Package 'unfoldr'

October 6, 2016

Title Stereological Unfolding for Spheroidal Particles

Version 0.6

Date 2016-10-04

Author Markus Baaske [aut, cre], Felix Ballani [ctb]

Maintainer Markus Baaske <markus.baaske@math.tu-freiberg.de>

Description Stereological unfolding as implemented in this package consists in the estimation of the joint size-shape-orientation distribution of spheroidal shaped particles based on the same measured quantities of corresponding planar vertical section profiles. A single trivariate discretized version of the (stereological) integral equation in the case of prolate and oblate spheroids is solved numerically by the EM algorithm. The estimation of diameter distribution of spheres from planar sections (Wicksell's corpuscle problem) is also implemented. Further, the package provides routines for the simulation of a Poisson germ-grain process with either spheroids, spherocylinders or spheres as grains together with functions for planar sections.
For the purpose of exact simulation a bivariate size-shape distribution is implemented.

Depends R (>= 3.1.0)

Suggests rgl

License GPL-2

Repository CRAN

LazyData no

RoxygenNote 5.0.1

NeedsCompilation yes

Date/Publication 2016-10-06 19:33:23

R topics documented:

`binning1d` *Binning numeric values*

Description

Vector of count data

Usage

```
binning1d(x, bin, na.rm = FALSE)
```

Arguments

`x`	numeric values to be binned
`bin`	non-decreasingly sorted breaks vector
`na.rm`	logical, removing missing values (including NaN) in the argument `x`?

Details

This function provides basic binning (grouping) of numeric values into classes defined by the breaks vector `bin`. The values are binned according to bin[i[j]]<x[j]≤ bin[i[j]+1] for interval i=1,...,N-1 for `length(bin)=N` and value x[j]. If x[j] > bin[N] or x[j] < bin[1] then x[j] is not counted at all.

Value

Vector of count data

Examples

```
x <- runif(100,0,1)
bin <- seq(0,1,by=0.1)
binning1d(x,bin)
```

`binning3d` *Histogram data*

Description

Count data of size, shape and orientation values.

Usage

```
binning3d(size, angle, shape, breaks, check = TRUE, na.rm = TRUE)
```

Arguments

`size`	vector of sizes
`angle`	vector of angles
`shape`	vector of aspect ratios
`breaks`	list of bin vectors
`check`	logical, default is `TRUE`
`na.rm`	logical, if NAs are to be removed in the data vectors, default is `TRUE`

Details

For each value of planar or spatial measured quantities `size,shape,orientation` the function counts the number of observations falling into each class. The `breaks` list is either obtained from function `setbreaks` or the user supplies his own variant, see `setbreaks` for details. If `check` is `TRUE` some checks on the `breaks` list are done.

Value

3d array of count data

coefficientMatrixSpheres

Calculate coefficients (spheres)

Description

Matrix of coefficients for Wicksell's corpuscle problem

Usage

```
coefficientMatrixSpheres(bin)
```

Arguments

bin	non-decreasing vector of class limits

Details

The function calculates the matrix of coefficients of the discretized integral equation for Wicksell's corpuscle problem.

Value

array of coefficients

References

Ohser, J. and Muecklich, F. Statistical analysis of microstructures in materials science J. Wiley & Sons, 2000

coefficientMatrixSpheroids

Calculate coefficients (spheroids)

Description

Calculate coefficients of discretized integral equation

Usage

```
coefficientMatrixSpheroids(breaks, stype = c("prolate", "oblate"),
  check = TRUE, nCores = getOption("par.unfoldr", 1))
```

Arguments

breaks	list of bin vectors
stype	either prolate or oblate
check	logical, whether to run some input checks
nCores	number of cores used to calculate the coefficients

Details

In order to apply the EM algorithm to the stereological unfolding procedure for the joint size-shape-orientation distribution one first has to calculate the coefficients of the discretized integral equation. This step is the most time consuming part of unfolding the parameters and therefore has been separated in its own function. Further, the number of classes for size, shape and orientation do not need to be equal, whereas the same class limits are used for binning spatial and planar values. This might be changed in future releases. Using multiple cpu cores is controlled by either setting the option 'par.unfoldr' in the global R environment or passing the number of cores nCores directly.

Value

coefficient array

Examples

```
## Not run: inst/examples/coeffarray.R
```

---

cylinders3d *Plot fibre system*

Description

Draw 3d spherocylinders

Usage

```
cylinders3d(S, box, draw.axes = FALSE, draw.box = TRUE, clipping = FALSE,
  ...)
```

Arguments

S	a list of cylinders
box	simulation box
draw.axes	logical: if true, draw the axes
draw.box	logical: if true, draw the bounding box
clipping	logical: if true clip to the bounding box
...	further material properties passed to 3d plotting functions

Details

The function requires the package `rgl` to be installed.

---

drawSpheroidIntersection *Plot Spheroid intersection*

Description

Drawing section profiles in 3D plane

Usage

```
drawSpheroidIntersection(E, n = c(0, 1, 0), np = 25)
```

Arguments

`E`	a list of spheroid intersections
`n`	the normal vector of the intersecting plane
`np`	number of points for polygon approximation of ellipses

Details

The function requires the package `rgl` to be installed.

---

em.saltykov *Expectation Maximization algorithm*

Description

Estimation of empirical sphere diameter distribution

Usage

```
em.saltykov(y, bin, maxIt = 32)
```

Arguments

`y`	vector of observed absolute frequencies of circle diameters
`bin`	non-decreasing vector of class limits
`maxIt`	maximum number of iterations used

Details

The function performs the EM algorithm.

Value

vector of count data of absolute frequenties of sphere diameters

References

Ohser, J. and Muecklich, F. Statistical analysis of microstructures in materials science J. Wiley & Sons, 2000

Examples

```
## beta distributed radii
lam <-3000
theta <- list("shape1"=2,"shape2"=4)
S <- simSphereSystem(theta,lam, rdist="rbeta", box=list(c(0,5)),pl=101)

sp <- planarSection(S,d=2.5)
ret <- unfold(sp,nclass=20)

## Point process intensity
cat("Intensities: ", sum(ret$N_V)/25, "vs.",lam,"\n")

## original diameters
r3d <- unlist(lapply(S,function(x) 2.0*x$r))
rest3d <- unlist(lapply(2:(length(ret$breaks)),
            function(i) rep(ret$breaks[i],sum(ret$N_V[i-1]))))

op <- par(mfrow = c(1, 2))
hist(r3d[r3d<=max(ret$breaks)], breaks=ret$breaks, main="Radius 3d",
     freq=FALSE, col="gray",xlab="r")
hist(rest3d, breaks=ret$breaks,main="Radius estimated",
     freq=FALSE, col="gray", xlab="r")
par(op)
```

---

em.spheroids *Trivariate stereological unfolding*

Description

Estimate the joint size-shape-orientation distribution of spheroids

Usage

```
em.spheroids(P, F, maxIt, nCores = getOption("par.unfoldr", 1))
```

Arguments

P	coefficient array
F	input histogram
maxIt	maximum number of EM iterations
nCores	number of cpu cores used

Details

Given an array of coefficients P and the input histogram F of measured planar section profiles, see binning3d, the function estimates the spatial joint size-shape-orientation distribution of spheroids as a trivariate histogram, triHist, by the EM algorithm. If the option 'par.unfoldr' is set to a user chosen amount of cores then parts of the EM iterations are done in parallel.

Value

trivariate histogram

References

Beneš, V. and Rataj, J. Stochastic Geometry: Selected Topics Kluwer Academic Publishers, Boston, 2004

Examples

```
## Not run:
## Spheroids with lognormal distributed length of axes
## set number of cpu cores (optional)
# options(par.unfoldr=8)

## Intensity: mean number of spheroids per unit volume
lam <- 2500

## simulation parameters
theta <- list("size"=list("meanlog"=-2.5,"sdlog"=0.5),
      "shape"=list("alpha"=0.5),"orientation"=list("kappa"=1.5))
## simualtion
set.seed(1234)
S <- simSpheroidSystem(theta,lam,size="rlnorm",
orientation="rbetaiso",box=list(c(0,5)),stype="prolate",pl=101)
## unfolding
sp <- verticalSection(S,2.5)
ret <- unfold(sp,c(15,12,10),kap=1.25)
cat("Intensities: ", sum(ret$N_V)/25, "vs.",lam,"\n")

## plot 3d trivariate histogram of joint distribution
# cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
# trivarHist(ret$N_V,col=cols,scale=0.9)

param3d <- parameters3d(S)
paramEst <- parameterEstimates(ret$N_V,ret$breaks)

## Marginal histograms
op <- par(mfrow = c(3, 2))
hist(param3d$a[param3d$a<max(ret$breaks$size)],
 main=expression(paste("3D Histogram ", a)),
 breaks=ret$breaks$size,col="gray",right=FALSE,freq=FALSE,xlab="a",ylim=c(0,25))
hist(paramEst$a,
 main=expression(paste("Estimated histogram ",hat(a))),
```

```
 breaks=ret$breaks$size,
 right=FALSE,freq=FALSE,col="gray",
 xlab=expression(hat(a)),ylim=c(0,25))
hist(param3d$Theta[param3d$Theta<max(ret$breaks$angle)],
 main=expression(paste("3D Histogram ", theta)),
 breaks=ret$breaks$angle,col="gray",right=FALSE,freq=FALSE,
 xlab=expression(theta),ylim=c(0,2))
hist(paramEst$Theta,
 main=expression(paste("Estimated Histogram ", hat(theta))),
 breaks=ret$breaks$angle,
 right=FALSE,freq=FALSE,col="gray",
 xlab=expression(hat(theta)),ylim=c(0,2))
hist(param3d$s,main=expression(paste("3D Histogram ", s)),
 col="gray",breaks=ret$breaks$shape,
 right=FALSE,freq=FALSE,xlab="s")
hist(paramEst$s,main=expression(paste("Estimated Histogram ", hat(s))),
 breaks=ret$breaks$shape,
right=FALSE,freq=FALSE,col="gray",xlab=expression(hat(s)))
par(op)

## End(Not run)
```

`getMaxRadius` *Maximum radius of exact simulated spheroids*

Description

Get maximum (random) radius

Usage

```
getMaxRadius(S)
```

Arguments

`S`	spheroid system, simulated by perfect simulation option

Details

In case of exact simulation the maximum radius of all randomly generated radii is returned.

Value

maximum radius

getSphereSystem *Get sphere system*

Description

Get the stored sphere system

Usage

```
getSphereSystem(S)
```

Arguments

S	result of simSphereSystem

Details

The sphere system is internally stored as a C data structure. This function copies and converts the sphere objects to the R level.

Value

list of spheres of class spheres

getSpheroidSystem *Get spheroid system*

Description

Get the internally stored spheroid system

Usage

```
getSpheroidSystem(S)
```

Arguments

S	result of simSpheroidSystem

Details

The spheroid system is stored as a C structure. This function copies and converts the spheroids to the R level.

Value

list of spheroids, either of class prolate or oblate

parameterEstimates *Estimated spatial histogram data*

Description

Get histogram data from estimated joint distribution

Usage

```
parameterEstimates(H, breaks)
```

Arguments

H	trivariate histogram
breaks	breaks as obtained from setbreaks

Details

Given the estimated joint distribution in histogram form the function returns a list of breaks replicates which equals the number of estimated count data for each class.

Value

list of size, angle and shape parameters, see parameters3d

parameters3d *Original parameters*

Description

Extract the original 3d size-shape-orientation parameters

Usage

```
parameters3d(S)
```

Arguments

S	list of spheroids

Details

The function simply extracts the parameters for the original 3d spheroid system either for oblate or prolate spheroids and returns a list which consists of sizes a, the orientation angles Theta and the shape parameters s.

Value

list

`planarSection` *Sphere planar section*

Description

Intersect sphere system

Usage

```
planarSection(S, d, intern = FALSE)
```

Arguments

`S`	list of spheres, see `simSphereSystem`
`d`	distance of the intersecting xy-plane to the origin
`intern`	`intern=FALSE` (default) return all sections otherwise only those which have their centers inside the intersection window

Details

Given a sphere system obtained from `simSphereSystem` the function returns the section radii from intersecting all spheres stored in `S`.

Value

vector of circle radii

`sectionProfiles` *Constuctor section profiles*

Description

Storing structure for section profiles

Usage

```
sectionProfiles(size, angle, type = c("prolate", "oblate"))
```

Arguments

`size`	matrix of major and minor axes sizes
`angle`	orientation angle
`type`	`prolate` or `oblate`, default class is `prolate`

Details

The function aggregates the necessary data for the trivariate unfolding procedure either for `type` `prolate` or `oblate` spheroids. Argument `size` is a numeric matrix containing the axes lengths as columns. The `angle` is the orientation angle between the major axis and the vertical axis direction in the section plane. If the angles range within $[0, 2\pi]$ these are transformed to $[0, \pi/2]$. The function returns a list which consists of either longer or shorter axis `A` of section profiles corresponding to the type of spheroids which are intended to be reconstructed, the aspect ratio as the shape parameter `S` with values in $(0, 1]$, and the orientation angle `alpha`.

Value

section profiles object, either of class `prolate` or `oblate`

---

setbreaks *Break vectors*

Description

Construct class limits vectors

Usage

```
setbreaks(nclass, maxSize, base = NULL, kap = 1, sizeType = c("linear",
  "exp"))
```

Arguments

`nclass`	number of classes
`maxSize`	maximum of `size` values
`base`	constant for size class construction
`kap`	constant for shape class construction
`sizeType`	either `linear` or `exp`, default is `linear`

Details

The function constructs the class limits for the size, shape and orientation parameters. One can either define `linear` class limits of 'size' as $a_i = i\delta, \delta = maxSize/M$ or using exponentially increasing limits: $base^i, i = 1, \ldots, M$. The orientation classes are defined as $\theta_j = j\omega, \omega = \pi/(2N), j = 1, \ldots, N$ in the range $[0, \pi/2]$, where M, N are the number of size classes and the number of orientation classes, respectively. Argument `base` must not be `NULL` if `sizeType` equals "exp".

Value

list of class limits vectors

setupSphereSystem *Setup sphere system*

Description

Reinitialize sphere system after R workspace reloading

Usage

```
setupSphereSystem(S, pl = 0)
```

Arguments

S	sphere system
pl	print level

Details

The internally stored sphere system has to be reinitialized after R workspace reloading. Calling this function is needed only in case one desires to get profile sections of the sphere system again which has been previously stored as an R (list) object and afterwards reloaded.

Value

NULL

setupSpheroidSystem *Setup spheroid system*

Description

Reinitialize spheroid system after R workspace reloading

Usage

```
setupSpheroidSystem(S, mu = c(0, 1, 0), pl = 0)
```

Arguments

S	sphere system
mu	main orientation vector
pl	print level

Details

The internally stored sphere system has to be reinitialized after R workspace reloading. Calling this function is needed only in case one desires to get profile sections of the spheroid system again which has been previously stored as an R (list) object and afterwards reloaded.

Value

NULL

---

simCylinderSystem *Simulation of cylinder system*

Description

Simulation of Poisson cylinder system

Usage

```
simCylinderSystem(theta, lam, size = "const", shape = "const",
  orientation = "rbetaiso", rjoint = NULL, box = list(c(0, 1)),
  mu = c(0, 1, 0), perfect = TRUE, pl = 0, label = "N")
```

Arguments

theta	simulation parameters
lam	mean number of spheroids per unit volume
size	size="const" (default) or name of random generating function a for specific size distribution
shape	either shape="const" as a constant portion of the axis length or shape="rbeta" as beta distributed shape factor. For a radius distribution use your own joint distribution, see details.
orientation	name of random generating function for orientation distribution
rjoint	name of joint random generating function
box	simulation box
mu	main orientation axis, mu=c(0,1,0) (default)
perfect	logical: perfect=TRUE (default) simulate perfect
pl	optional: print level
label	optional: set a label to all generated spheroids

Details

The function simulates a Poisson (sphero)cylinder system according to the supplied simulation parameter `theta` in a predefined simulation box. The argument `size` is of type string and denotes the major-axis length random generating function name.

For the directional orientation of the cylinder axis one has the choice of a uniform (`runifdir`), isotropic random planar (`rbetaiso`, see reference) or von Mises-Fisher (`rvMisesFisher`) distribution. The simulation box is a list containing of vector arguments which correspond to the lower and upper points in each direction. If the argument `box` has only one element, i.e. `list(c(0,1)`, the same extent is used for the other dimensions. If `rjoint="rmulti"` names a joint random generating function then argument `size` is ignored (see example file "sim.R"). For the purpose of exact simulation setting `size` equal to `rbinorm` declares a bivarite size-shape distribution which leads to a lognormal distributed cylinder axis (named `u`) half length h (without caps) and a scaled radius `r`. If $[X, Y]$ follow a bivariate normal distribution with correlation parameter ρ then $h = 2.0 * exp(x)$ defines the sample cylinder axis length together with the scaled radius $r = 0.5 * h * s$ and shape parameter set to $s = 1/(1 + exp(-y))$. The parameter ρ defines the degree of correlation between the cylinder axis length and cylinder radius which must be provided as part of the list of simulation parameters `theta`. The method of exact simulation is tailored to the above described model. For a general approach please see the given reference below. Other (univariate) cylinder axis lengths types include the beta, gamma, lognormal and uniform distribution where the shape factor to get the radius either follows a beta distribution or is set to a constant. Despite the case of constant size simulations all other simulations are done as perfect simulations. The current implementation does not include routines for unfolding the joint 3d size-shape-orientation distribution of cylinders so far. However, this feature this might be provided in a later version.

The argument `pl` denotes the print level of output information during simulation. Currently, only `pl`=0 for no output and `pl`>100 for some additional info is implemented.

Value

list of cylinders

References

- Ohser, J. and Schladitz, K. 3D images of materials structures Wiley-VCH, 2009
- C. Lantuéjoul. Geostatistical simulation. Models and algorithms. Springer, Berlin, 2002. Zbl 0990.86007

Examples

```
## Not run:

lam <- 10
box <- list("xrange"=c(0,3),"yrange"=c(0,3),"zrange"=c(0,9))

# cylinders of constant length
theta <- list("size"=list(0.25),
  "shape"=list(0.5),
  "orientation"=list("kappa"=1))

S <- simCylinderSystem(theta,lam,size="const", shape="const",
```

```
orientation="rbetaiso",box=box,pl=101)

# cylinders of constant length with
# beta distributed radius
theta <- list("size"=list(0.35),
              "shape"=list("a"=1,"b"=5),
  "orientation"=list("kappa"=1.5))

S <- simCylinderSystem(theta,lam,size="const", shape="rbeta",
orientation="rbetaiso",box=box,pl=101)

# bivariate length-shape distribution
# possibly correlated
param <- list("mx"=-1.0,"my"=-2.5, "sdx"=0.15,"sdy"=0.2,"rho"=0.0,"kappa"=1.0)
theta <- list("size"=list("mx"=param$mx,"sdx"=param$sdx,
  "my"=param$my,"sdy"=param$sdy,
  "rho"=param$rho),
  "orientation"=list("kappa"=param$kappa),
  "shape"=list())

S <- simCylinderSystem(theta,lam,size="rbinorm",orientation="rbetaiso",box=box,pl=101)

## show cylinder system
#cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
#cylinders3d(S, box, col=cols)


## End(Not run)
```

`simModel3d` *Simulate spheroid system*

Description

Simulate a spheroid system by perfect simulation

Usage

```
simModel3d(param, cond)
```

Arguments

`param`	parameters
`cond`	condition object

Details

Simulate a spheroid system by perfect simulation with log normal sizes and transformed shape parameter, see `simSpheroidSystem`. This function is intended to be a condensed version of `simSpheroidSystem` just for ease of use of exact simulation with random planar orientation of spheroids.

Value

spheroid system

Examples

```
## Not run:

# directional distribution
rbetaiso <- function(kappa) {
   phi <- runif(1,0,1)*2*pi
   q <- runif(1,0,1)
   theta=acos((1-2*q)/sqrt(kappa*kappa-(1-2*q)*(1-2*q)*(kappa*kappa-1)))
   list("u"=c(sin(theta)*cos(phi),sin(theta)*sin(phi),cos(theta)),
   "theta"=theta,"phi"=phi)
}

# no perfect simualtion here for 'rmulti'
# multivariate size distribution and orientation distribution
rmulti <- function(m,s,kappa) {
   dir <- rbetaiso(kappa)
   M <- chol(s, pivot = TRUE)
   M <- M[, order(attr(M, "pivot"))]
   x <- exp(matrix(m,nrow=1) +
          matrix(rnorm(ncol(s)), nrow = 1, byrow = TRUE) %*%M)
   a <- min(x)
   b <- max(x)

   list("a"=a,"b"=b,"u"=dir$u,"shape"=a/b,
        "theta"=dir$theta, "phi"=dir$phi)

}

set.seed(1234)
lam <- 100
box <- list("xrange"=c(0,5),"yrange"=c(0,5),"zrange"=c(0,5))

# joint (user defined random generating function) simualtion
sigma <- matrix(c(0.1,0.1,0.1,0.25), ncol=2)
theta <- list("m"=c(-3.0,-2.0),"s"=sigma,"kappa"=0.5)
S <- simSpheroidSystem(theta,lam,rjoint="rmulti",box=box,pl=101)

# Spheroids with log normal distributed semi-major axis length
# distribution and independent orientation distribution
theta <- list("size"=list("meanlog"=-2.5,"sdlog"=0.5),
              "shape"=list("s"=0.5),
              "orientation"=list("kappa"=1.5))

S <- simSpheroidSystem(theta,lam,size="rlnorm",shape="const",orientation="rbetaiso",
box=box,pl=101)

# log normal, rbeta shape
theta <- list("size"=list("meanlog"=-2.5,"sdlog"=0.5),
```

```
  "shape"=list("a"=1,"b"=2),
      "orientation"=list("kappa"=1.5))

S <- simSpheroidSystem(theta,lam,size="rlnorm",shape="rbeta",orientation="rbetaiso",
box=box,pl=101)

# Spheroids of constant sizes, const shape
theta <- list("size"=list(0.25),
              "shape"=list("s"=0.5),
              "orientation"=list("kappa"=1))
S <- simSpheroidSystem(theta,lam,size="const",shape="const",orientation="rbetaiso",
                       box=box,pl=101)

# constant size, rbeta shape
theta <- list("size"=list(0.25),
  "shape"=list("a"=1,"b"=2),
  "orientation"=list("kappa"=1.5))

S <- simSpheroidSystem(theta,lam,size="const",shape="rbeta",orientation="rbetaiso",
   box=box,pl=101)


# Spheroids exact simulation
param <- list("mx"=-1.0,"my"=-2.5, "sdx"=0.15,"sdy"=0.2,"rho"=0.0,"kappa"=1.0)
theta <- list("size"=list("mx"=param$mx,
      "sdx"=param$sdx,
  "my"=param$my,
  "sdy"=param$sdy,
  "rho"=param$rho),
  "orientation"=list("kappa"=param$kappa),
  "shape"=list())

S <- simSpheroidSystem(theta,lam,size="rbinorm",orientation="rbetaiso",box=box,pl=101)


## show spheroid system
#cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
#spheroids3d(S[1:500],box=box, col=cols)


## End(Not run)
```

---

simSphereSystem *Simulation of sphere system*

Description

The function simulates a Poisson sphere system of intensity `lam` where each sphere center is uniformly distributed in a box. The function returns a list of spheres with elements `id`, `center` and radius `r`.

Usage

```
simSphereSystem(theta, lam, rdist, box = list(c(0, 1)), perfect = TRUE,
  pl = 0, label = "N")
```

Arguments

theta	simulation parameters
lam	mean number of spheres per unit volume
rdist	string, radii random generating function name
box	simualtion box
perfect	logical: perfect=TRUE (default) simulate perfect
pl	print level
label	some character as a label, 'N' (default)

Details

Any random generating function, passed as a name, for the radii distribution is accepted as long as the formal function parameter names match the actual parameter names exactly as defined in the parameter list theta.

The simulation box is of type list. The vector arguments correspond to the lower and upper points in x,y and z direction. If box has only one element, i.e. list(c(0,1), the same extent is used for the other dimensions. The argument pl denotes the print level of information during simulation. Currently, only pl=0 for no output and pl>100 is implemented. Argument cond$rdist is of type string naming the (user defined) radii random generating function. Setting size equal to 'rlnorm' generates log normally distributed radii for a stationary Poisson ball system according to a general approach of perfect simulation (see reference below). Other distributions currently available are the beta, gamma and uniform distribution.

Value

list of class spheres if pl>100 or empty list

References

- C. Lantuéjoul. Geostatistical simulation. Models and algorithms. Springer, Berlin, 2002. Zbl 0990.86007

Examples

```
theta <- list("meanlog"=-2.5,"sdlog"=0.2)
S <- simSphereSystem(theta,lam=1000,rdist="rlnorm",pl=101)
```

simSpheroidSystem *Simulation of spheroid system*

Description

Simulation of Poisson spheroid system

Usage

```
simSpheroidSystem(theta, lam, size = "const", shape = "const",
  orientation = "rbetaiso", stype = c("prolate", "oblate"), rjoint = NULL,
  box = list(c(0, 1)), mu = c(0, 0, 1), perfect = TRUE, pl = 0,
  label = "N")
```

Arguments

theta	simulation parameters
lam	mean number of spheroids per unit volume
size	name of random generating function for size distribution
shape	shape="const" (default) as a constant shape
orientation	name of random generating function for orientation distribution
stype	spheroid type
rjoint	name of joint random generating function
box	simulation box
mu	main orientation axis, mu=c(0,0,1) (default)
perfect	logical: perfect=TRUE (default) simulate perfect
pl	optional: print level
label	optional: set a label to all generated spheroids

Details

The function simulates a Poisson spheroid system according to the supplied simulation parameter theta in a predefined simulation box. The argument size is of type string and denotes the major-axis length random generating function name.

Further the function simulates either stype="prolate" or stype='oblate' spheroids. For the directional orientation of the spheroid's major-axis one has the choice of a uniform (runifdir), isotropic random planar (rbetaiso, see reference) or von Mises-Fisher (rvMisesFisher) distribution. The simulation box is a list containing of vector arguments which correspond to the lower and upper points in each direction. If the argument box has only one element, i.e. list(c(0,1), the same extent is used for the other dimensions. If rjoint names a joint random generating function then argument size is ignored. For the purpose of exact simulation setting size equal to rbinorm declares a bivarite size-shape distribution which leads to a log normal distributed semi-major axis a and a scaled semi-minor axis length c. If $[X, Y]$ follow a bivariate normal distribution with correlation

parameter ρ then $a = exp(x)$ defines the sample semi-major axis length together with the scaled semi-minor axis length $c = a * s$ and shape parameter set to $s = 1/(1 + exp(-y))$. The parameter ρ defines the degree of correlation between the semi-axes lengths which must be provided as part of the list of simulation parameters `theta`. The method of exact simulation is tailored to the above described model. For a general approach please see the given reference below. Other (univariate) major-axis lengths types include the beta, gamma, lognormal and uniform distribution where the shape factor which determines the minor-axis length either follows a beta distribution or is set to a constant. Despite the case of constant size simulations all other simulations are done as perfect simulations.

The argument `pl` denotes the print level of output information during simulation. Currently, only `pl=0` for no output and `pl>100` for some additional info is implemented.

Value

list of spheroids either of class `prolate` or `oblate`

References

- Ohser, J. and Schladitz, K. 3D images of materials structures Wiley-VCH, 2009
- C. Lantuéjoul. Geostatistical simulation. Models and algorithms. Springer, Berlin, 2002. Zbl 0990.86007

Examples

```
## Not run:

# directional distribution
rbetaiso <- function(kappa) {
   phi <- runif(1,0,1)*2*pi
   q <- runif(1,0,1)
   theta=acos((1-2*q)/sqrt(kappa*kappa-(1-2*q)*(1-2*q)*(kappa*kappa-1)))
   list("u"=c(sin(theta)*cos(phi),sin(theta)*sin(phi),cos(theta)),
   "theta"=theta,"phi"=phi)
}

# no perfect simualtion here for 'rmulti'
# multivariate size distribution and orientation distribution
rmulti <- function(m,s,kappa) {
   dir <- rbetaiso(kappa)
   M <- chol(s, pivot = TRUE)
   M <- M[, order(attr(M, "pivot"))]
   x <- exp(matrix(m,nrow=1) +
          matrix(rnorm(ncol(s)), nrow = 1, byrow = TRUE) %*%M)
   a <- min(x)
   b <- max(x)

   list("a"=a,"b"=b,"u"=dir$u,"shape"=a/b,
        "theta"=dir$theta, "phi"=dir$phi)

}
```

```
set.seed(1234)
lam <- 100
box <- list("xrange"=c(0,5),"yrange"=c(0,5),"zrange"=c(0,5))

# joint (user defined random generating function) simualtion
sigma <- matrix(c(0.1,0.1,0.1,0.25), ncol=2)
theta <- list("m"=c(-3.0,-2.0),"s"=sigma,"kappa"=0.5)
S <- simSpheroidSystem(theta,lam,rjoint="rmulti",box=box,pl=101)

# Spheroids with log normal distributed semi-major axis length
# distribution and independent orientation distribution
theta <- list("size"=list("meanlog"=-2.5,"sdlog"=0.5),
              "shape"=list("s"=0.5),
              "orientation"=list("kappa"=1.5))

S <- simSpheroidSystem(theta,lam,size="rlnorm",shape="const",orientation="rbetaiso",
box=box,pl=101)

# log normal, rbeta shape
theta <- list("size"=list("meanlog"=-2.5,"sdlog"=0.5),
  "shape"=list("a"=1,"b"=2),
      "orientation"=list("kappa"=1.5))

S <- simSpheroidSystem(theta,lam,size="rlnorm",shape="rbeta",orientation="rbetaiso",
box=box,pl=101)

# Spheroids of constant sizes, const shape
theta <- list("size"=list(0.25),
              "shape"=list("s"=0.5),
              "orientation"=list("kappa"=1))
S <- simSpheroidSystem(theta,lam,size="const",shape="const",orientation="rbetaiso",
                       box=box,pl=101)

# constant size, rbeta shape
theta <- list("size"=list(0.25),
  "shape"=list("a"=1,"b"=2),
  "orientation"=list("kappa"=1.5))

S <- simSpheroidSystem(theta,lam,size="const",shape="rbeta",orientation="rbetaiso",
   box=box,pl=101)


# Spheroids exact simulation
param <- list("mx"=-1.0,"my"=-2.5, "sdx"=0.15,"sdy"=0.2,"rho"=0.0,"kappa"=1.0)
theta <- list("size"=list("mx"=param$mx,
      "sdx"=param$sdx,
  "my"=param$my,
  "sdy"=param$sdy,
  "rho"=param$rho),
  "orientation"=list("kappa"=param$kappa),
  "shape"=list())

S <- simSpheroidSystem(theta,lam,size="rbinorm",orientation="rbetaiso",box=box,pl=101)
```

```
## show spheroid system
#cols <- c("#0000FF","#00FF00","#FF0000","#FF00FF","#FFFF00","#00FFFF")
#spheroids3d(S[1:500],box=box, col=cols)


## End(Not run)
```

---

spheroids3d *Plot particle system*

Description

Draw particle system as defined by S.

Usage

```
spheroids3d(S, box, draw.axes = FALSE, draw.box = TRUE, draw.bg = TRUE,
  bg.col = "white", clipping = FALSE, ...)
```

Arguments

S	a list of spheroids
box	simulation box
draw.axes	logical: if true, draw the axes
draw.box	logical: if true, draw the bounding box
draw.bg	logical: if true, draw the a gray background box
bg.col	background color used to draw the box background
clipping	logical: if true clip to the bounding box
...	further material properties passed to 3d plotting functions

Details

The function requires the package rgl to be installed.

trivarHist *Trivariate histogram*

Description

Plot trivariate histogram of size, shape, orientation distribution

Usage

```
trivarHist(A, main = paste("Trivariate Histogram"), scale = 0.5, col, ...)
```

Arguments

A	3d array of count data
main	main title
scale	factor to scale the spheres
col	vector of color values repeatedly used
...	optional graphic arguments

Details

The (estimated spatial) joint size-shape-orientation distribution is plotted in a box with corresponding axes shown. The axes intersect in the first class number. The ball volumes visualize the relative frequencies of count data for each class which can be scaled by the user for non-overlapping spheres. Balls within the same size class have the same color.

Value

the graphical output: trivariate histogram

unfold *Stereological unfolding*

Description

Unfolding the (joint) distribution of planar parameters

Usage

```
unfold(sp, nclass, maxIt = 64, nCores = getOption("par.unfoldr", 1), ...)
```

Arguments

sp	section profiles
nclass	number of classes
maxIt	maximum number of EM iterations
nCores	number of cpu cores used
...	optional arguments passed to setbreaks

Details

This is a S3 method for either trivariate stereological unfolding or estimation of 3d diameter distribution of spheres (Wicksell's corpuscle problem). The function aggregates all intermediate calculation steps required for the unfolding procedure given the data in the prescribed format and returning the parameters as count data in histogram form. The section profiles object sp, see sectionProfiles, is either of class prolate or oblate for the reconstruction of spheroids or in case of spheres the sp is simply a numeric vector of circle diameters. Here, the class of section profiles corresponds to the type of objects that will be reconstructed. The number of bin classes is set by the argument nclass which is either a scalar value in case of Wicksell's corpuscle problem or a vector of length three defined in the order of the number of size, angle and shape class limits. Using multiple cpu cores during the calculations is controlled by either setting the option 'par.unfoldr' to a user chosen amount of cores or by the argument nCores directly. The return value of the function is an object of class unfold whose arguments are as follows

- N_A (trivariate) histogram of section profile parameters
- N_V (trivariate) histogram of reconstructed parameters
- P array of coefficients
- breaks list of class limits for binning the parameter values

Value

object of class unfold

See Also

setbreaks, binning3d

---

updateIntersections *Updated spheroid intersections*

Description

Determine intersections of spheroids with bounding (simulation) box

Usage

```
updateIntersections(S, box)
```

Arguments

S	geometric objects system
box	the simulation box

Details

For a given list of spheroids, spheres or cylinders calculate if these objects intersect the simulation box or not.

Value

integer vector indicating intersection=1 or non intersection by 0

---

verticalSection *Spheroid vertical section*

Description

Vertical section of spheroid system

Usage

```
verticalSection(S, d, n = c(0, 1, 0), intern = FALSE)
```

Arguments

S	list of spheroids, see simSpheroidSystem
d	distance of intersecting plane to the origin
n	normal vector of intersting plane
intern	intern=FALSE (default) return all section profiles otherwise only those which have their centers inside the intersection window

Details

The function performs a vertical intersection defined by the normal vector n=c(0,1,0) which depends on the main orientation axis of the coordinate system and has to be parallel to this.

Value

list of size, shape and angle of section profiles

Index