d > 3 or if nsubdiv > 4, users may need to adjust the arguments maxEvals and ... The default value maxEvals=100000 workw in most 3 dim. problems, and it takes a few seconds to calculate. (For an idea of the size and time required, a d=4 dim. case used maxEvals=1e+7 and took around 5 minutes. A d=5 dim. case used maxEvals=1e+8, used 160167 simplices and took over 2 days.) Note that this calculation is only done once; calculating densities and simulating is still fast in higher dimensions. It may be useful to save a complicated/large contour object so that it can be reused across R sessions via `save(cfunc)` and `load(cfunc)`.

Note: the first time `cfunc.finish` is called, a warning message about "no degenerate regions are returned" is printed by the package **geometry**. I do not know how to turn that off; so just ignore it.

`cfunc.eval` is used to evaluate a completed contour function.

Value

`cfunc.new` and `cfunc.add.term` return a list that is an incomplete definition of a contour function. `cfunc.finish` completes the definition and returns an S3 object of class "gensphere.contour" with fields:

`d`	dimension

`m` number of terms in the contour function, i.e. the number of times `cfunc.add.term` was called

`term` a vector length m of type list, with each list describing a term in the contour function

`norm.const` norming constant

`functionEvaluations`
number of times the integrand (contour) function is evaluated by `adaptIntegrateSphereTri` when computing norm.const

`tessellation` an object of type "mvmesh" that gives a geometrical description of the contour. It is used to plot the contour and to simulate from the contour

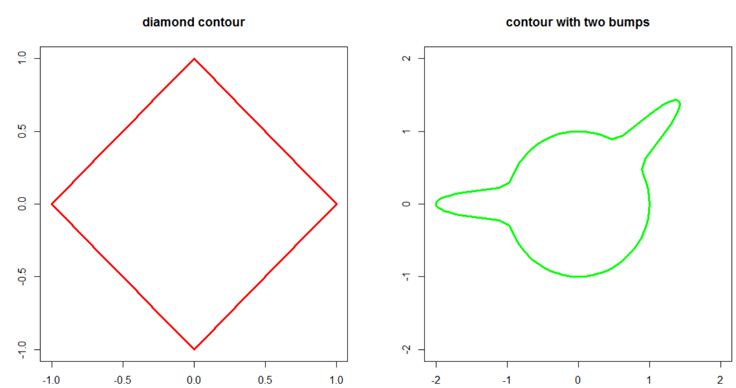
`tessellation.weights`
weights used in simulation; surface area of the simplices in the tessellation

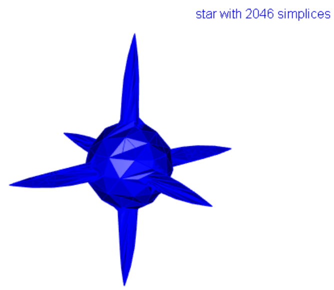
`simplex.count` vector of length 3 giving the number of simplices used at the end of three internal stages of construction: after initial subdivision, after refining the sphere based on cones and bumps, and final count have adaptive integration routine partitions the sphere

`norm.const.method`
value of input argument norm.const.method

`cfunc.eval` returns a vector of length n=nrow(x); y[i] = cfunc(x[,i]) = value of the contour function at point x[,i].

The plots below show the three contours defined in the examples below.





Examples

```
# 2-dim diamond
cfunc1 <- cfunc.new(d=2)
cfunc1 <- cfunc.add.term( cfunc1,"lp.norm",k=c(1,1))
cfunc1 <- cfunc.finish( cfunc1 )
cfunc1
cfunc.eval( cfunc1, c(sqrt(2)/2, sqrt(2)/2) )
plot( cfunc1, col='red', lwd=3, main="diamond contour")

# 2-dim blob
cfunc2 <- cfunc.new(d=2)
cfunc2 <- cfunc.add.term( cfunc2,"constant",k=1)
cfunc2 <- cfunc.add.term( cfunc2,"proj.normal",k=c( 1, sqrt(2)/2, sqrt(2)/2, 0.1) )
cfunc2 <- cfunc.add.term( cfunc2,"proj.normal",k=c( 1, -1,0, 0.1) )
cfunc2 <- cfunc.finish( cfunc2, nsubdiv=4 )

plot( cfunc2, col='green', lwd=3, main="contour with two bumps")

# 3-dim star-like contour
cfunc3 <- cfunc.new(d=3)
cfunc3 <- cfunc.add.term( cfunc3, "constant", k=0.5 )
for (i in 1:3) {
  u <- c(0,0,0);  u[i] <- 1  # i-th unit vector
  cfunc3 <- cfunc.add.term( cfunc3, "cone", k=c(1,u,.2) )
  cfunc3 <- cfunc.add.term( cfunc3, "cone", k=c(1,-u,.2) )
}
cfunc3 <- cfunc.finish( cfunc3, maxEvals=400000 ) # this is slow, ~ 15 min

plot( cfunc3, show.faces=TRUE, col='blue')
nS <- dim(cfunc3$tessellation$S)[3]
title3d( paste("star with",nS,"simplices"))
```

gensphere *Generalized spherical distribution definition, density, simulation*

Description

Define a generalized spherical distribution by specifying a contour function, a radial density function, a radial simulation function, and a value of the density at the origin. Once it is defined, compute density and simulate that distribution.

Usage

```
gensphere(cfunc, dradial, rradial, g0)
dgensphere(x, gs.dist)
rgensphere(n, gs.dist)
```

Arguments

cfunc	contour function object defined by `cfunc.new`, `cfunc.add.term` and `cfunc.finish`
dradial	a function to evaluate the density for the radial component of distribution
rradial	a function to simulate values of the radial distribution
g0	g(0) = value of the multivariate density at the origin
x	(d x n) matrix of point where the density is to be evaluated. Columns x[,i] are vectors in d-space
gs.dist	a generalized spherical distribution, an object returned by function `gensphere`
n	number of values to generate

Details

A generalized spherical distribution is specified by calling function `gensphere` with the contour function (defined via function `cfunc.new`, `cfunc.add.term` and `cfunc.finish`), a function to compute the density of the radial term R, a runction to simulate from the radial term R, and g(0)=the value of the density at the origin. See the general representation of generalized spherical laws in gensphere-package.

If the distribution is d dimensional and the radial term is a gamma distribution with shape=shape and scale=1,g(0)=0 if d < shape, g(0)=cfunc$norm.const if d=shape, $g(0) = \infty$ if d > shape. In general, $g(0) = \lim_{r \to 0^+} r^{1-d} dradial(r)$.

Value

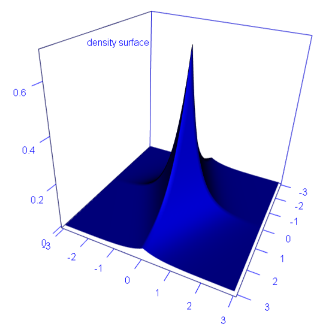
`gensphere` returns an S3 object of class "gensphere.distribution" with components:

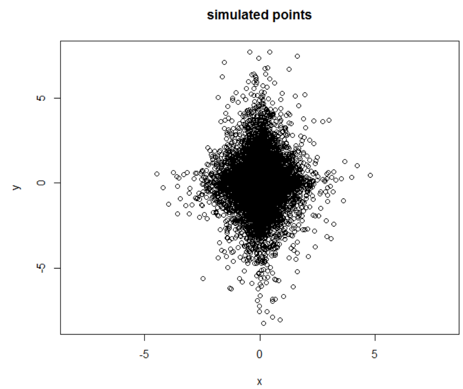
cfunc	a contour function defined with `cfunc.new`, etc.
dradial	a function that evaluates the desnity of the radial component
rradial	a function that simulates values of the radial component
g0	g(0), the value of the multivariate density g(x) at the origin

dgensphere returns a numeric vector y that contains the value of the density of **X**: y[i]=g(x[,i]), i=1,...,n. Note that g(x) is the density of the vector **X**, whereas `dradial` is the denis of the univariate radial term R.

`rgensphere` returns a (d x n) matrix of simulated values of **X**. Note that these values are an approximation to the distribution of X because the contour is approximated to a limited accuracy in `cfund.finish`.

Here are plots of the density surface and simulated points generated by the examples below.





See Also

gensphere-package, cfunc.new

Examples

```
# define a diamond shaped contour
cfunc1 <- cfunc.new(d=2)
#cfunc1 <- cfunc.add.term( cfunc1,"lp.norm",k=c(1,1))
cfunc1 <- cfunc.add.term( cfunc1,"gen.lp.norm",k=c(1,1,2,0,0,1))
cfunc1 <- cfunc.finish( cfunc1 )
cfunc1


# define a generalized spherical distribution
rradial <- function( n ) { rgamma( n, shape=2 ) }
dradial <- function( x ) { dgamma( x, shape=2 ) }
dist1 <- gensphere( cfunc1, dradial, rradial, g0=cfunc1$norm.const )
dist1

# calculate density at a few points
dgensphere( x=matrix( c(0,0, 0,1, 0,2), nrow=2, ncol=3), dist1 )

# calculate and plot density surface on a grid
xy.grid <- seq(-3,3,.1)
z <- gs.pdf2d.plot( dist1, xy.grid )
title3d("density surface")
```

```
# simulate values from the distribution
x <- rgensphere( 10000, dist1 )
plot(t(x),xlab="x",ylab="y",main="simulated points")
```

---

genspheremisc — *Miscellaneous functions used in working with generalized spherical laws*

Description

Miscellaneous internal functions for the gensphere package.

Usage

```
gs.cone(x, mu, theta0)
gs.elliptical(x, B)
gs.gen.lp.norm(x, p, A)
gs.lp.norm(x,p)
gs.proj.normal(x, mu, sigma)
gs.vfunc.eval(cfunc, x)
gs.pdf2d.plot(gs.dist, xy.grid = seq(-10, 10, 0.1) )
RefineSphericalTessellation(V1, V2)
NearbyPointsOnSphere(x, epsilon)
RotateInNDimensions(x, y)
## S3 method for class 'gensphere.contour'
print(x,...)
## S3 method for class 'gensphere.distribution'
print(x,...)
## S3 method for class 'gensphere.contour'
plot(x,multiplier=1,...)
```

Arguments

`x,y`	vectors representing points in d-dimensional space
`mu`	direction of the mode for a cone/normal bump
`theta0`	angle between peak of the cone and the base of the cone
`B`	(d x d) positive definite shape matrix
`A`	matrix used to compute || A x ||_p
`p`	power of the l^p norm; p=2 is Euclidean distance
`gs.dist`	object of class "gensphere.distribution" defined by `gensphere`
`xy.grid`	a matrix of (x,y) values in 2-dimensions
`cfunc`	an object of class "gensphere.contour" defined by `cfunc.new`, etc.
`...`	optional arguments to the 2-dimensional plot, e.g. col='red', etc.

`sigma`	scale parameter for a normal bump
`epsilon`	vector of positive numbers where there are points added around a particular direction
`V1,V2`	matrices of vertices which are joined together to get a refinement of the grid
`multiplier`	a positive number used to scale the contour

Details

These are undocumented functions that are used internally. The functions `gs.cone`, `gs.elliptical`, `gs.gen.lp.norm`, `gs.lp.norm`, `gs.proj.normal`, `gs.vfunc.eval` are used in evalutating a contour function. `RefineSphericalTessellation`, `NearbyPointsOnSphere` are used in defining the tessellation of the contour that identifies bumps and cones. `gs.pdf2d.plot` and the plot/print methods are initial attempts at plotting and printing a summary of objects.

These functions may change or disappear in the future.

Index