

# Package ‘pgs’

January 2, 2012

**Version** 0.3-0

**Date** February 2010

**Title** Precision of Geometric Sampling

**Description** Computation of mean squared errors of stereological predictors.

**Author** Kien Kieu <Kien.Kieu@jouy.inra.fr>, Marianne Mora <Marianne.Mora@u-paris10.fr>

**Maintainer** Kien Kieu <Kien.Kieu@jouy.inra.fr>

**Depends** methods, gsl, R2Cuba

**License** CeCILL

**Encoding** latin1

**Repository** CRAN

**Date/Publication** 2010-02-09 09:16:51

## R topics documented:

area.mse . . . . .	2
area.mse.est . . . . .	3
BCRectLat3 . . . . .	4
content . . . . .	5
covariogram . . . . .	6
dvol.mse . . . . .	6
dvol.mse.est . . . . .	7
Ezeta . . . . .	8
FCRectLat3 . . . . .	9
FigLat . . . . .	10
FigLat-class . . . . .	11
FigLatData . . . . .	12
FigLatData-class . . . . .	12
Figure-class . . . . .	13
HexLat2 . . . . .	14

latscale . . . . .	14
LLat2 . . . . .	15
ltransf . . . . .	16
M . . . . .	17
pgs-internal . . . . .	17
PointPattern . . . . .	19
PointPattern-class . . . . .	19
PP2 . . . . .	20
PP3 . . . . .	21
PPBCRectLat3 . . . . .	22
PPFCRectLat3 . . . . .	23
PPHexLat2 . . . . .	24
PPQcxLat2 . . . . .	25
PPRectLat2 . . . . .	26
PPRectLat3 . . . . .	27
QcxLat2 . . . . .	28
QHexLat2 . . . . .	28
QQcxLat2 . . . . .	29
QRectLat2 . . . . .	30
Quadrat . . . . .	30
Quadrat-class . . . . .	31
RectLat2 . . . . .	32
RectLat3 . . . . .	32
scaling . . . . .	33
Segment . . . . .	34
Segment-class . . . . .	34
SHexLat2 . . . . .	35
SQcxLat2 . . . . .	36
SRectLat2 . . . . .	37
VecLat . . . . .	37
VecLat-class . . . . .	38
vol.mse . . . . .	39
vol.mse.est . . . . .	40
<b>Index</b>	<b>41</b>

---

area.mse

*MSE approximation for area predictors*

---

### Description

Compute a MSE approximation for area predictors. The structure of interest is an isotropic planar random compact set. The sampling device is a uniform random lattice of figures (point patterns, line segments, quadrats...). The approximation depends only on sampling parameters and on the mean perimeter (to be provided) of the structure.

**Usage**

```
area.mse(x, B = 1, L = 3)
```

**Arguments**

x a lattice of figures, object of class [FigLat-class](#).  
 B the mean perimeter. Default: 1.  
 L an integer, the criterion for stopping summation of the Epstein zeta function. Argument of the function [Ezeta](#). Default: 3.

**Value**

The MSE approximation as a numeric.

**References**

Kieu, K. and Mora, M. (2006). Precision of stereological planar area predictors. *J. Microsc.*, 222(3), 201-211.

**See Also**

[vol.mse](#), [dvol.mse](#).

**Examples**

```
# Sampling by a unit hexagonal point lattice
area.mse(PPHexLat2())
# Sampling by a unit square point lattice
area.mse(PPRectLat2())
# Sampling by a lattice of point patterns
area.mse(PPRectLat2(n=5, hp=0.1))
# Sampling by quadrats (may be slow)
## Not run: area.mse(QRectLat2(hq=0.5, vq=0.7))
# Sampling by a square lattice of segments (may be slow)
## Not run: area.mse(SRectLat2(end=c(0.5, 0.1)))
# Sampling by an hexagonal lattice of segments (may be slow)
## Not run: area.mse(SHexLat2(end=c(0.2, 0.15)))
```

---

area.mse.est

*MSE estimation for planar area predictors*

---

**Description**

Compute a MSE estimate for planar area predictors. The structure of interest is a random compact set. The sampling device is a uniform random lattice of figures.

**Usage**

```
area.mse.est(fldata,mse.only=TRUE,iso=FALSE,diff2use)
```

**Arguments**

<code>fldata</code>	data collected for planar area prediction, object of class <a href="#">FigLatData-class</a> .
<code>mse.only</code>	TRUE (default) if only the MSE estimate must be returned.
<code>iso</code>	FALSE (default) if the boundary is not assumed to be isotropic.
<code>diff2use</code>	a family of lattice vectors defining the data covariations to be used for the MSE estimation. A matrix with as many columns as lattice vectors. Each column contains the vector coordinates in the basis defined by the lattice generating matrix. Optional: a default value is computed. If isotropy is assumed, the lattice generating matrix (2 main basis vectors). Otherwise, 2 diagonal lattice vectors are added to the 2 main basis vectors.

**Details**

`area.mse.est` invokes function [area.mse](#).

**Value**

If `mse.only` is TRUE, the MSE estimate as a scalar. Otherwise a list with components:

<code>B.est</code>	estimate of the boundary length.
<code>deformation</code>	used when the boundary is not assumed to be isotropic. It is assumed that there exists an area preserving linear transformation which makes the boundary isotropic. This list component contains the estimated transformation matrix.
<code>mse.est</code>	MSE estimate.

**See Also**

[vol.mse.est](#), [dvol.mse.est](#).

---

BCRectLat3

*Generator of 3D body-centered rectangular lattices as VecLat objects*

---

**Description**

Create an object of class "VecLat" representing a 3D body-centered rectangular vector lattice.

**Usage**

```
BCRectLat3(dx=1,dy=dx,dz=dx)
```

**Arguments**

dx	spacing along the x-axis. Default: 1.
dy	spacing along the y-axis. Default: dx.
dz	spacing along the z-axis. Default: dy.

**Details**

Note that default parameter values do not define a body-centered cubic lattice. A 3D body-centered rectangular lattice is considered as a series of horizontal 2D rectangular vector lattices. The arguments dx and dy define the spacings in the horizontal planes. A 2D rectangular lattice seen in a given horizontal plane is shift halfway (in both horizontal directions) in the next plane. The argument dz defines the distance between consecutive horizontal planes. Therefore in order to define a (unit) body-centered cubic lattice, one should set dz=0.5.

**Value**

A [VecLat-class](#) object.

**See Also**

Generators [VecLat](#), [RectLat3](#), [FCRectLat3](#).

**Examples**

```
BCRectLat3()
BCRectLat3(1, 1, 3)
```

---

content

*Content of Figure objects*

---

**Description**

Generic function computing the content (cardinality, length, area, volume) of simple bounded subsets.

**Usage**

```
content(x)
```

**Arguments**

x an object of a class extending [Figure-class](#).

**Value**

The cardinality, length, area or volume of x depending of its class.

**See Also**

[Quadrat-class](#), [Segment-class](#), [PointPattern-class](#).

---

covariogram	<i>Geometric covariogram of Figure objects</i>
-------------	--

---

**Description**

Generic function for geometric covariogram computations. A geometric covariogram is a measure. The integral of a (test) function w.r.t. the geometric covariogram is computed.

**Usage**

```
covariogram(x, f, sym = FALSE)
```

**Arguments**

x	an object of a class extending <a href="#">Figure-class</a> .
f	a function. It must be defined on the space containing the figure. E.g. on the plane for a planar figure.
sym	boolean. True if f is symmetric. Default: FALSE.

**Value**

The numerical value of the integral of f w.r.t. the geometric covariogram of x.

**See Also**

[Quadrat-class](#), [Segment-class](#), [PointPattern-class](#).

---

dvol.mse	<i>MSE approximation for d-dimensional volume predictors</i>
----------	--

---

**Description**

Compute a MSE approximation for d-dimensional volume predictors. The structure of interest is an isotropic d-dimensional random compact set. The sampling device is a uniform random lattice of figures (point patterns, line segments...). The approximation depends only on sampling parameters and on the mean surface (to be provided) of the structure.

**Usage**

```
dvol.mse(x, S = 1, L = 3)
```

**Arguments**

x	a lattice of figures, object of class <a href="#">FigLat-class</a> .
S	the mean surface. Default: 1.
L	an integer, the criterion for stopping summation of the Epstein zeta function. Argument of the function <a href="#">Ezeta</a> . Default: 3.

**Value**

The MSE approximation as a numeric.

**See Also**

[area.mse](#), [vol.mse](#).

**Examples**

```
# Area prediction. Sampling by a lattice of point patterns
area.mse(FigLat(2,RectLat2()),PP2(5,0.1))

# Volume prediction. Sampling by a unit cubic point lattice
vol.mse(FigLat(3,VecLat(diag(3)),PointPattern(rep(0,3))))
```

---

dvol.mse.est

---

*MSE estimation for volume predictors*


---

**Description**

Compute a MSE estimate for volume predictors in a space with arbitrary dimension. The structure of interest is an random compact set. The sampling device is a uniform random lattice of figures.

**Usage**

```
dvol.mse.est(fldata,mse.only=TRUE,iso=FALSE,diff2use)
```

**Arguments**

fldata	data collected for volume prediction, object of class <a href="#">FigLatData-class</a> .
mse.only	TRUE (default) if only the MSE estimate must be returned.
iso	FALSE (default) if the boundary is not assumed to be isotropic.
diff2use	a family of lattice vectors defining the data covariations to be used for the MSE estimation. A matrix with as many columns as lattice vectors. Each column contains the vector coordinates in the basis defined by the lattice generating matrix.

**Value**

If `mse.only` is `TRUE`, the MSE estimate as a scalar. Otherwise a list with components:

<code>S.est</code>	estimate of the boundary surface area.
<code>deformation</code>	used when the boundary is not assumed to be isotropic. It is assumed that there exists a volume preserving linear transformation which makes the boundary isotropic. This list component contains the estimated transformation matrix.
<code>mse.est</code>	MSE estimate.

**See Also**

[vol.mse.est](#), [area.mse.est](#).

---

Ezeta

*Epstein zeta function*

---

**Description**

Numerical computation of the Epstein zeta function.

**Usage**

```
Ezeta(s, vlat, h = rep(0, vlat@dimspace), L = 3, prepare = FALSE, norm = TRUE)
```

**Arguments**

<code>s</code>	the exponent parameter as a numeric. See details below.
<code>vlat</code>	a vector lattice as a <a href="#">VecLat-class</a> object.
<code>h</code>	a phase vector or a matrix of phase column vectors.
<code>L</code>	the stopping criterion for the numerical approximation of the Epstein zeta function. Default: 3. Increase L for better precision.
<code>prepare</code>	a logical or a list.
<code>norm</code>	logical. Should the phase be normalized? Default: <code>TRUE</code> . See details below.

**Details**

The Epstein zeta function is a multidimensional version of the Riemann zeta function defined as the sum of

$$\frac{\exp(-2\pi I\langle h, x \rangle)}{\|x\|^s}$$

for all non-null vectors  $x$  of the lattice.

When considered as a function of the phase  $h$ , the Epstein zeta function is invariant under any translation by a lattice vector. The phase vector  $h$  provided to `Ezeta` must lie in the fundamental tile of the vector lattice `vlat`. If `norm` is `TRUE`,  $h$  is automatically normalized.

The algorithm used for computation of the Epstein zeta function is provided in a paper by Richard E. Crandall, see reference below. In this implementation, all preliminary computations not depending on the phase  $h$  can be made separately.

**Value**

If prepare is FALSE, the result as a numeric. If prepare is TRUE, preliminary computations not depending on the phase are returned as a list. If prepare is a list as computed when prepare is TRUE, the final result as a numeric.

**References**

Crandall, R.E. (1998). *Fast evaluation of Epstein zeta functions*. Manuscript. <http://www.reed.edu/~crandall/papers/epstein.pdf>

**See Also**

[VecLat-class](#)

**Examples**

```
Ezeta(3,RectLat2(),h=c(1.1,3.8))
Ezeta(3,HexLat2())
```

---

FCRectLat3

---

*Generator of 3D face-centered rectangular lattices as VecLat objects*


---

**Description**

Create an object of class "VecLat" representing a 3D face-centered rectangular vector lattice.

**Usage**

```
FCRectLat3(d=1,dx=sqrt(2)*d,dz=d)
```

**Arguments**

d	the distance between two neighbour diagonal locations on the horizontal plane. Default: 1.
dx	the distance between two neighbour locations along the x-axis on the horizontal plane. Default: $\sqrt{2}d$ .
dz	the vertical spacing between two neighbour horizontal planes. Default: d.

**Details**

Note that the default arguments do not define a face-centered cubic lattice. A face-centered rectangular lattice is considered as a series of horizontal quincunx 2D lattices. The arguments d and dx define the spacings of the horizontal quincunx. The default values of d and dx yields unit horizontal quincunx lattices. A quincunx lying in a given horizontal plane is shifted halfway in a single direction into the next horizontal plane. The argument dz defines the distance separating two consecutive horizontal planes. The unit face-centered cubic lattice is obtained with  $d = \sqrt{2}/2$ ,  $dx = 1$  and  $dz = 0.5$ .

**Value**

A [VecLat-class](#) object.

**See Also**

Generators [VecLat](#), [RectLat3](#), [BCRectLat3](#).

**Examples**

```
FCRectLat3()
```

---

FigLat

*Generator of FigLat objects*

---

**Description**

Create an object of class "FigLat" representing a lattice of figures.

**Usage**

```
FigLat(d, vlat, fig, lmat = matrix(0, nrow = d, ncol = 1))
```

**Arguments**

d	the dimension of the space where the lattice of figures lies.
vlat	a <a href="#">VecLat-class</a> object representing the (translation) vector lattice.
fig	a <a href="#">Figure-class</a> object representing the figure.
lmat	a matrix generating the $L$ vector subspace.

**Value**

A [FigLat-class](#) object.

**Examples**

```
# Square lattice of quadrats
FigLat(2,RectLat2(),Quadrat(.5,.5))
# Lattice of horizontal lines
FigLat(2,VecLat(c(0,1)),PointPattern(rep(0,2)),lmat=c(1,0))
```

---

FigLat-class

Class "FigLat"

---

### Description

Objects of class "FigLat" represent lattices of figures of the form  $\Lambda + L + F$ , where  $\Lambda$  is a (translation) vector lattice,  $L$  is a vector subspace (may be reduced to the null vector) and  $F$  is a bounded figure.

### Details

The vector subspace  $L$  should be perpendicular to the support of the vector lattice  $\Lambda$  and to the figure  $F$ .

### Objects from the Class

Objects can be created by calls of the form `new("FigLat", ...)` or using generators such as [FigLat](#).

### Slots

`vlat`: Object of class "VecLat" representing  $\Lambda$ .

`fig`: Object of class "Figure" representing  $F$ .

`lmat`: Object of class "matrix" representing  $L$ .

### Methods

**ltransf** signature(`x = "FigLat"`, `m = "matrix"`): [ltransf](#) for a FigLat object.

**plot** signature(`x = "FigLat"`, `y = "missing"`): plot a FigLat object. Implemented only for planar figures of lattices. The arguments `xlim` and `ylim` are mandatory in order to define the plot region. If the extra argument `add` (default: FALSE) is TRUE, the lattice of figures is added to the current plot.

**print** signature(`x = "FigLat"`): print a FigLat object.

**scaling** signature(`x = "FigLat"`, `s = "numeric"`): [scaling](#) for a FigLat object.

### See Also

Generator [FigLat](#).

---

FigLatData	<i>FigLatData-class generator</i>
------------	-----------------------------------

---

**Description**

Generator for "FigLatData-class" objects.

**Usage**

```
FigLatData(figlat,array)
```

**Arguments**

figlat	a lattice of figures as a <a href="#">FigLat-class</a> object.
array	a data array. The number of array dimensions must be equal to figlat@vlat@dim <sub>supp</sub> (dimension of the vector lattice support) or to figlat@vlat@dim <sub>supp</sub> +1 if there are more than one sampled structures.

**Value**

A FigLatData-class object

**See Also**

[FigLatData-class](#).

---

FigLatData-class	<i>Class "FigLatData"</i>
------------------	---------------------------

---

**Description**

Objects of class "FigLatData" store data associated with the figures of a figure lattice.

**Slots**

**flat** a "FigLat-class" object defining the figure lattice.

**data** an array containing the figure data. Each element of data stores the content of a figure. For instance, in 2D, data[i, j, k] is the content of the figure lying at  $ie_1 + je_2$  where  $e_1$  is the first basis vector of the lattice and  $e_2$  is the second basis vector. The last index k numbers the sampled structure.

**Methods**

**print** signature("FigLatData"): print slot data.

**dataCovariogram** signature(x="FigLatData", coord="matrix"): compute the empirical covariogram of data associated with figures in a figure lattice. The matrix columns of coord contain the coordinates (with respect to the vector lattice generating matrix) of shift vectors. The empirical covariogram is returned as a matrix. The first columns contain the shift vector coordinates (with respect to the canonical basis), the last column the empirical covariogram values.

**See Also**

[FigLatData](#)

---

Figure-class

Class "Figure"

---

**Description**

Objects of class "Figure" represent simple geometric bounded subsets (such as rectangles, segments, point patterns) in spaces with arbitrary dimensions. The subsets are defined up to a translation. See extensions such as [Quadrat](#), [Segment](#) or [PointPattern](#).

**Objects from the Class**

Objects can be created by calls of the form `new("Figure", ...)`.

**Slots**

**dimspace**: Object of class "numeric". Dimension of the space containing the figure. For instance, 2 for a planar figure.

**coord**: Object of class "matrix". Each column contains the Cartesian coordinates of special points which define the figure. For instance, the end of a segment starting at the origin.

**Methods**

**ltransf** signature(x = "Figure", m = "matrix"): [ltransf](#) for a figure.

**scaling** signature(x = "Figure", s = "numeric"): [scaling](#) for a figure.

**Note**

Objects of class "Figure" should not be created directly. Use extensions (see below).

**See Also**

Extensions from Figure class: [Quadrat-class](#), [Segment-class](#), [PointPattern-class](#).

---

HexLat2

*Generator of 2D hexagonal lattices as VecLat objects*


---

**Description**

Create an object of class "VecLat" representing a planar hexagonal vector lattice.

**Usage**

```
HexLat2(det=1)
```

**Arguments**

det                    the area of a fundamental tile of the hexagonal lattice. Default: 1.

**Value**

A [VecLat-class](#) object.

**See Also**

Generators [VecLat](#), [RectLat2](#), [QcxLat2](#).

**Examples**

```
HexLat2()
HexLat2(sqrt(3)/8)
```

---

latscale

*Sampling design for planar area prediction*


---

**Description**

Planar area can be predicted based on sampling by a lattice of figures  $u\Lambda + L + F$ . The function `latscale` computes the scaling parameter  $u$  such that the prediction coefficient of error is equal to a given value.

**Usage**

```
latscale(x,A,shape,CE.n,upper,maxiter=100,tol=.Machine$double.eps^0.25,
        lower=.Machine$double.eps ^ 0.5,L=3,only.root=TRUE)
```

**Arguments**

x	the lattice of figures as a <code>FigLat</code> object. The vector lattice <code>x@vlat</code> must be unit.
A	a (rough) estimate of the mean area.
shape	a (rough) estimate of the shape parameter $B/\sqrt{A}$ where $B$ is the mean perimeter.
CE.n	the given value of the prediction coefficient of error.
lower	the lower point of the interval where the scaling parameter is to be searched. Argument of the function <code>uniroot</code> . Default: <code>.Machine\$double.eps ^ 0.5</code> .
upper	the upper point of the interval where the scaling parameter is to be searched. Argument of the function <code>uniroot</code> .
maxiter	other argument passed to the function <code>uniroot</code> .
tol	other argument passed to the function <code>uniroot</code> . Default: <code>.Machine\$double.eps^0.25</code> .
L	an integer, the criterion for stopping summation of the Epstein Zeta function. Default: 3.
only.root	a Boolean controlling the returned value, see below. Default: TRUE.

**Value**

If `only.root` is TRUE, the function returns the numeric value of the scaling parameter  $u$ . Else, the function returns a list with four components: `scale` the numeric value of  $u$ , `CE` the coefficient of error computed for  $u$ , `iter` the number of iterations used, `prec` an approximate estimated precision for  $u$ .

**Examples**

```
latscale(FigLat(2,RectLat2(),PointPattern(rep(0,2))),A=1,shape=5,CE.n=0.05,upper=2,only.root=FALSE)
```

---

 LLat2

*Generator of 2D lattices of lines*


---

**Description**

Create a `FigLat`-class object representing a lattice of lines in the plane.

**Usage**

```
LLat2(delta=1,theta=0)
```

**Arguments**

delta	the distance between two neighbour lines. Default: 1.
theta	the angle defining the lines orientation. Default: 0 (horizontal lines).

**Value**

A [FigLat-class](#) object.

**See Also**

Generator [FigLat](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [PPRectLat2](#), [QHexLat2](#), [QQcxLat2](#), [QRectLat2](#), [SHexLat2](#), [SQcxLat2](#), [SRectLat2](#).

**Examples**

```
# Unit lattice of horizontal lines
LLat2()
# Lattice of vertical lines
LLat2(0.5,pi/2)
```

---

ltransf

---

*Linear transform of Figure, VecLat or FigLat objects*


---

**Description**

Linear transform of a subset.

**Usage**

```
ltransf(x, m)
```

**Arguments**

**x**                    a [Figure-class](#), [VecLat-class](#) or [FigLat-class](#) object.  
**m**                    a matrix defining the linear transform.

**Value**

An object of the same class as **x**.

**See Also**

[Figure-class](#), [VecLat-class](#), [FigLat-class](#).

---

M *Compute the M-function for a lattice of figures*

---

### Description

The M-function is the integral of the Epstein zeta function associated with the lattice with respect to the geometric covariogram of the figure.

### Usage

`M(vlat, fig, s, L)`

### Arguments

`vlat` a vector lattice, object of class [VecLat-class](#).  
`fig` a figure, object of class [Figure-class](#).  
`s` the exponent of the Epstein zeta function.  
`L` an integer, the criterion for stopping summation of the Epstein Zeta function. Argument of the function [Ezeta](#).

### Value

The result as a numeric.

### References

Kieu, K. and Mora, M. (2006). Precision of stereological planar area predictors. *J. Microsc.*, 222(3), 201-211.

---

pgs-internal *Internal pgs functions*

---

### Description

Internal pgs functions. These functions should not be called by most users.

### Usage

```
cubicgrid(s,d=2)
dualmat(x)
gaic(a,x)
in.upper.halfspace(x)
normphase(h,E,Einv=solve(E))
pointcov(x,tol=.Machine$double.eps ^ 0.5)
sphereSurface(d=2)
ellipsoidSurface(a)
```

**Arguments**

s	a list of vectors or a vector.
d	an integer, the space dimension. Default: 2.
x	an array, a matrix or a vector.
a	a numeric.
h	a vector or a matrix.
E	a matrix.
Einv	a matrix.
tol	a numeric.

**Details**

`cubicgrid` computes the coordinates of the points of the Cartesian product set  $s_1 \times \dots \times s_d$  where the  $s_i$ 's are subsets of reals. The  $s_i$ 's may be provided as a list of vectors or a vector  $s$ . In the latter case, the vector  $s$  is replicated  $d$ -times in the list `list(s,...,s)`.

`dualmat` computes the dual of a non-singular square matrix  $x$ .

`gaic` computes the incomplete gamma function with parameter  $a$  for a vector or an array of reals  $x$ .

`in.upper.halfspace` is used to test if the vector  $x$  is in the upper half-space. Upper half-space: null vector or last non-zero coordinate is greater than 0.

`normphase` normalizes the vector  $h$  with respect to a vector lattice. The lattice is defined by its generating matrix  $E$ . The normalized vector lies inside the fundamental tile defined by  $E$  and differs from  $h$  by a lattice vector. When  $h$  is a matrix, the transformation is applied to each column vector.

`pointcov` computes the set  $S-S$  for a given finite set of points  $S$ . The points are defined as the columns of a matrix  $x$ . The result is given as a list with two components  $ud$  and  $n$ . The component  $ud$  contains the points of  $S-S$  lying in the upper half-space ( $S-S$  is symmetric). The component  $n$  provides multiplicities:  $n[i]$  is the multiplicity of  $ud[, i]$ . The parameter `tol` controls the precision level in comparisons.

`sphereSurface` computes the surface area of the unit sphere in the  $d$ -dimensional space.

`ellipsoidSurface` computes the surface area of an ellipsoid in a space of arbitrary dimension. The argument  $a$  is a vector containing the semi-axis lengths. The algorithm is based on Garry Tee (2005) Surface area and capacity of ellipsoids in  $n$  dimensions. *New Zealand Journal of Mathematics*, 34(2), 165–198. It involves numerical one-dimensional integration.

**Examples**

```
cubicgrid(c(1,2,5))
dualmat(diag(1:2))
gaic(1.2,seq(0.5,10,length=10))
in.upper.halfspace(c(-1,0))
in.upper.halfspace(c(1,0))
normphase(c(1.8,1.5),diag(1:2))
pointcov(matrix(c(0,0,1,0,1,1,0,1,1/2,1/2),nrow=2))
sphereSurface(2)
```

---

PointPattern	<i>Generator of PointPattern objects</i>
--------------	--

---

**Description**

Generate a PointPattern-class object representing a finite point pattern.

**Usage**

```
PointPattern(coord)
```

**Arguments**

coord	a matrix or a vector. If coord is a vector, it is transformed into a matrix with one column. Each column contains the Cartesian coordinates of a point. Hence the number of rows of coord defines the dimension of the space.
-------	---

**Value**

An object of [PointPattern-class](#).

**See Also**

[PointPattern-class](#), generators [PP2](#), [PP3](#), generators of higher-dimensional figures [Quadrat](#), [Segment](#).

**Examples**

```
## Random planar point pattern
PointPattern(matrix(runif(10),2,5))
## Planar point pattern which consists of a single point
PointPattern(rep(0,2))
## Random point pattern on the line
PointPattern(matrix(runif(5),1,5))
```

---

PointPattern-class	<i>Class "PointPattern"</i>
--------------------	-----------------------------

---

**Description**

Extend class "Figure". Representation of finite point patterns.

**Objects from the Class**

Objects can be created by calls of the form `new("PointPattern", ...)`. It is recommended to use the generator [PointPattern](#).

**Slots**

**dimspace:** Object of class "numeric". The dimension of the space where the points lie.

**coord:** Object of class "matrix". The matrix columns contain the Cartesian coordinates of the points.

**Extends**

Class "Figure", directly.

**Methods**

**content** signature(x = "PointPattern"): compute the number of points.

**covariogram** signature(x = "PointPattern", f = "function"): [covariogram](#) for "PointPattern" objects.

**plot** signature(x = "PointPattern", y = "missing"): plot a PointPattern object. Implemented only for planar point patterns. The extra argument origin (default: null vector) may be used to plot the point pattern translated by the vector origin. If the extra argument add (default: FALSE) is TRUE, the point pattern is added to the current plot.

signature(x = "PointPattern", y = "matrix"): plot the pattern of parallel lines defined as the Minkowski sum of x and the line spanned by the y. Only implemented in 2D. Extra arguments origin and add are described above.

**print** signature(x = "PointPattern"): print a PointPattern object.

**See Also**

Generators [PointPattern](#), [Segment-class](#), [Quadrat-class](#).

---

 PP2

---

*Generator of 2D PointPattern objects with 1,4,5,6,7,8 or 9 points*


---

**Description**

Create a PointPattern-class object representing a 2D point pattern with 1,4,5,6,7,8 or 9 points lying in a rectangle.

**Usage**

```
PP2(n = 4, h = 1, v = h, h3 = TRUE)
```

**Arguments**

n	the number of points. Must be 1,4,5,6,7,8 or 9. Default, 4.
h	the horizontal side length of the rectangle. Default, 1.
v	the vertical side length of the rectangle. Default, h.
h3	if TRUE, alignment of 3 points (see details below) are horizontal (vertical if FALSE). Used only for n=6, 7 or 8.

**Details**

When  $n=4$ , the four points lie at the four corners of the rectangle. When  $n=5$ , the fifth point lies at the center of the rectangle. When  $n=6$  (and  $h3$  is TRUE), four points lie at the rectangle corners and the two other points lie at the centres of the top and bottom edges. When  $n=7$  (and  $h3$  is TRUE), the point pattern consists of 3 horizontal alignments of 2, 3 and 2 points. When  $n=8$  (and  $h3$  is TRUE), the point pattern consists of 3 horizontal alignments of 3, 2 and 3 points. When  $n=9$ , the point pattern consists of 3 (horizontal or vertical) alignments of 3 points.

**Value**

A `PointPattern-class` object.

**See Also**

Generators `PointPattern`, `PP3`.

**Examples**

```
plot(PP2(7))
```

---

PP3

*Generator of 3D horizontal PointPattern objects with 1,4,5,6,7,8 or 9 points*

---

**Description**

Create a `PointPattern-class` object representing a 3D point pattern with 1,4,5,6,7,8 or 9 points lying in a horizontal rectangle.

**Usage**

```
PP3(n=4, xp=1, yp=xp, h3=TRUE)
```

**Arguments**

<code>n</code>	the number of points. Must be 1,4,5,6,7,8 or 9. Default, 4.
<code>xp</code>	the side length of the bounding rectangle, parallel to the x-axis. Default, 1.
<code>yp</code>	the side length of the bounding rectangle, parallel to the y-axis. Default, <code>xp</code> .
<code>h3</code>	if TRUE, alignments of 3 points (see details below) are made in the direction of the x-axis (in the direction of the y-axis if FALSE). Used only for $n=6, 7$ or 8.

## Details

The 3D point pattern follows the construction rules of the 2D point pattern generator [PP2](#): when  $n=4$ , the four points lie at the four corners of the rectangle. When  $n=5$ , the fifth point lies at the center of the rectangle. When  $n=6$  (and  $h3$  is TRUE), four points lie at the rectangle corners and the two other points lie at the centres of the "top" and "bottom" edges (parallel to the x-axis). When  $n=7$  (and  $h3$  is TRUE), the point pattern consists of 3 "horizontal" alignments of 2, 3 and 2 points. When  $n=8$  (and  $h3$  is TRUE), the point pattern consists of 3 "horizontal" alignments of 3, 2 and 3 points. When  $n=9$ , the point pattern consists of 3 ("horizontal" or "vertical") alignments of 3 points.

## Value

A [PointPattern-class](#) object.

## See Also

Generators of point patterns [PointPattern](#), [PP2](#), generators of point pattern lattices [PPBRectLat3](#), [PPFRectLat3](#), [PPRectLat3](#).

## Examples

```
PP3(5,1,2)
```

---

PPBRectLat3	<i>Generator of 3D body-centered rectangular lattices of horizontal point patterns</i>
-------------	--

---

## Description

Create a [FigLat-class](#) object representing a 3D body-centered rectangular lattice of horizontal point patterns lying in a rectangle.

## Usage

```
PPBRectLat3(dx=1, dy=dx, dz=dx, n=1, xp=dx/5, yp=xp, h3=TRUE)
```

## Arguments

dx	the spacing of the vector lattice along the x-axis. Default: 1.
dy	the spacing of the vector lattice along the y-axis. Default: dx.
dz	the spacing of the vector lattice along the z-axis. Default: dx.
n	the number of points in each point pattern. Valid values for n: 1, 4, 5, 6, 7, 8 or 9 (see <a href="#">PP3</a> or <a href="#">PP2</a> documentations). Default: 1 (lattice of points).
xp	the side length of the bounding rectangle, parallel to the x-axis. Default: dx/5.
yp	the side length of the bounding rectangle, parallel to the y-axis. Default: xp (square bounding box for each point pattern).
h3	determines the orientation of the point pattern when $n=6, 7$ or $8$ , see <a href="#">PP3</a> or <a href="#">PP2</a> documentations.

**Value**

A `FigLat-class` object.

**See Also**

Generators `FigLat`, `PointPattern`, `PP3`, `BCRectLat3`, generators of other 3D lattices of point patterns `PPFCRectLat3`, `PPRectLat3`.

**Examples**

```
PPBCRectLat3()# 3D unit body-centered rectangular vector lattice
print(PPBCRectLat3(dx=1,n=7))
```

---

PPFCRectLat3	<i>Generator of 3D face-centered rectangular lattices of horizontal point patterns</i>
--------------	--

---

**Description**

Create a `FigLat-class` object representing a 3D face-centered rectangular lattice of horizontal point patterns lying in a rectangle.

**Usage**

```
PPFCRectLat3(d=1, dx=sqrt(2)*d, dz=d, n=1, xp=dx/5, yp=xp, h3=TRUE)
```

**Arguments**

d	the distance between two neighbour diagonal locations on the horizontal plane. Default: 1.
dx	the distance between two neighbour locations along the x-axis on the horizontal plane. Default: $\sqrt{2}d$ .
dz	the vertical spacing between two neighbour horizontal planes. Default: d.
n	the number of points in each point pattern. Valid values for n: 1, 4, 5, 6, 7, 8 or 9 (see <code>PP3</code> or <code>PP2</code> documentations). Default: 1 (lattice of points).
xp	the side length of the bounding rectangle, parallel to the x-axis. Default: dx/5.
yp	the side length of the bounding rectangle, parallel to the y-axis. Default: xp (square bounding box for each point pattern).
h3	determines the orientation of the point pattern when n=6, 7 or 8, see <code>PP3</code> or <code>PP2</code> documentations.

**Value**

A `FigLat-class` object.

**See Also**

Generators [FigLat](#), [PointPattern](#), [PP3](#), [FCRectLat3](#), generators of other 3D lattices of point patterns [PPBCRectLat3](#), [PPRectLat3](#).

**Examples**

```
PPFCRectLat3()# 3D unit face-centered rectangular vector lattice
print(PPFCRectLat3(dx=1,n=7))
```

---

 PPHexLat2

*Generator of 2D hexagonal lattices of point patterns*


---

**Description**

Create a FigLat-class object representing a 2D hexagonal lattice of point patterns.

**Usage**

```
PPHexLat2(delta=sqrt(2/sqrt(3)), n=1, hp=delta/5, vp=hp, h3 = TRUE)
```

**Arguments**

delta	the distance between homologous points in two neighbour point patterns. Default : $\sqrt{2/\sqrt{3}}$ (unit lattice).
n	the number of points in each point pattern. Valid values for n: 1, 4, 5, 6, 7, 8 or 9 (see <a href="#">PP2</a> documentation). Default: 1.
hp	the horizontal side length of the rectangle bounding the point pattern. Default: delta/5.
vp	the vertical side length of the rectangle bounding the point pattern. Default: hp (square bounding box for each point pattern).
h3	determines the orientation of the point pattern when n=6, 7 or 8, see <a href="#">PP2</a> documentation.

**Value**

A FigLat-class object.

**See Also**

Generators [FigLat](#), [PointPattern](#), [PP2](#), [HexLat2](#), other generators of 2D figure lattices [PPQcxLat2](#), [PPRectLat2](#), [QHexLat2](#), [QQcxLat2](#), [QRectLat2](#), [SHexLat2](#), [SQcxLat2](#), [SRectLat2](#), [LLat2](#).

**Examples**

```
PPHexLat2() # Unit hexagonal lattice of points
PPHexLat2(n=7)
PPHexLat2(n=7, hp=.4, h3=FALSE)
```

---

 PPQcxLat2

*Generator of 2D quincunx lattices of point patterns*


---

### Description

Create a FigLat-class object representing a 2D quincunx lattice of point patterns.

### Usage

```
PPQcxLat2(d=1, dx=sqrt(2)*d, n=1, hp=dx/5, vp=hp, h3=TRUE)
```

### Arguments

d	the distance between two neighbour diagonal locations. Default: 1.
dx	the distance between two neighbour horizontal locations. Default value: quincunx inside a square.
n	the number of points in each point pattern. Valid values for n: 1, 4, 5, 6, 7, 8 or 9 (see <a href="#">PP2</a> documentation). Default: 1 (lattice of points).
hp	the horizontal side length of the rectangle bounding the point pattern. Default: dx/5.
vp	the vertical side length of the rectangle bounding the point pattern. Default: hp (square bounding box for each point pattern).
h3	determines the orientation of the point pattern when n=6, 7 or 8, see <a href="#">PP2</a> documentation.

### Value

A FigLat-class object.

### See Also

Generators [FigLat](#), [PointPattern](#), [PP2](#), [QcxLat2](#), other generators of 2D figure lattices [PPHexLat2](#), [PPRectLat2](#), [QHexLat2](#), [QQcxLat2](#), [QRectLat2](#), [SHexLat2](#), [SQcxLat2](#), [SRectLat2](#), [LLat2](#).

### Examples

```
plot(PPQcxLat2(n=6, h3=FALSE), xlim=c(0, 3), ylim=c(0, 3))
```

---

 PPRectLat2

*Generator of 2D rectangular lattices of point patterns*


---

### Description

Create a FigLat-class object representing a 2D rectangular lattice of point patterns.

### Usage

```
PPRectLat2(h1=1, v1=h1, n=1, hp=h1/5, vp=hp, h3 = TRUE)
```

### Arguments

h1	the horizontal distance between homologous points in two neighbour point patterns. Default : 1.
v1	the vertical distance between homologous points in two neighbour point patterns. Default: h1 (square lattice).
n	the number of points in each point pattern. Valid values for n: 1, 4, 5, 6, 7, 8 or 9 (see <a href="#">PP2</a> documentation). Default: 1 (lattice of points).
hp	the horizontal side length of the rectangle bounding the point pattern. Default: h1/5.
vp	the vertical side length of the rectangle bounding the point pattern. Default: hp (square bounding box for each point pattern).
h3	determines the orientation of the point pattern when n=6, 7 or 8, see <a href="#">PP2</a> documentation.

### Value

A FigLat-class object.

### See Also

Generators [FigLat](#), [PointPattern](#), [PP2](#), [RectLat2](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [QHexLat2](#), [QQcxLat2](#), [QRectLat2](#), [SHexLat2](#), [SQcxLat2](#), [SRectLat2](#), [LLat2](#).

### Examples

```
PPRectLat2()# Unit square lattice of points
PPRectLat2(h1=1,n=7)
PPRectLat2(1,1,7, .2, .2,FALSE)
```

---

PPRectLat3

*Generator of 3D rectangular lattices of horizontal point patterns*

---

## Description

Create a FigLat-class object representing a 3D rectangular lattice of horizontal point patterns lying in a rectangle.

## Usage

```
PPRectLat3(dx=1, dy=dx, dz=dx, n=1, xp=dx/5, yp=xp, h3=TRUE)
```

## Arguments

dx	the spacing of the vector lattice along the x-axis. Default: 1.
dy	the spacing of the vector lattice along the y-axis. Default: dx.
dz	the spacing of the vector lattice along the z-axis. Default: dx.
n	the number of points in each point pattern. Valid values for n: 1, 4, 5, 6, 7, 8 or 9 (see <a href="#">PP3</a> or <a href="#">PP2</a> documentations). Default: 1 (lattice of points).
xp	the side length of the bounding rectangle, parallel to the x-axis. Default: dx/5.
yp	the side length of the bounding rectangle, parallel to the y-axis. Default: xp (square bounding box for each point pattern).
h3	determines the orientation of the point pattern when n=6, 7 or 8, see <a href="#">PP3</a> or <a href="#">PP2</a> documentations.

## Value

A FigLat-class object.

## See Also

Generators [FigLat](#), [PointPattern](#), [PP3](#), [RectLat3](#), generators of other 3D lattices of point patterns [PPBRectLat3](#), [PPFRectLat3](#).

## Examples

```
PPRectLat3()# Unit cubic lattice of points  
PPRectLat3(dx=1, n=7)
```

---

 QcxLat2

*Generator of 2D quincunx lattices as VecLat objects*


---

**Description**

Create an object of class "VecLat" representing a planar quincunx vector lattice.

**Usage**

```
QcxLat2(d=1,dx=sqrt(2)*d)
```

**Arguments**

d	the distance between two neighbour diagonal locations. Default: 1.
dx	the distance between two neighbour horizontal locations. Default value: quincunx inside a square.

**Value**

A [VecLat-class](#) object.

**See Also**

Generators [VecLat](#), [RectLat2](#), [HexLat2](#).

**Examples**

```
QcxLat2()  
QcxLat2(d=2,dx=sqrt(2))
```

---

 QHexLat2

*Generator of 2D hexagonal lattices of quadrats*


---

**Description**

Create a FigLat-class object representing an hexagonal lattice of quadrats in the plane.

**Usage**

```
QHexLat2(delta=sqrt(2/sqrt(3)),hq=delta/5,vq=hq)
```

**Arguments**

delta	the distance between homologous points in two neighbour quadrats. Default: $\sqrt{2/\sqrt{3}}$ (unit lattice).
hq	the horizontal side length of the quadrat. Default: delta/5.
vq	the vertical side length of the quadrat. Default: hq.

**Value**

A [FigLat-class](#) object.

**See Also**

Generators [FigLat](#), [HexLat2](#), [Quadrat](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [PPRectLat2](#), [QQcxLat2](#), [QRectLat2](#), [SHexLat2](#), [SQcxLat2](#), [SRectLat2](#), [LLat2](#).

**Examples**

```
QHexLat2(sqrt(3)/8, hq=0.1)
```

---

QQcxLat2

*Generator of 2D quincunx lattices of quadrats*

---

**Description**

Create a `FigLat-class` object representing a planar quincunx lattice of quadrats.

**Usage**

```
QQcxLat2(d=1, dx=sqrt(2)*d, hq=dx/5, vq=hq)
```

**Arguments**

<code>d</code>	the distance between two neighbour diagonal locations. Default: 1.
<code>dx</code>	the distance between two neighbour horizontal locations. Default value: quincunx inside a square.
<code>hq</code>	the horizontal side length of the quadrat. Default: $dx/5$ .
<code>vq</code>	the vertical side length of the quadrat. Default: <code>hq</code> .

**Value**

A [FigLat-class](#) object.

**See Also**

Generators [FigLat](#), [QcxLat2](#), [Quadrat](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [PPRectLat2](#), [QHexLat2](#), [QRectLat2](#), [SHexLat2](#), [SQcxLat2](#), [SRectLat2](#), [LLat2](#).

**Examples**

```
print(QQcxLat2(d=1, hq=0.1))
plot(QQcxLat2(d=1, hq=0.5), xlim=c(0, 3), ylim=c(0, 3))
```

---

 QRectLat2

*Generator of 2D rectangular lattices of quadrats*


---

**Description**

Creates a FigLat-class object representing a rectangular lattice of quadrats in the plane.

**Usage**

```
QRectLat2(hl=1, vl=hl, hq=hl/5, vq=hq)
```

**Arguments**

hl	the horizontal distance between homologous points in two neighbour quadrats. Default : 1 (unit lattice).
vl	the vertical distance between homologous points in two neighbour quadrats. Default: hl (square lattice).
hq	the horizontal side length of the quadrat. Default: hl/5.
vq	the vertical side length of the quadrat. Default: hq.

**Value**

A FigLat-class object.

**See Also**

Generators [FigLat](#), [RectLat2](#), [Quadrat](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [PPRectLat2](#), [QHexLat2](#), [QQcxLat2](#), [SHexLat2](#), [SQcxLat2](#), [SRectLat2](#), [LLat2](#).

**Examples**

```
# Square unit lattice of quadrats
QRectLat2(hl=1, hq=0.5)
```

---

 Quadrat

*Generator of Quadrat objects*


---

**Description**

Create a Quadrat object representing a planar quadrat (rectangle).

**Usage**

```
Quadrat(hsize, vsize=hsize)
```

**Arguments**

hsize            the horizontal side length.  
 vsize            the vertical side length. Default: hsize.

**Value**

An object of [Quadrat-class](#).

**See Also**

[Quadrat-class](#), generators [Segment](#), [PointPattern](#).

**Examples**

Quadrat(2,1)

Quadrat-class            *Class "Quadrat"*

**Description**

Extend class "Figure". Representation of planar quadrats (rectangles).

**Objects from the Class**

Objects can be created by calls of the form `new("Quadrat", ...)`. It is recommended to use the generator [Quadrat](#).

**Slots**

**dimspace:** Object of class "numeric". Dimension of the space where the quadrat lies. Equal to 2.  
**coord:** Object of class "matrix". A 2x1 matrix containing the Cartesian coordinates of a vertex of the quadrat. The opposite vertex is defined as the origin.

**Extends**

Class "Figure", directly.

**Methods**

**content** signature(x = "Quadrat"): compute the quadrat area.  
**covariogram** signature(x = "Quadrat", f = "function"): [covariogram](#) for a Quadrat object.  
**plot** signature(x = "Quadrat", y = "missing"): plot a Quadrat object. The extra argument `origin` (default: null vector) may be used to plot the quadrat translated by the vector `origin`. If the extra argument `add` (default: FALSE) is TRUE, the quadrat is added to the current plot.  
**print** signature(x = "Quadrat"): print a Quadrat object.

**See Also**

Generator [Quadrat](#), classes [Segment-class](#), [PointPattern-class](#).

---

 RectLat2

*Generator of rectangular lattices as VecLat objects*


---

**Description**

Create an object of class "VecLat" representing a planar rectangular vector lattice.

**Usage**

```
RectLat2(h = 1, v = h)
```

**Arguments**

h                    the horizontal side length of a fundamental tile. Default: 1.  
 v                    the vertical side length of a fundamental tile. Default: h (square lattice).

**Value**

A [VecLat-class](#) object.

**See Also**

Generators [VecLat](#), [HexLat2](#), [QcxLat2](#).

**Examples**

```
## Planar square unit vector lattice
RectLat2()
## Planar rectangular vector lattice
RectLat2(2,1)
```

---

 RectLat3

*Generator of 3D rectangular lattices as VecLat objects*


---

**Description**

Create an object of class "VecLat" representing a 3D rectangular vector lattice.

**Usage**

```
RectLat3(dx=1, dy=dx, dz=dx)
```

**Arguments**

dx	spacing along the x-axis. Default: 1.
dy	spacing along the y-axis. Default: dx.
dz	spacing along the z-axis. Default: dy.

**Value**

A [VecLat-class](#) object.

**See Also**

Generators [VecLat](#), [BCRectLat3](#), [FCRectLat3](#).

**Examples**

```
RectLat3()  
RectLat3(1,1,3)
```

---

scaling

*Scale Figure, VecLat or FigLat objects*

---

**Description**

Scale (multiply by a real value) a subset.

**Usage**

```
scaling(x, s)
```

**Arguments**

x	a <a href="#">Figure-class</a> , <a href="#">VecLat-class</a> or <a href="#">FigLat-class</a> object.
s	the scaling factor.

**Value**

An object of the same class as x.

**See Also**

[Figure-class](#), [VecLat-class](#), [FigLat-class](#).

---

Segment	<i>Generator of Segment objects</i>
---------	-------------------------------------

---

**Description**

Generate a Segment object representing a line segment starting at the origin.

**Usage**

```
Segment(end)
```

**Arguments**

end	a vector containing the Cartesian coordinates of the end point which does not lie at the origin.
-----	--

**Value**

An object of [Segment-class](#).

**See Also**

[Segment-class](#), generators [Quadrat](#), [PointPattern](#).

**Examples**

```
Segment(c(1,0))
```

---

Segment-class	<i>Class "Segment"</i>
---------------	------------------------

---

**Description**

Extend class "Figure". Representation of line segments starting from the origin.

**Objects from the Class**

Objects can be created by calls of the form `new("Segment", ...)`. It is recommended to use the generator [Segment](#).

**Slots**

**dimspace:** Object of class "numeric". Dimension of the space where the segment lies.

**coord:** Object of class "matrix". A 2x1 matrix containing the Cartesian coordinates of the end point which does not lie at the origin.

**Extends**

Class "Figure", directly.

**Methods**

**content** signature(x = "Segment"): compute the segment length.

**covariogram** signature(x = "Segment", f = "function"): [covariogram](#) for a Segment object.

**plot** signature(x = "Segment", y = "missing"): plot a Segment object. Implemented only for planar segments. The extra argument origin (default: null vector) may be used to plot the segment translated by the vector origin. If the extra argument add (default: FALSE) is TRUE, the segment is added to the current plot.

signature(x = "Segment", y = "matrix"): Plot the strip defined as the Minkowski sum of x and the linear subspace spanned by the column vectors of y. Only implemented in 2D. In addition to the two arguments origin and add described above, the arguments density, angle, col, border and lty (see [polygon](#)) can be provided in order to customize the graphical representation of the strip.

**print** signature(x = "Segment"): print a segment object.

**See Also**

Generator [Segment](#), [Quadrat-class](#), [PointPattern-class](#).

---

 SHexLat2

*Generator of 2D hexagonal lattices of segments*


---

**Description**

Create a FigLat-class object representing an hexagonal lattice of segments in the plane.

**Usage**

```
SHexLat2(delta=sqrt(2/sqrt(3)),end=c(delta/5,0))
```

**Arguments**

delta            the distance between homologous points in two neighbour segments. Default:  $\sqrt{2/\sqrt{3}}$  (unit lattice).

end              a vector containing the Cartesian coordinates of the segment end point (the other end point is supposed to lie at the origin). Default: c(delta/5, 0).

**Value**

A FigLat-class object.

**See Also**

Generators [FigLat](#), [HexLat2](#), [Segment](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [PPRectLat2](#), [QHexLat2](#), [QQcxLat2](#), [QRectLat2](#), [SQcxLat2](#), [SRectLat2](#), [LLat2](#).

**Examples**

```
SHexLat2(end=c(0.5,0.5))
```

---

SQcxLat2

*Generator of 2D quincunx lattices of line segments*

---

**Description**

Create a FigLat-class object representing a planar quincunx lattice of line segments.

**Usage**

```
SQcxLat2(d=1, dx=sqrt(2)*d, end=c(dx/5, 0))
```

**Arguments**

d	the distance between two neighbour diagonal locations. Default: 1.
dx	the distance between two neighbour horizontal locations. Default value: quincunx inside a square.
end	a vector containing the Cartesian coordinates of the segment end point (the other end point is supposed to lie at the origin). Default: c(dx/5, 0).

**Value**

A FigLat-class object.

**See Also**

Generators [FigLat](#), [HexLat2](#), [Segment](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [PPRectLat2](#), [QHexLat2](#), [QQcxLat2](#), [QRectLat2](#), [SHexLat2](#), [SRectLat2](#), [LLat2](#).

**Examples**

```
SQcxLat2(1, sqrt(2), end=c(1/5, 0))
plot(SQcxLat2(1, sqrt(2), end=c(1/5, 0)), xlim=c(0, 5), ylim=c(0, 5))
```

---

SRectLat2                      *Generator of 2D rectangular lattices of segments*

---

### Description

Create a FigLat-class object representing a rectangular lattice of segments in the plane.

### Usage

```
SRectLat2(h1=1, v1=h1, end=c(h1/5, 0))
```

### Arguments

h1	the horizontal distance between homologous points in two neighbour segments. Default : 1.
v1	the vertical distance between homologous points in two neighbour segments. Default: h1 (square lattice).
end	a vector containing the Cartesian coordinates of the segment end point (the other end point is supposed to lie at the origin). Default: c(h1/5, 0).

### Value

A FigLat-class object.

### See Also

Generators [FigLat](#), [HexLat2](#), [Segment](#), other generators of 2D figure lattices [PPHexLat2](#), [PPQcxLat2](#), [PPRectLat2](#), [QHexLat2](#), [QQcxLat2](#), [QRectLat2](#), [SHexLat2](#), [SQcxLat2](#), [LLat2](#).

### Examples

```
SRectLat2(h1=1, end=c(0.5, 0.5))
```

---

VecLat                              *Generator of VecLat objects*

---

### Description

Create an object of class "VecLat" representing a vector lattice.

### Usage

```
VecLat(gmat)
```

**Arguments**

**gmat** a vector or a matrix. If **gmat** is a vector, it is transformed into a one-column matrix.

**Value**

A [VecLat-class](#) object.

**See Also**

[VecLat-class](#), generators for 2D lattices [HexLat2](#), [QcxLat2](#), [RectLat2](#), generators for 3D lattices [BCRectLat3](#), [FRectLat3](#), [RectLat3](#).

**Examples**

```
## Planar square lattice
VecLat(diag(2))
## Planar discrete horizontal line
VecLat(c(1,0))
```

---

VecLat-class

*Class "VecLat"*

---

**Description**

Objects of class "VecLat" represent vector lattices.

**Objects from the Class**

Objects can be created by calls of the form `new("VecLat", ...)` or using generators such as [VecLat](#), [RectLat2](#), [HexLat2](#).

**Slots**

**dimspace**: Object of class "numeric". Dimension of the space where the lattice lies.

**dimsupp**: Object of class "numeric". Dimension of the support of the lattice, defined as the smallest vector subspace containing the lattice.

**gmat**: Object of class "matrix". Generating matrix of the lattice.

**gmat0**: Object of class "matrix". Unit version of the generating matrix.

**det**: Object of class "numeric". Determinant of the lattice.

**Methods**

**ltransf** signature(`x = "VecLat"`, `m = "matrix"`): [ltransf](#) for a VecLat object.

**print** signature(`x = "VecLat"`): print a VecLat object.

**scaling** signature(`x = "VecLat"`, `s = "numeric"`): [scaling](#) for a VecLat object.

**References**

Conway, J. and Sloane, N. (editors) (1999). *Sphere Packings, Lattices and Groups*. Springer-Verlag.

**See Also**

Generators [VecLat](#), [RectLat2](#), [HexLat2](#).

---

 vol.mse

*MSE approximation for volume predictors*


---

**Description**

Compute a MSE approximation for volume predictors. The structure of interest is an isotropic 3D random compact set. The sampling device is a uniform random lattice of figures (point patterns, line segments, sections...). The approximation depends only on sampling parameters and on the mean surface (to be provided) of the structure.

**Usage**

```
vol.mse(x, S = 1, L = 3)
```

**Arguments**

x	a lattice of figures, object of class <a href="#">FigLat-class</a> .
S	the mean surface. Default: 1.
L	an integer, the criterion for stopping summation of the Epstein zeta function. Argument of the function <a href="#">Ezeta</a> . Default: 3.

**Value**

The MSE approximation as a numeric.

**References**

Kieu, K. and Mora, M. (2005). Stereological estimation of mean volume: precision of three sampling designs. Technical report 2005-1, Unite de Mathematiques et informatique appliquees, INRA. [http://www.inra.fr/bia/J/nosdoc/rapport\\_miaj\\_2005\\_1.pdf](http://www.inra.fr/bia/J/nosdoc/rapport_miaj_2005_1.pdf).

**See Also**

[area.mse](#), [dvol.mse](#).

**Examples**

```
# Sampling by a unit cubic point lattice
vol.mse(FigLat(3,VecLat(diag(3)),PointPattern(rep(0,3))))
# Sampling by serial sections
vol.mse(FigLat(3,VecLat(c(0,0,1)),PointPattern(rep(0,3)),lmat=rbind(diag(2),rep(0,2))))
```

---

 vol.mse.est

*MSE estimation for 3D-volume predictors*


---

### Description

Compute a MSE estimate for 3D-volume predictors. The structure of interest is a random compact set. The sampling device is a uniform random lattice of figures.

### Usage

```
vol.mse.est(fldata,mse.only=TRUE,iso=FALSE,diff2use)
```

### Arguments

fldata	data collected for volume prediction, object of class <a href="#">FigLatData-class</a> .
mse.only	TRUE (default) if only the MSE estimate must be returned.
iso	FALSE (default) if the boundary is not assumed to be isotropic.
diff2use	a family of lattice vectors defining the data covariations to be used for the MSE estimation. A matrix with as many columns as lattice vectors. Each column contains the vector coordinates in the basis defined by the lattice generating matrix. Optional: a default value is computed. If isotropy is assumed, the lattice generating matrix (3 main basis vectors). Otherwise, 3 face diagonal lattice vectors are added to the 3 main basis vectors.

### Details

vol.mse.est invokes the function [vol.mse](#).

### Value

If mse.only is TRUE, the MSE estimate as a scalar. Otherwise a list with components:

S.est	estimate of the boundary surface area.
deformation	used when the boundary is not assumed to be isotropic. It is assumed that there exists a volume preserving linear transformation which makes the boundary isotropic. This list component contains the estimated transformation matrix.
mse.est	MSE estimate.

# Index

## \*Topic **classes**

- FigLat-class, 11
- Figure-class, 13
- PointPattern-class, 19
- Quadrat-class, 31
- Segment-class, 34
- VecLat-class, 38

## \*Topic **htest**

- area.mse, 2
- dvol.mse, 6
- vol.mse, 39

## \*Topic **nonparametric**

- area.mse, 2
- dvol.mse, 6
- vol.mse, 39

## \*Topic **spatial**

- area.mse, 2
- area.mse.est, 3
- BCRectLat3, 4
- content, 5
- covariogram, 6
- dvol.mse, 6
- dvol.mse.est, 7
- Ezeta, 8
- FCRectLat3, 9
- FigLat, 10
- FigLat-class, 11
- FigLatData, 12
- FigLatData-class, 12
- Figure-class, 13
- HexLat2, 14
- latscale, 14
- LLat2, 15
- ltransf, 16
- M, 17
- pgs-internal, 17
- PointPattern, 19
- PointPattern-class, 19
- PP2, 20

- PP3, 21
- PPBCRectLat3, 22
- PPFCRectLat3, 23
- PPHexLat2, 24
- PPQcxLat2, 25
- PPRectLat2, 26
- PPRectLat3, 27
- QcxLat2, 28
- QHexLat2, 28
- QQcxLat2, 29
- QRectLat2, 30
- Quadrat, 30
- Quadrat-class, 31
- RectLat2, 32
- RectLat3, 32
- scaling, 33
- Segment, 34
- Segment-class, 34
- SHexLat2, 35
- SQcxLat2, 36
- SRectLat2, 37
- VecLat, 37
- VecLat-class, 38
- vol.mse.est, 40

- area.mse, 2, 4, 7, 39
- area.mse.est, 3, 8

- BCRectLat3, 4, 10, 23, 33, 38

- content, 5
- content, PointPattern-method  
(PointPattern-class), 19
- content, Quadrat-method (Quadrat-class),  
31
- content, Segment-method (Segment-class),  
34
- covariogram, 6, 20, 31, 35
- covariogram, PointPattern, function-method  
(PointPattern-class), 19

- covariogram, *Quadrat*, function-method (*Quadrat*-class), 31
- covariogram, *Segment*, function-method (*Segment*-class), 34
- cubicgrid (pgs-internal), 17
- dataCovariogram (*FigLatData*-class), 12
- dataCovariogram, *FigLatData*, matrix-method (*FigLatData*-class), 12
- dualmat (pgs-internal), 17
- dvol.mse, 3, 6, 39
- dvol.mse.est, 4, 7
- ellipsoidSurface (pgs-internal), 17
- Ezeta, 3, 7, 8, 17, 39
- FCRectLat3, 5, 9, 24, 33, 38
- FigLat, 10, 11, 15, 16, 23–27, 29, 30, 36, 37
- FigLat-class, 3, 7, 10, 12, 23, 25–27, 30, 33
- FigLat-class, 11, 16, 24, 29, 35–37, 39
- FigLatData, 12, 13
- FigLatData-class, 4, 7, 12, 40
- FigLatData-class, 12
- Figure-class, 5, 33
- Figure-class, 6, 10, 13, 16, 17
- gaic (pgs-internal), 17
- HexLat2, 14, 24, 28, 29, 32, 36–39
- in.upper.halfspace (pgs-internal), 17
- latscale, 14
- LLat2, 15, 24–26, 29, 30, 36, 37
- ltransf, 11, 13, 16, 38
- ltransf, *FigLat*, matrix-method (*FigLat*-class), 11
- ltransf, *Figure*, matrix-method (*Figure*-class), 13
- ltransf, *Quadrat*, matrix-method (*Quadrat*-class), 31
- ltransf, *VecLat*, matrix-method (*VecLat*-class), 38
- M, 17
- normphase (pgs-internal), 17
- pgs-internal, 17
- plot, *FigLat*, missing-method (*FigLat*-class), 11
- plot, *PointPattern*, matrix-method (*PointPattern*-class), 19
- plot, *PointPattern*, missing-method (*PointPattern*-class), 19
- plot, *Quadrat*, missing-method (*Quadrat*-class), 31
- plot, *Segment*, matrix-method (*Segment*-class), 34
- plot, *Segment*, missing-method (*Segment*-class), 34
- pointcov (pgs-internal), 17
- PointPattern*, 19, 19–27, 31, 34
- PointPattern*-class, 6, 13, 19, 21, 22, 32, 35
- PointPattern*-class, 6, 19
- polygon, 35
- PP2, 19, 20, 22–27
- PP3, 19, 21, 21–24, 27
- PPBCRectLat3, 22, 22, 24, 27
- PPFCRectLat3, 22, 23, 23, 27
- PPHexLat2, 16, 24, 25, 26, 29, 30, 36, 37
- PPQcxLat2, 16, 24, 25, 26, 29, 30, 36, 37
- PPRectLat2, 16, 24, 25, 26, 29, 30, 36, 37
- PPRectLat3, 22–24, 27
- print, *FigLat*-method (*FigLat*-class), 11
- print, *FigLatData*-method (*FigLatData*-class), 12
- print, *PointPattern*-method (*PointPattern*-class), 19
- print, *Quadrat*-method (*Quadrat*-class), 31
- print, *Segment*-method (*Segment*-class), 34
- print, *VecLat*-method (*VecLat*-class), 38
- QcxLat2, 14, 25, 28, 29, 32, 38
- QHexLat2, 16, 24–26, 28, 29, 30, 36, 37
- QQcxLat2, 16, 24–26, 29, 29, 30, 36, 37
- QRectLat2, 16, 24–26, 29, 30, 36, 37
- Quadrat*, 19, 29, 30, 30–32, 34
- Quadrat*-class, 6, 13, 20, 31, 35
- Quadrat*-class, 6, 31
- RectLat2*, 14, 26, 28, 30, 32, 38, 39
- RectLat3*, 5, 10, 27, 32, 38
- scaling, 11, 13, 33, 38
- scaling, *FigLat*, numeric-method (*FigLat*-class), 11
- scaling, *Figure*, numeric-method (*Figure*-class), 13

scaling,Quadrat,numeric-method  
(Quadrat-class), 31

scaling,VecLat,numeric-method  
(VecLat-class), 38

Segment, 19, 31, 34, 34–37

Segment-class, 6, 13, 20, 32, 34

Segment-class, 6, 34

SHexLat2, 16, 24–26, 29, 30, 35, 36, 37

sphereSurface (pgs-internal), 17

SQcxLat2, 16, 24–26, 29, 30, 36, 36, 37

SRectLat2, 16, 24–26, 29, 30, 36, 37

uniroot, 15

vec.norm (pgs-internal), 17

VecLat, 5, 10, 14, 28, 32, 33, 37, 38, 39

VecLat-class, 32, 33

VecLat-class, 5, 8–10, 14, 16, 17, 28, 33, 38,  
38

vol.mse, 3, 7, 39, 40

vol.mse.est, 4, 8, 40