

# Package ‘ads’

April 17, 2009

**Type** Package

**Title** Spatial point patterns analysis

**Version** 1.2-9

**Date** 2009-01-19

**Author** R. Pelissier and F. Goreaud

**Maintainer** Raphael Pelissier <Raphael.Pelissier@ird.fr>

**Suggests** spatstat

**Description** Perform first- and second-order multi-scale analyses derived from Ripley’s K-function, for univariate, multivariate and marked mapped data in rectangular, circular or irregular shaped sampling windows, with test of statistical significance based on Monte Carlo simulations.

**License** GPL-2

**Repository** CRAN

**Date/Publication** 2009-01-20 08:54:17

## R topics documented:

Allogny . . . . .	2
area.swin . . . . .	3
BPoirier . . . . .	4
Couepia . . . . .	5
dval . . . . .	6
inside.swin . . . . .	8
k12fun . . . . .	9
k12val . . . . .	12
kfun . . . . .	14
ki.fun . . . . .	17
kjfun . . . . .	18
kmfun . . . . .	20
kval . . . . .	22

plot.fads . . . . .	24
plot.spp . . . . .	26
plot.vads . . . . .	28
spp . . . . .	30
swin . . . . .	33
triangulate . . . . .	35

<b>Index</b>	<b>37</b>
--------------	-----------

---

Allogny	<i>Spatial pattern of oaks suffering from frost shake in Allogny, France.</i>
---------	---

---

### Description

Spatial pattern of sound and splited oaks (*Quercus petraea*) suffering from frost shake in a 2.35-ha plot in Allogny, France.

### Usage

```
data(Allogny)
```

### Format

A list with 4 components:

`$rect` is a vector of coordinates ( $xmin, ymin, xmax, ymax$ ) of the origin and the opposite corner of a 125 by 188 m square plot.

`$trees` is a list of tree coordinates ( $x, y$ ).

`$status` is a factor with 2 levels ("splited", "sound").

### Source

Grandjean, G., Jabiol, B., Bruchiamacchie, M. & Roustan, F. 1990. *Recherche de corrélations entre les paramètres édaphiques, et plus spécialement texture, hydromorphie et drainage interne, et la réponse individuelle des chenes sessiles et pédonculés à la gélivure.* Rapport de recherche ENITEF, Nogent sur Vernisson, France.

### References

Goreaud, F. & Pélissier, R. 2003. Avoiding misinterpretation of biotic interactions with the intertype *K12*-function: population independence vs. random labelling hypotheses. *Journal of Vegetation Science*, 14: 681-692.

### Examples

```
data(Allogny)
allo.spp<-spp(Allogny$trees,mark=Allogny$status,window=Allogny$rect)
plot(allo.spp)
```

---

area.swin	<i>Area of a sampling window</i>
-----------	----------------------------------

---

## Description

Function `area.swin` computes the area of a sampling window.

## Usage

```
area.swin(w)
```

## Arguments

`w` an object of class "swin" defining the sampling window.

## Details

For "simple" sampling windows, returns simply the area of the rectangle or circle delineating the study region.

For "complex" sampling windows, returns the area of the initial rectangle or circle, minus the total area of the triangles to remove (see [swin](#)).

## Value

The area of the sampling window.

## Author(s)

[Raphael.Pelissier@ird.fr](mailto:Raphael.Pelissier@ird.fr)

## See Also

[swin](#).

## Examples

```
#rectangle of size [0,110] x [0,90]
wr<-swin(c(0,0,110,90))
area.swin(wr)

#circle with radius 50 centred on (55,45)
wc<-swin(c(55,45,50))
area.swin(wc)

# polygon (diamond shape)
t1<-c(0,0,55,0,0,45)
t2<-c(55,0,110,0,110,45)
t3<-c(0,45,0,90,55,90)
t4<-c(55,90,110,90,110,45)
```

```
wp<-swin(wr, rbind(t1,t2,t3,t4))
area.swin(wp)
```

---

BPoirier

*Tree spatial pattern in Beau Poirier plot, Haye forest, France*


---

## Description

Spatial pattern of 162 beeches, 72 oaks and 3 hornbeams in a 1-ha 140 yr-old temperate forest plot in Haye, France.

## Usage

```
data(BPoirier)
```

## Format

A list with 8 components:

`$rect` is a vector of coordinates  $(x_{min}, y_{min}, x_{max}, y_{max})$  of the origin and the opposite corner of a 110 by 90 m rectangular plot.

`$tri1` is a list of vertice coordinates  $(ax, ay, bx, by, cx, cy)$  of contiguous triangles covering the denser part of the plot.

`$tri2` is a list of vertice coordinates  $(ax, ay, bx, by, cx, cy)$  of contiguous triangles covering the sparser part of the plot.

`$poly1` is a list of vertice coordinates  $(x, y)$  of the polygon enclosing `BPoirier$tri1`.

`$poly2` is a list of two polygons vertice coordinates  $(x, y)$  enclosing `BPoirier$tri2`.

`$trees` is a list of tree coordinates  $(x, y)$ .

`$species` is a factor with 3 levels ("beech", "oak", "hornbeam") corresponding to species names of the trees.

`$dbh` is a vector of tree size (diameter at breast height in cm).

## Source

Pardé, J. 1981. De 1882 à 1976/80 : les places d'expérience de sylviculture du hêtre en forêt domaniale de Haye. *Revue Forestière Française*, 33: 41-64.

## References

Goreaud, F. 2000. *Apports de l'analyse de la structure spatiale en forêt tempérée à l'étude et la modélisation des peuplements complexes*. Thèse de doctorat, ENGREF, Nancy, France.

Pélissier, R. & Goreaud, F. 2001. A practical approach to the study of spatial structure in simple cases of heterogeneous vegetation. *Journal of Vegetation Science*, 12: 99-108.

## Examples

```
data(BPoirier)
BP.spp<-spp(BPoirier$trees, mark=BPoirier$species, window=BPoirier$rect)
plot(BP.spp)
```

---

Couepia	<i>Spatial pattern of Couepia caryophylloides in Paracou, a canopy tree species of French Guiana.</i>
---------	---

---

### Description

Spatial pattern of 34 mature individuals and 173 young individuals of the tree species *Couepia caryophylloides* (Chrysobalanaceae) in a 25-ha forest plot in Paracou, French Guiana.

### Usage

```
data(Couepia)
```

### Format

A list with 4 components:

`$rect` is a vector of coordinates ( $xmin, ymin, xmax, ymax$ ) of the origin and the opposite corner of a 500 by 500 m rectangular plot.

`$tri` is a list of vertice coordinates ( $ax, ay, bx, by, cx, cy$ ) of contiguous triangles covering swampy parts of the plot.

`$trees` is a list of tree coordinates ( $x, y$ ).

`$stage` is a factor with 2 levels ("mature", "young").

### Source

Collinet, F. 1997. *Essai de regroupement des principales espèces structurantes d'une forêt dense humide d'après l'analyse de leur répartition spatiale (forêt de Paracou - Guyane)*. Thèse de doctorat, Université Claude Bernard, Lyon, France.

### References

Goreaud, F. & Pélissier, R. 2003. Avoiding misinterpretation of biotic interactions with the intertype *K12*-function: population independence vs. random labelling hypotheses. *Journal of Vegetation Science*, 14: 681-692.

### Examples

```
data(Couepia)
coca.spp<-spp(Couepia$trees,mark=Couepia$stage,window=Couepia$rect,triangles=Couepia$tri)
plot(coca.spp)
```

dval

*Multiscale local density of a spatial point pattern***Description**

Computes local density estimates of a spatial point pattern, i.e. the number of points per unit area, within sample circles of regularly increasing radii  $r$ , centred at the nodes of a grid covering a simple (rectangular or circular) or complex sampling window (see Details).

**Usage**

```
dval(p, upto, by, nx, ny)
```

**Arguments**

<code>p</code>	a "spp" object defining a spatial point pattern in a given sampling window (see <a href="#">spp</a> ).
<code>upto</code>	maximum radius of the sample circles (see Details).
<code>by</code>	interval length between successive sample circles radii (see Details).
<code>nx, ny</code>	number of sample circles regularly spaced out in $x$ and $y$ directions.

**Details**

The local density is estimated for a regular sequence of sample circles radii given by `seq(by, upto, by)` (see [seq](#)). The sample circles are centred at the nodes of a regular grid with size  $nx$  by  $ny$ . Ripley's edge effect correction is applied when the sample circles overlap boundary of the sampling window (see Ripley (1977) or Goreaud & Pélissier (1999) for an extension to circular and complex sampling windows). Due to edge effect correction, `upto`, the maximum radius of the sample circles, is half the longer side for a rectangle sampling window (i.e.  $0.5 * \max((x_{max} - x_{min}), (y_{max} - y_{min}))$ ) and the radius  $r_0$  for a circular sampling window (see [swin](#)).

**Value**

A list of class `c("vads", "dval")` with essentially the following components:

<code>r</code>	a vector of regularly spaced out distances ( <code>seq(by, upto, by)</code> ).
<code>xy</code>	a data frame of $(nx * ny)$ observations giving $(x, y)$ coordinates of the centers of the sample circles (the grid nodes).
<code>cval</code>	a matrix of size $(nx * ny, length(r))$ giving the estimated number of points of the pattern per sample circle with radius $r$ .
<code>dval</code>	a matrix of size $(nx * ny, length(r))$ giving the estimated number of points of the pattern per unit area per sample circle with radius $r$ .

**Warning**

In its current version, function `dval` ignores the marks of multivariate and marked point patterns (they are all considered to be univariate patterns).

**Note**

There are printing, summary and plotting methods for "vads" objects.

**Author(s)**

<Raphael.Pelissier@ird.fr>

**References**

Goreaud, F. and Pélissier, R. 1999. On explicit formula of edge effect correction for Ripley's  $K$ -function. *Journal of Vegetation Science*, 10:433-438.

Pélissier, R. and Goreaud, F. 2001. A practical approach to the study of spatial structure in simple cases of heterogeneous vegetation. *Journal of Vegetation Science*, 12:99-108.

Ripley, B.D. 1977. Modelling spatial patterns. *Journal of the Royal Statistical Society B*, 39:172-212.

**See Also**

[plot.vads](#), [spp](#).

**Examples**

```
data(BPoirier)
BP<-BPoirier
# spatial point pattern in a rectangle sampling window of size [0,110] x [0,90]
swr<-spp(BP$trees,win=BP$rect)
dswr<-dval(swr,25,1,11,9)
summary(dswr)
plot(dswr)

# spatial point pattern in a circle with radius 50 centred on (55,45)
swc<-spp(BP$trees,win=c(55,45,45))
dswc<-dval(swc,25,1,9,9)
summary(dswc)
plot(dswc)

# spatial point pattern in a complex sampling window
swrt<-spp(BP$trees,win=BP$rect,tri=BP$tril)
dswrt<-dval(swrt,25,1,11,9)
summary(dswrt)
plot(dswrt)
```

---

inside.swin      *Test whether points are inside a sampling window*

---

### Description

Function `inside.swin` tests whether points lie inside or outside a given sampling window.

### Usage

```
inside.swin(x, y, w, bdry=TRUE)
```

### Arguments

<code>x</code>	a vector of <code>x</code> coordinates of points.
<code>y</code>	a vector of <code>y</code> coordinates of points.
<code>w</code>	an object of class "swin" (see <a href="#">swin</a> ) defining the sampling window.
<code>bdry</code>	by default <code>bdry = TRUE</code> . If <code>FALSE</code> , points located on the boundary of the sampling window are considered to be outside.

### Value

A logical vector whose `i`th entry is `TRUE` if the corresponding point  $(x[i], y[i])$  is inside `w`, `FALSE` otherwise.

### Note

For "complex" sampling windows, points inside the triangles to remove or on their boundary, are considered outside.

### Author(s)

⟨Raphael.Pelissier@ird.fr⟩

### See Also

[swin.](#)

### Examples

```
data(BPoirier)
BP<-BPoirier
wr<-swin(BP$rect)
sum(inside.swin(BP$trees$x,BP$trees$y,wr))

wc<-swin(c(55,45,45))
sum(inside.swin(BP$trees$x,BP$trees$y,wc))

wrt<-swin(BP$rect,triangles=BP$tril)
sum(inside.swin(BP$trees$x,BP$trees$y,wrt))
```

---

k12fun	<i>Multiscale second-order neighbourhood analysis of a bivariate spatial point pattern</i>
--------	--

---

## Description

Computes estimates of the intertype  $K12$ -function and associated neighbourhood functions from a bivariate spatial point pattern in a simple (rectangular or circular) or complex sampling window. Computes optionally local confidence limits of the functions under the null hypotheses of population independence or random labelling (see Details).

## Usage

```
k12fun(p, upto, by, nsim = 0, H0=c("pi", "r1"), prec = 0.01, alpha = 0.01, marks)
```

## Arguments

p	a "spp" object defining a multivariate spatial point pattern in a given sampling window (see <a href="#">spp</a> ).
upto	maximum radius of the sample circles (see Details).
by	interval length between successive sample circles radii (see Details).
nsim	number of Monte Carlo simulations to estimate local confidence limits of the selected null hypothesis (see Details). By default <code>nsim=0</code> , so that no confidence limits are computed.
H0	one of <code>c("pi", "r1")</code> to select either the null hypothesis of population independence ( <code>H0="pi"</code> ) or random labelling ( <code>H0="r1"</code> ) (see Details). By default, the null hypothesis is population independence.
prec	if <code>nsim&gt;0</code> and <code>H0="pi"</code> , precision of the random vector generated during simulations. By default <code>prec=0.01</code> .
alpha	if <code>nsim&gt;0</code> , significant level of the confidence limits. By default $\alpha = 0.01$ .
marks	by default <code>c(1,2)</code> , otherwise a vector of two numbers or character strings identifying the types (the <code>p\$marks</code> levels) of points of type 1 and 2, respectively.

## Details

Function `k12fun` computes the intertype  $K12(r)$  function of second-order neighbourhood analysis and the associated functions  $g12(r)$ ,  $n12(r)$  and  $L12(r)$ .

For a homogeneous isotropic bivariate point process of intensities  $\lambda_1$  and  $\lambda_2$ , the second-order property could be characterized by a function  $K12(r)$  (Lotwick & Silverman 1982), so that the expected number of neighbours of type 2 within a distance  $r$  of an arbitrary point of type 1 is:  $N12(r) = \lambda_2 * K12(r)$ .

$K12(r)$  is an intensity standardization of  $N12(r)$ :  $K12(r) = N12(r)/\lambda_2$ .

$n12(r)$  is an area standardization of  $N12(r)$ :  $n12(r) = N12(r)/(\pi * r^2)$ , where  $\pi * r^2$  is the area of the disc of radius  $r$ .

$L12(r)$  is a linearized version of  $K12(r)$ , which has an expectation of 0 under population independence:  $L12(r) = \sqrt{(K12(r)/\pi)} - r$ .  $L12(r)$  becomes positive when the two population show attraction and negative when they show repulsion. Under the null hypothesis of random labelling, the expectation of  $L12(r)$  is  $L(r)$ . It becomes greater than  $L(r)$  when the types tend to be positively correlated and lower when they tend to be negatively correlated.

$g12(r)$  is the derivative of  $K12(r)$  or bivariate pair density function, so that the expected number of points of type 2 at a distance  $r$  of an arbitrary point of type 1 (i.e. within an annuli between two successive circles with radii  $r$  and  $r - by$ ) is:  $O12(r) = \lambda2 * g12(r)$  (Wiegand & Moloney 2004).

The program introduces an edge effect correction term according to the method proposed by Ripley (1977) and extended to circular and complex sampling windows by Goreaud & Pélissier (1999).

Theoretical values under the null hypothesis of either population independence or random labelling as well as local Monte Carlo confidence limits and p-values of departure from the null hypothesis (Besag & Diggle 1977) are estimated at each distance  $r$ .

The population independence hypothesis "pi" assumes that the location of points of a given population is independent from the location of points of the other. It is therefore tested conditionally to the intrinsic spatial pattern of each population, which are just shifted each other by a random vector. The random labelling hypothesis "r1" assumes that the probability to bear a given mark is the same for all points of the pattern and doesn't depends on neighbours. It is therefore tested conditionally to the whole spatial pattern, by randomizing the marks over the points' locations kept unchanged (see Goreaud & Pélissier 2003 for further details).

## Value

A list of class "fads" with essentially the following components:

<code>r</code>	a vector of regularly spaced out distances ( <code>seq(by, upto, by)</code> ).
<code>g12</code>	a data frame containing values of the bivariate pair density function $g12(r)$ .
<code>n12</code>	a data frame containing values of the bivariate local neighbour density function $n12(r)$ .
<code>k12</code>	a data frame containing values of the intertype function $K12(r)$ .
<code>l12</code>	a data frame containing values of the modified intertype function $L12(r)$ .

Each component except `r` is a data frame with the following variables:

obs	a vector of estimated values for the observed point pattern.
theo	a vector of theoretical values expected under the selected null hypothesis.
sup	(optional) if <code>nsim&gt;0</code> a vector of the upper local confidence limits of the selected null hypothesis at a significant level $\alpha$ .
inf	(optional) if <code>nsim&gt;0</code> a vector of the lower local confidence limits of the selected null hypothesis at a significant level $\alpha$ .
pval	(optional) if <code>nsim&gt;0</code> a vector of local p-values of departure from the selected null hypothesis.

### Note

There are printing and plotting methods for "fads" objects.

### Author(s)

<Raphael.Pelissier@ird.fr>

### References

- Besag J.E. & Diggle P.J. 1977. Simple Monte Carlo tests spatial patterns. *Applied Statistics*, 26:327-333.
- Goreaud F. & Pélissier R. 1999. On explicit formulas of edge effect correction for Ripley's K-function. *Journal of Vegetation Science*, 10:433-438.
- Goreaud, F. & Pélissier, R. 2003. Avoiding misinterpretation of biotic interactions with the intertype *K12*-function: population independence vs. random labelling hypotheses. *Journal of Vegetation Science*, 14: 681-692.
- Lotwick, H.W. & Silverman, B.W. 1982. Methods for analysing spatial processes of several types of points. *Journal of the Royal Statistical Society B*, 44:403-413.
- Ripley B.D. 1977. Modelling spatial patterns. *Journal of the Royal Statistical Society B*, 39:172-192.
- Wiegand, T. & Moloney, K.A. 2004. Rings, circles, and null-models for point pattern analysis in ecology. *Oikos*, 104:209-229.

### See Also

[plot.fads](#), [spp](#), [k12val](#), [kfun](#), [kijfun](#), [ki.fun](#), [kmfun](#).

### Examples

```
data(BPoirier)
BP<-BPoirier
# spatial point pattern in a rectangle sampling window of size [0,110] x [0,90]
swrm<-spp(BP$trees,win=BP$rect,marks=BP$species)
#testing population independence hypothesis
```

```

k12swrm.pi<-k12fun(swrn,25,1,500,marks=c("beech","oak"))
plot(k12swrm.pi)
#testing random labelling hypothesis
k12swrm.rl<-k12fun(swrn,25,1,500,H0="r1",marks=c("beech","oak"))
plot(k12swrm.rl)

# spatial point pattern in a circle with radius 50 centred on (55,45)
swc<-spp(BP$trees,win=c(55,45,45),marks=BP$species)
k12swc.pi<-k12fun(swc,25,1,500,marks=c("beech","oak"))
plot(k12swc.pi)

# spatial point pattern in a complex sampling window
swrt<-spp(BP$trees,win=BP$rect,tri=BP$tri2,marks=BP$species)
k12swrt.pi<-k12fun(swrt,25,1,500,marks=c("beech","oak"))
plot(k12swrt.pi)

```

---

k12val	<i>Multiscale local second-order neighbour density of a bivariate spatial point pattern</i>
--------	---

---

## Description

Computes local second-order neighbour density estimates for a bivariate spatial point pattern, i.e. the number of neighbours of type 2 per unit area within sample circles of regularly increasing radii  $r$ , centred at each type 1 point of the pattern (see Details).

## Usage

```
k12val(p, upto, by, marks)
```

## Arguments

p	a "spp" object defining a multivariate spatial point pattern in a given sampling window (see <a href="#">spp</a> ).
upto	maximum radius of the sample circles (see Details).
by	interval length between successive sample circles radii (see Details).
marks	by default <code>c(1, 2)</code> , otherwise a vector of two numbers or character strings identifying the types (the <code>p\$marks</code> levels) of points of type 1 and 2, respectively.

## Details

Function `K12val` returns individual values of  $KI2(r)$  and associated functions (see [k12fun](#)) estimated at each type 1 point of the pattern. For a given distance  $r$ , these values can be mapped within the sampling window, as in Getis & Franklin 1987 or Pélissier & Goreaud 2001.

**Value**

A list of class `c("vads", "k12val")` with essentially the following components:

<code>r</code>	a vector of regularly spaced distances ( <code>seq(by, upto, by)</code> ).
<code>xy</code>	a data frame with 2 components giving $(x, y)$ coordinates of type 1 points of the pattern.
<code>g12val</code>	a matrix of size $(length(xy), length(r))$ giving individual values of the bivariate pair density function $g12(r)$ .
<code>n12val</code>	a matrix of size $(length(xy), length(r))$ giving individual values of the bivariate neighbour density function $n12(r)$ .
<code>k12val</code>	a matrix of size $(length(xy), length(r))$ giving individual values of the inter-type function $K12(r)$ .
<code>l12val</code>	a matrix of size $(length(xy), length(r))$ giving individual values the modified intertype function $L12(r)$ .

**Note**

There are printing, summary and plotting methods for "vads" objects.

**Author(s)**

⟨Raphael.Pelissier@ird.fr⟩

**References**

Getis, A. and Franklin, J. 1987. Second-order neighborhood analysis of mapped point patterns. *Ecology*, 68:473-477.

Pélissier, R. and Goreaud, F. 2001. A practical approach to the study of spatial structure in simple cases of heterogeneous vegetation. *Journal of Vegetation Science*, 12:99-108.

**See Also**

`plot.vads`, `k12fun`, `dval`, `kval`.

**Examples**

```
data(BPoirier)
BP<-BPoirier
# spatial point pattern in a rectangle sampling window of size [0,110] x [0,90]
swrm<-spp(BP$trees,win=BP$rect,marks=BP$species)
k12vswrm<-k12val(swrm,25,1,marks=c("beech","oak"))
summary(k12vswrm)
plot(k12vswrm)

# spatial point pattern in a circle with radius 50 centred on (55,45)
swc<-spp(BP$trees,win=c(55,45,45),marks=BP$species)
k12vswc<-k12val(swc,25,1,marks=c("beech","oak"))
summary(k12vswc)
```

```

plot(k12vswc)

# spatial point pattern in a complex sampling window
swrt<-spp(BP$trees,win=BP$rect,tri=BP$tri2,marks=BP$species)
k12vswrt<-k12val(swrt,25,1,marks=c("beech","oak"))
summary(k12vswrt)
plot(k12vswrt)

```

---

kfun	<i>Multiscale second-order neighbourhood analysis of an univariate spatial point pattern</i>
------	--

---

### Description

Computes estimates of Ripley's  $K$ -function and associated neighbourhood functions from an univariate spatial point pattern in a simple (rectangular or circular) or complex sampling window. Computes optionally local confidence limits of the functions under the null hypothesis of Complete Spatial Randomness (see Details).

### Usage

```
kfun(p, upto, by, nsim = 0, prec = 0.01, alpha = 0.01)
```

### Arguments

p	a "spp" object defining a spatial point pattern in a given sampling window (see <a href="#">spp</a> ).
upto	maximum radius of the sample circles (see Details).
by	interval length between successive sample circles radii (see Details).
nsim	number of Monte Carlo simulations to estimate local confidence limits of the null hypothesis of complete spatial randomness (CSR) (see Details). By default nsim=0, so that no confidence limits are computed.
prec	if nsim>0, precision of points' coordinates generated during simulations. By default prec=0.01.
alpha	if nsim>0, significant level of the confidence limits. By default $\alpha = 0.01$ .

### Details

Function `kfun` computes Ripley's  $K(r)$  function of second-order neighbourhood analysis and the associated functions  $g(r)$ ,  $n(r)$  and  $L(r)$ .

For a homogeneous isotropic point process of intensity  $\lambda$ , Ripley (1977) showed that the second-order property could be characterized by a function  $K(r)$ , so that the expected number of neighbours within a distance  $r$  of an arbitrary point of the pattern is:  $N(r) = \lambda * K(r)$ .

$K(r)$  is a intensity standardization of  $N(r)$ , which has an expectation of  $\pi * r^2$  under the null

hypothesis of CSR:  $K(r) = N(r)/\lambda$ .

$n(r)$  is an area standardization of  $N(r)$ , which has an expectation of  $\lambda$  under the null hypothesis of CSR:  $n(r) = N(r)/(\pi * r^2)$ , where  $\pi * r^2$  is the area of the disc of radius  $r$ .

$L(r)$  is a linearized version of  $K(r)$  (Besag 1977), which has an expectation of 0 under the null hypothesis of CSR:  $L(r) = \sqrt{K(r)/\pi} - r$ .  $L(r)$  becomes positive when the pattern tends to clustering and negative when it tends to regularity.

$g(r)$  is the derivative of  $K(r)$  or pair density function (Stoyan et al. 1987), so that the expected number of neighbours at a distance  $r$  of an arbitrary point of the pattern (i.e. within an annuli between two successive circles with radii  $r$  and  $r - by$ ) is:  $O(r) = \lambda * g(r)$ .

The program introduces an edge effect correction term according to the method proposed by Ripley (1977) and extended to circular and complex sampling windows by Goreaud & Pélissier (1999).

Theoretical values under the null hypothesis of CSR as well as local Monte Carlo confidence limits and p-values of departure from CSR (Besag & Diggle 1977) are estimated at each distance  $r$ .

## Value

A list of class "fads" with essentially the following components:

r	a vector of regularly spaced out distances (seq(by, upto, by)).
g	a data frame containing values of the pair density function $g(r)$ .
n	a data frame containing values of the local neighbour density function $n(r)$ .
k	a data frame containing values of Ripley's function $K(r)$ .
l	a data frame containing values of the modified Ripley's function $L(r)$ .

Each component except r is a data frame with the following variables:

obs	a vector of estimated values for the observed point pattern.
theo	a vector of theoretical values expected for a Poisson pattern.
sup	(optional) if nsim>0 a vector of the upper local confidence limits of a Poisson pattern at a significant level $\alpha$ .
inf	(optional) if nsim>0 a vector of the lower local confidence limits of a Poisson pattern at a significant level $\alpha$ .
pval	(optional) if nsim>0 a vector of local p-values of departure from a Poisson pattern.

**Warning**

Function `kfun` ignores the marks of multivariate and marked point patterns, which are analysed as univariate patterns.

**Note**

There are printing and plotting methods for "fads" objects.

**Author(s)**

⟨Raphael.Pelissier@ird.fr⟩

**References**

- Besag J.E. 1977. Discussion on Dr Ripley's paper. *Journal of the Royal Statistical Society B*, 39:193-195.
- Besag J.E. & Diggle P.J. 1977. Simple Monte Carlo tests spatial patterns. *Applied Statistics*, 26:327-333.
- Goreaud F. & Pélissier R. 1999. On explicit formulas of edge effect correction for Ripley's K-function. *Journal of Vegetation Science*, 10:433-438.
- Ripley B.D. 1977. Modelling spatial patterns. *Journal of the Royal Statistical Society B*, 39:172-192.
- Stoyan D., Kendall W.S. & Mecke J. 1987. *Stochastic geometry and its applications*. Wiley, New-York.

**See Also**

[plot.fads](#), [spp](#), [kval](#), [k12fun](#), [kijfun](#), [ki.fun](#), [kmfun](#).

**Examples**

```
data(BPoirier)
BP<-BPoirier
# spatial point pattern in a rectangle sampling window of size [0,110] x [0,90]
swr<-spp(BP$trees,win=BP$rect)
kswr<-kfun(swr,25,1,500)
plot(kswr)

# spatial point pattern in a circle with radius 50 centred on (55,45)
swc<-spp(BP$trees,win=c(55,45,45))
kswc<-kfun(swc,25,1,500)
plot(kswc)

# spatial point pattern in a complex sampling window
swrt<-spp(BP$trees,win=BP$rect,tri=BP$tril)
kswrt<-kfun(swrt,25,1,500)
plot(kswrt)
```

---

ki.fun	<i>Multiscale second-order neighbourhood analysis of a multivariate spatial point pattern</i>
--------	---

---

### Description

Computes a set of  $K12$ -functions between all possible marks  $i$  and the other marks in a multivariate spatial point pattern defined in a simple (rectangular or circular) or complex sampling window (see Details).

### Usage

```
ki.fun(p, upto, by)
```

### Arguments

p	a "spp" object defining a multivariate spatial point pattern in a given sampling window (see <a href="#">spp</a> ).
upto	maximum radius of the sample circles (see Details).
by	interval length between successive sample circles radii (see Details).

### Details

Function `ki.fun` is simply a wrapper to `k12fun`, which computes  $K12(r)$  between each mark  $i$  of the pattern and all other marks grouped together (the  $j$  points).

### Value

A list of class "fads" with essentially the following components:

r	a vector of regularly spaced distances ( <code>seq(by, upto, by)</code> ).
labi	a vector containing the levels $i$ of <code>p\$marks</code> .
gi.	a data frame containing values of the pair density function $g12(r)$ .
ni.	a data frame containing values of the local neighbour density function $n12(r)$ .
ki.	a data frame containing values of the $K12(r)$ function.
li.	a data frame containing values of the modified $L12(r)$ function.

Each component except `r` is a data frame with the following variables:

obs	a vector of estimated values for the observed point pattern.
theo	a vector of theoretical values expected under the null hypothesis of population independence (see <a href="#">k12fun</a> ).

**Note**

There are printing and plotting methods for "fads" objects.

**Author(s)**

⟨Raphael.Pelissier@ird.fr⟩

**See Also**

[plot.fads](#), [spp](#), [kfun](#), [k12fun](#), [kijfun](#).

**Examples**

```
data(BPoirier)
BP<-BPoirier
# multivariate spatial point pattern in a rectangle sampling window
swrm<-spp(BP$trees,win=BP$rect,marks=BP$species)
ki.swrm<-ki.fun(swrn,25,1)
plot(ki.swrm)

# multivariate spatial point pattern in a circle with radius 50 centred on (55,45)
swcm<-spp(BP$trees,win=c(55,45,45),marks=BP$species)
ki.swcm<-ki.fun(swcm,25,1)
plot(ki.swcm)

# multivariate spatial point pattern in a complex sampling window
swrtm<-spp(BP$trees,win=BP$rect,tri=BP$tri2,marks=BP$species)
ki.swrtm<-ki.fun(swrtm,25,1)
plot(ki.swrtm)
```

---

kijfun

*Multiscale second-order neighbourhood analysis of a multivariate spatial point pattern*

---

**Description**

Computes a set of  $K$ - and  $K12$ -functions for all possible pairs of marks  $(i, j)$  in a multivariate spatial point pattern defined in a simple (rectangular or circular) or complex sampling window (see Details).

**Usage**

```
kijfun(p, upto, by)
```

**Arguments**

**p** a "spp" object defining a multivariate spatial point pattern in a given sampling window (see [spp](#)).

**upto** maximum radius of the sample circles (see Details).

**by** interval length between successive sample circles radii (see Details).

**Details**

Function `kijfun` is simply a wrapper to `kfun` and `k12fun`, which computes either  $K(r)$  for points of mark  $i$  when  $i = j$  or  $K12(r)$  between the marks  $i$  and  $j$  otherwise.

**Value**

A list of class "fads" with essentially the following components:

<code>r</code>	a vector of regularly spaced distances ( <code>seq(by, upto, by)</code> ).
<code>labij</code>	a vector containing the $(i, j)$ paired levels of <code>p\$marks</code> .
<code>gij</code>	a data frame containing values of the pair density functions $g(r)$ and $g12(r)$ .
<code>nij</code>	a data frame containing values of the local neighbour density functions $n(r)$ and $n12(r)$ .
<code>kij</code>	a data frame containing values of the $K(r)$ and $K12(r)$ functions.
<code>lij</code>	a data frame containing values of the modified $L(r)$ and $L12(r)$ functions.

Each component except `r` is a data frame with the following variables:

<code>obs</code>	a vector of estimated values for the observed point pattern.
<code>theo</code>	a vector of theoretical values expected under the null hypotheses of spatial randomness (see <code>kfun</code> ) and population independence (see <code>kijfun</code> ).

**Note**

There are printing and plotting methods for "fads" objects.

**Author(s)**

`<Raphael.Pelissier@ird.fr>`

**See Also**

`plot.fads`, `spp`, `kfun`, `k12fun`, `ki.fun`.

**Examples**

```
data(BPoirier)
BP<-BPoirier
# multivariate spatial point pattern in a rectangle sampling window
swrm<-spp(BP$trees, win=BP$rect, marks=BP$species)
kijswrm<-kijfun(swrm, 25, 1)
plot(kijswrm)

# multivariate spatial point pattern in a circle with radius 50 centred on (55, 45)
swcm<-spp(BP$trees, win=c(55, 45, 45), marks=BP$species)
kijswcm<-kijfun(swcm, 25, 1)
```

```

plot(kijswcm)

# multivariate spatial point pattern in a complex sampling window
swrtm<-spp(BP$trees,win=BP$rect,tri=BP$tri2,marks=BP$species)
kijswrtm<-kijfun(swrtm,25,1)
plot(kijswrtm)

```

---

kmfun	<i>Multiscale second-order neighbourhood analysis of a marked spatial point pattern</i>
-------	---

---

### Description

Computes estimates of the mark correlation  $Km$ -function and associated neighbourhood functions from a marked spatial point pattern in a simple (rectangular or circular) or complex sampling window. Computes optionally local confidence limits of the functions under the null hypothesis of no correlation between marks (see Details).

### Usage

```
kmfun(p, upto, by, nsim = 0, alpha = 0.01)
```

### Arguments

p	a "spp" object defining a marked spatial point pattern in a given sampling window (see <a href="#">spp</a> ).
upto	maximum radius of the sample circles (see Details).
by	interval length between successive sample circles radii (see Details).
nsim	number of Monte Carlo simulations to estimate local confidence limits of the null hypothesis of no correlation between marks (see Details). By default <code>nsim=0</code> , so that no confidence limits are computed.
alpha	if <code>nsim&gt;0</code> , significant level of the confidence limits. By default $\alpha = 0.01$ .

### Details

Function `kmfun` computes the mark correlation function  $Km(r)$  and the associated function  $gm(r)$ .

It is defined from a general definition of spatial autocorrelation (Goreaud 2000) as:

$$Km(r) = (COV(X_i, X_j) | d(i, j) < r) / VAR(X)$$

where  $X$  is a quantitative random variable attached to each point of the pattern.

$Km(r)$  has a very similar interpretation than more classical correlation functions, such as Moran's  $I$ : it takes values between -1 and 1, with an expectation of 0 under the null hypothesis of no spatial

correlation between the values of  $X$ , becomes positive when values of  $X$  at distance  $r$  are positively correlated and negative when values of  $X$  at distance  $r$  are negatively correlated.

$gm(r)$  is the derivative of  $Km(r)$  or pair mark correlation function, which gives the correlation of marks within an annuli between two successive circles with radii  $r$  and  $r - by$ ).

The program introduces an edge effect correction term according to the method proposed by Ripley (1977) and extended to circular and complex sampling windows by Goreaud & Pélissier (1999).

Local Monte Carlo confidence limits and p-values of departure from the null hypothesis of no correlation are estimated at each distance  $r$ , after reallocating at random the values of  $X$  over all points of the pattern, the location of trees being kept unchanged.

### Value

A list of class "fads" with essentially the following components:

<code>r</code>	a vector of regularly spaced out distances ( <code>seq(by, upto, by)</code> ).
<code>gm</code>	a data frame containing values of the pair mark correlation function $gm(r)$ .
<code>km</code>	a data frame containing values of the mark correlation function $Km(r)$ .

Each component except `r` is a data frame with the following variables:

<code>obs</code>	a vector of estimated values for the observed point pattern.
<code>theo</code>	a vector of theoretical values expected for the null hypothesis of no correlation between marks.
<code>sup</code>	(optional) if <code>nsim&gt;0</code> a vector of the upper local confidence limits of the null hypothesis at a significant level $\alpha$ .
<code>inf</code>	(optional) if <code>nsim&gt;0</code> a vector of the lower local confidence limits of the null hypothesis at a significant level $\alpha$ .
<code>pval</code>	(optional) if <code>nsim&gt;0</code> a vector of local p-values of departure from the null hypothesis.

### Note

Applications of this function can be found in Oddou-Muratorio *et al.* (2004) and Madelaine *et al.* (submitted).

### Author(s)

<Raphael.Pelissier@ird.fr>

## References

Goreaud, F. 2000. *Apports de l'analyse de la structure spatiale en forêt tempérée à l'étude et la modélisation des peuplements complexes*. Thèse de doctorat, ENGREF, Nancy, France.

Goreaud F. & Pélissier R. 1999. On explicit formulas of edge effect correction for Ripley's K-function. *Journal of Vegetation Science*, 10:433-438.

Madelaine, C., Pélissier, R., Vincent, G., Molino, J.-F., Sabatier, D., Prévost, M.-F. & de Namur, C. 2006. Mortality and recruitment in a lowland tropical rainforest of French Guiana: effects of soil types and species guilds. *Journal of Tropical Ecology*, submitted. Oddou-Muratorio, S., Demesure-Musch, B., Pélissier, R. & Gouyon, P.-H. 2004. Impacts of gene flow and logging history on the local genetic structure of a scattered tree species, *Sorbus torminalis* L. *Molecular Ecology*, 13:3689-3702.

Ripley B.D. 1977. Modelling spatial patterns. *Journal of the Royal Statistical Society B*, 39:172-192.

## See Also

`plot.fads`, `spp`, `kfun`, `k12fun`, `kijfun`, `ki.fun`.

## Examples

```
data(BPoirier)
BP<-BPoirier
# spatial point pattern in a rectangle sampling window of size [0,110] x [0,90]
swrm<-spp(BP$trees,win=BP$rect,marks=BP$dbh)
kmswr<-kmfun(swrm,25,1,500)
plot(kmswr)

# spatial point pattern in a circle with radius 50 centred on (55,45)
swc<-spp(BP$trees,win=c(55,45,45),marks=BP$dbh)
kmswc<-kmfun(swc,25,1,500)
plot(kmswc)

# spatial point pattern in a complex sampling window
swrt<-spp(BP$trees,win=BP$rect,tri=BP$tri2,marks=BP$dbh)
kmswrt<-kmfun(swrt,25,1,500)
plot(kmswrt)
```

**Description**

Computes local second-order neighbour density estimates for an univariate spatial point pattern, i.e. the number of neighbours per unit area within sample circles of regularly increasing radii  $r$ , centred at each point of the pattern (see Details).

**Usage**

```
kval(p, upto, by)
```

**Arguments**

`p` a "spp" object defining a spatial point pattern in a given sampling window (see [spp](#)).

`upto` maximum radius of the sample circles (see Details).

`by` interval length between successive sample circles radii (see Details).

**Details**

Function `kval` returns individual values of  $K(r)$  and associated functions (see [kfun](#)) estimated for each point of the pattern. For a given distance  $r$ , these values can be mapped within the sampling window (Getis & Franklin 1987, Pélissier & Goreaud 2001).

**Value**

A list of class `c("vads", "kval")` with essentially the following components:

`r` a vector of regularly spaced out distances (`seq(by, upto, by)`).

`xy` a data frame with 2 components giving  $(x, y)$  coordinates of points of the pattern.

`gval` a matrix of size  $(length(xy), length(r))$  giving individual values of the pair density function  $g(r)$ .

`nval` a matrix of size  $(length(xy), length(r))$  giving individual values of the neighbour density function  $n(r)$ .

`kval` a matrix of size  $(length(xy), length(r))$  giving individual values of Ripley's function  $K(r)$ .

`lval` a matrix of size  $(length(xy), length(r))$  giving individual values the modified Ripley's function  $L(r)$ .

**Warning**

Function `kval` ignores the marks of multivariate and marked point patterns (they are all considered to be univariate patterns).

**Note**

There are printing, summary and plotting methods for "vads" objects.

**Author(s)**

⟨Raphael.Pelissier@ird.fr⟩

**References**

Getis, A. and Franklin, J. 1987. Second-order neighborhood analysis of mapped point patterns. *Ecology*, 68:473-477.

Pélissier, R. and Goreaud, F. 2001. A practical approach to the study of spatial structure in simple cases of heterogeneous vegetation. *Journal of Vegetation Science*, 12:99-108.

**See Also**

[plot.vads](#), [kfun](#), [dval](#), [k12val](#).

**Examples**

```
data(BPoirier)
BP<-BPoirier
# spatial point pattern in a rectangle sampling window of size [0,110] x [0,90]
swr<-spp(BP$trees,win=BP$rect)
kvswr<-kval(swr,25,1)
summary(kvswr)
plot(kvswr)

# spatial point pattern in a circle with radius 50 centred on (55,45)
swc<-spp(BP$trees,win=c(55,45,45))
kvswc<-kval(swc,25,1)
summary(kvswc)
plot(kvswc)

# spatial point pattern in a complex sampling window
swrt<-spp(BP$trees,win=BP$rect,tri=BP$tri1)
kvswrt<-kval(swrt,25,1)
summary(kvswrt)
plot(kvswrt)
```

---

plot.fads

*Plot second-order neighbourhood functions*

---

**Description**

Plot second-order neighbourhood function estimates returned by functions [kfun](#), [kfun](#), [kfun](#), [kfun](#) or [kfun](#).

**Usage**

```
## S3 method for class 'fads':
plot(x,opt,cols,lty,main,sub,legend,csize,...)
```

**Arguments**

<code>x</code>	an object of class "fads" (see Details).
<code>opt</code>	one of <code>c("all", "L", "K", "n", "g")</code> to display either all or one of the functions in a single window. By default <code>opt = "all"</code> for fads objects of subclass "kfun", "k12fun", or "kmfun"; by default <code>opt = "L"</code> for fads objects of subclass "kij", or "ki".
<code>cols</code>	(optional) colours used for plotting functions.
<code>lty</code>	(optional) line types used for plotting functions.
<code>main</code>	by default, the value of argument <code>x</code> , otherwise a text to be displayed as a title of the plot. <code>main=NULL</code> displays no title.
<code>sub</code>	by default, the name of the function displayed, otherwise a text to be displayed as function subtitle. <code>sub=NULL</code> displays no subtitle.
<code>legend</code>	If <code>legend = TRUE</code> (the default) a legend for the plotting functions is displayed.
<code>csize</code>	scaling factor for font size so that actual font size is <code>par("cex") * csize</code> . By default <code>csize = 1</code> .
<code>...</code>	extra arguments that will be passed to the plotting functions <code>plot.swin</code> , <code>plot.default</code> , <code>symbols</code> and/or <code>points</code> .

**Details**

Function `plot.fads` displays second-order neighbourhood function estimates as a function of interpoint distance, with expected values as well as confidence interval limits when computed. Argument `x` can be any fads object returned by functions `kfun`, `kfun`, `kfun`, `kfun` or `kfun`.

**Value**

none.

**Author(s)**

⟨Raphael.Pelissier@ird.fr⟩

**See Also**

`kfun`, `k12fun`, `kmfun`, `kijfun`, `ki.fun`.

**Examples**

```
data(BPoirier)
BP<-BPoirier
# Ripley's function
swr<-spp(BP$trees,win=BP$rect)
k.swr<-kfun(swr,25,1,500)
plot(k.swr)

# Intertype function
```

```

swrm<-spp(BP$trees,win=BP$rect,marks=BP$species)
k12.swrm<-k12fun(swrm,25,1,500,marks=c("beech","oak"))
plot(k12.swrm,opt="L",cols=1)

# Mark correlation function
swrm<-spp(BP$trees,win=BP$rect,marks=BP$dbh)
km.swrm<-kmfun(swrm,25,1,500)
plot(km.swrm,main="Example 1",sub=NULL,legend=FALSE)

```

---

plot.spp

---

*Plot a Spatial Point Pattern object*


---

## Description

Plot a Spatial Point Pattern object returned by function [spp](#).

## Usage

```

## S3 method for class 'spp':
plot(x, main, out = FALSE, use.marks = TRUE, cols, chars, cols.out, chars.out, maxsize)

```

## Arguments

<code>x</code>	an object of class "spp" (see <a href="#">spp</a> ).
<code>main</code>	by default, the value of argument <code>x</code> , otherwise a text to be displayed as a title of the plot. <code>main=NULL</code> displays no title.
<code>out</code>	by default <code>out = FALSE</code> . If <code>TRUE</code> points of the pattern located outside the sampling window are plotted.
<code>use.marks</code>	by default <code>use.marks = TRUE</code> . If <code>FALSE</code> different symbols are not used for each mark of multivariate or marked point patterns, so that they are plotted as univariate (see <a href="#">spp</a> ).
<code>cols</code>	(optional) the colour(s) used to plot points located inside the sampling window (see <a href="#">Details</a> ).
<code>chars</code>	(optional) plotting character(s) used to plot points located inside the sampling window (see <a href="#">Details</a> ).
<code>cols.out</code>	(optional) if <code>out = TRUE</code> , the colour(s) used to plot points located outside the sampling window (see <a href="#">Details</a> ).
<code>chars.out</code>	(optional) if <code>out = TRUE</code> , plotting character(s) used to plot points located outside the sampling window (see <a href="#">Details</a> ).
<code>maxsize</code>	(optional) maximum size of plotting symbols. By default <code>maxsize</code> is automatically adjusted to plot size.
<code>csize</code>	scaling factor for font size so that actual font size is <code>par("cex") * csize</code> . By default <code>csize = 1</code> .
<code>scale</code>	If <code>scale = TRUE</code> (the default) graduations giving plot size are displayed.

legend	If <code>legend = TRUE</code> (the default) a legend for plot symbols is displayed (multivariate and marked types only).
add	by default <code>add = FALSE</code> . If <code>TRUE</code> a new window is not created and just the points are plotted over the existing plot.
...	extra arguments that will be passed to the plotting functions <code>plot.default</code> , <code>points</code> and/or <code>symbols</code> .

## Details

The sampling window `x$window` is plotted first, through a call to function `plot.swin`. Then the points themselves are plotted, in a fashion that depends on the type of spatial point pattern (see `spp`).

- **univariate pattern:** if `x$type = c("univariate")`, i.e. the point pattern does not have marks, or if `use.marks = FALSE`, then the locations of all points is plotted using a single plot character.
- **multivariate pattern:** if `x$type = c("multivariate")`, i.e. the marks are levels of a factor, then each level is represented by a different plot character.
- **marked pattern:** if `x$type = c("marked")`, i.e. the marks are real numbers, then points are represented by circles (argument `chars = "circles"`, the default) or squares (argument `chars = "squares"`) proportional to their marks' value (positive values are filled, while negative values are unfilled).

Arguments `cols` and `cols.out` (if `out = TRUE`) determine the colour(s) used to display the points located inside and outside the sampling window, respectively. Colours may be specified as codes or colour names (see `HYPERLINK(par("col"))(par("col"))`). For univariate and marked point patterns, `cols` and `cols.out` are single character strings, while for multivariate point patterns they are character vectors of same length as `levels(x$marks)` and `levels(x$marksout)`, respectively.

Arguments `chars` and `chars.out` (if `out = TRUE`) determine the symbol(s) used to display the points located inside and outside the sampling window, respectively. Symbols may be specified as codes or character strings (see `HYPERLINK(par("pch"))(par("pch"))`). For univariate point patterns, `chars` and `chars.out` are single character strings, while for multivariate point patterns they are character vectors of same length as `levels(x$marks)` and `levels(x$marksout)`, respectively. For marked point patterns, `chars` and `chars.out` can only take the value `"circles"` or `"squares"`.

## Value

none.

## Author(s)

⟨Raphael.Pelissier@mpl.fr⟩

## See Also

`spp`, `swin`, `plot.swin`.

**Examples**

```

data(BPoirier)
BP<-BPoirier

# a univariate point pattern in a rectangle sampling window
plot(spp(BP$trees,win=BP$rect))

# a univariate point pattern in a circular sampling window
#with all points and graduations displayed
plot(spp(BP$trees,win=c(55,45,45)),out=TRUE,scale=TRUE)

# a univariate point pattern in a complex sampling window
#with points outside the sampling window displayed (in red colour)
plot(spp(BP$trees,win=BP$rect,tri=BP$tril),out=TRUE)

# a multivariate point pattern in a rectangle sampling window
plot(spp(BP$trees,win=BP$rect,marks=BP$species))

# a multivariate point pattern in a circular sampling window
#with all points inside the sampling window displayed in blue colour
#and all points outside displayed with the symbol "+" in red colour
plot(spp(BP$trees,win=c(55,45,45),marks=BP$species),out=TRUE,cols=c("blue","blue","blue"),c

# a marked point pattern in a rectangle sampling window
#with circles in green colour
plot(spp(BP$trees,win=BP$rect,marks=BP$dbh),cols="green")

# a marked point pattern in a circular sampling window
#with squares in red colour inside and circles in blue colour outside
plot(spp(BP$trees,win=c(55,45,45),marks=BP$dbh),out=TRUE,chars="squares",cols="red",cols.c

```

---

plot.vads

*Plot local density values*


---

**Description**

Plot local density estimates returned by functions `dval`, `dval` or `dval`.

**Usage**

```

## S3 method for class 'vads':
plot(x, main, opt, select, chars, cols, maxsize, char0, col0, legend, csize, ...)

```

**Arguments**

`x` an object of class 'vads' (see Details).

`main` by default, the value of argument `x`, otherwise a text to be displayed as a title of the plot. `main=NULL` displays no title.

opt	(optional) a character string to change the type of values to be plotted (see Details).
select	(optional) a vector of selected distances in <code>x\$r</code> . By default, a multiple window displays all distances.
chars	one of <code>c("circles", "squares")</code> plotting symbols with areas proportional to local density values. By default, circles are plotted.
cols	(optional) the colour used for the plotting symbols. Black colour is the default.
maxsize	(optional) maximum size of the circles/squares plotted. By default, maxsize is automatically adjusted to plot size.
char0	(optional) the plotting symbol used to represent null values. By default, null values are not plotted.
col0	(optional) the colour used for the null values plotting symbol. By default, the same as argument <code>cols</code> .
legend	If <code>legend = TRUE</code> (the default) a legend for the plotting values is displayed.
csize	scaling factor for font size so that actual font size is <code>par("cex") * csize</code> . By default <code>csize = 1</code> .
...	extra arguments that will be passed to the plotting functions <code>plot.swin</code> , <code>plot.default</code> , <code>symbols</code> and/or <code>points</code> .

### Details

Function `plot.vads` displays a map of first-order local density or second-order local neighbour density values as symbols with areas proportional to the values estimated at the plotted points. Positive values are represented by coloured symbols, while negative values are represented by open symbols. The plotted function values depend upon the type of 'vads' object:

- if `class(x) = c("vads", "dval")`, the plotted values are first-order local densities and argument `opt="dval"` by default, but is potentially one of `c("dval", "cval")` returned by `dval`.
- if `class(x) = c("vads", "kval")` or `class(x) = c("vads", "k12val")`, the plotted values are univariate or bivariate second-order local neighbour densities. Argument `opt="lval"` by default, but is potentially one of `c("lval", "kval", "nval", "gval")` returned by `kval` and `k12val`.

### Value

none.

### Author(s)

⟨Raphael.Pelissier@ird.fr⟩

### See Also

`dval`, `kval`, `k12val`.

## Examples

```

data(BPoirier)
BP<-BPoirier
# local density in a rectangle sampling window
dswr<-dval(spp(BP$trees,win=BP$rect),25,1,11,9)
plot(dswr)
# display only distance r from 5 to 10 with null symbols as red crosses
plot(dswr,select=c(5:10),char0=3,col0="red")

# local L(r) values in a circular sampling window
lvswc<-kval(spp(BP$trees,win=c(55,45,45)),25,0.5)
plot(lvswc)
# display square symbols in blue for selected values of r and remove title
plot(lvswc,chars="squares",cols="blue",select=c(5,7.5,10,12.5,15),main=NULL)

# local K12(r) values (1="beech", 2="oak") in a complex sampling window
k12swrt<-k12val(spp(BP$trees,win=BP$rect,tri=BP$tri1,marks=BP$species),25,1)
plot(k12swrt,opt="kval")

```

---

spp

---

*Creating a spatial point pattern*


---

## Description

Function `spp` creates an object of class "spp", which represents a spatial point pattern observed in a finite sampling window (or study region). The `ads` library supports univariate, multivariate and marked point patterns observed in simple (rectangular or circular) or complex sampling windows.

## Usage

```

spp(x, y=NULL, window)
spp(x, y=NULL, window, triangles)
spp(x, y=NULL, window, marks, int2fac = TRUE)
spp(x, y=NULL, window, triangles, marks, int2fac = TRUE)

```

## Arguments

<code>x, y</code>	if <code>y=NULL</code> , <code>x</code> is a list of two vectors of point coordinates, else both <code>x</code> and <code>y</code> are atomic vectors of point coordinates.
<code>window</code>	a "swin" object or a vector defining the limits of a simple sampling window: <code>c(xmin, ymin, xmax, ymax)</code> for a rectangle ; <code>c(x0, y0, r0)</code> for a circle.
<code>triangles</code>	(optional) a list of triangles removed from a simple initial window to define a complex sampling window (see <a href="#">swin</a> ).
<code>marks</code>	(optional) a vector of mark values, which may be factor levels or numerical values (see <a href="#">Details</a> ).
<code>int2fac</code>	if TRUE, integer marks are automatically coerced into factor levels.

## Details

A spatial point pattern is assumed to have been observed within a specific sampling window (a finite study region) defined by the `window` argument. If `window` is a simple "swin" object, it may be coerced into a complex type by adding a `triangles` argument (see [swin](#)). A spatial point pattern may be of 3 different types.

- **univariate pattern:** by default when argument `marks` is not given.
- **multivariate pattern:** `marks` is a factor, which levels are interpreted as categorical marks (e.g. colours, species, etc.) attached to points of the pattern. Integer marks may be automatically coerced into factor levels when argument `int2fac = TRUE`.
- **marked pattern:** `marks` is a vector of real numbers attached to points of the pattern. Integer values may also be considered as numerical values if argument `int2fac = FALSE`.

## Value

An object of class "spp" describing a spatial point pattern observed in a given sampling window.

`this-is-escaped-codenormal-bracket52bracket-normal`  
 a character string indicating if the spatial point pattern is "univariate", "multivariate" or "marked".

`this-is-escaped-codenormal-bracket58bracket-normal`  
 an swin object describing the sampling window (see [swin](#)).

`this-is-escaped-codenormal-bracket64bracket-normal`  
 an integer value giving the number of points of the pattern located inside the sampling window (points on the boundary are considered to be inside).

`this-is-escaped-codenormal-bracket67bracket-normal`  
 a vector of  $x$  coordinates of points located inside the sampling window.

`this-is-escaped-codenormal-bracket71bracket-normal`  
 a vector of  $y$  coordinates of points located inside the sampling window.

`this-is-escaped-codenormal-bracket75bracket-normal`  
 (optional) an integer value giving the number of points of the pattern located outside the sampling window.

`this-is-escaped-codenormal-bracket78bracket-normal`  
 (optional) a vector of  $x$  coordinates of points located outside the sampling window.

`this-is-escaped-codenormal-bracket82bracket-normal`  
 (optional) a vector of  $y$  coordinates of points located outside the sampling window.

`this-is-escaped-codenormal-bracket86bracket-normal`  
 (optional) a vector of the marks attached to points located inside the sampling window.

`this-is-escaped-codenormal-bracket89bracket-normal`  
 (optional) a vector of the marks attached to points located outside the sampling window.

**Note**

There are printing, summary and plotting methods for "spp" objects.  
 Function `spp2ppp(spp.object)` coerces an object of class "spp" into a `ppp.object` suitable for use with package `spatstat`.

**Author(s)**

⟨Raphael.Pelissier@ird.fr⟩

**References**

Goreaud, F. and Pélissier, R. 1999. On explicit formula of edge effect correction for Ripley's  $K$ -function. *Journal of Vegetation Science*, 10:433-438.

**See Also**

`plot.spp`, `swin`

**Examples**

```
data(BPoirier)
BP<-BPoirier
# univariate pattern in a rectangle of size [0,110] x [0,90]
swr<-spp(BP$trees,win=BP$rect)
# an alternative using atomic vectors of point coordinates
#swr<-spp(BP$trees,win=BP$rect)
summary(swr)
plot(swr)

# univariate pattern in a circle with radius 50 centred on (55,45)
swc<-spp(BP$trees,win=c(55,45,50))
summary(swc)
plot(swc)
plot(swc, out=TRUE) # plot points outside the circle

# multivariate pattern in a rectangle of size [0,110] x [0,90]
swrm<-spp(BP$trees,win=BP$rect,marks=BP$species)
summary(swrm)
plot(swrm)
plot(swrm, chars=c("b","h","o")) # replace symbols by letters

# marked pattern in a rectangle of size [0,110] x [0,90]
swrn<-spp(BP$trees,win=BP$rect,marks=BP$dbh)
summary(swrn)
plot(swrn)

# multivariate pattern in a complex sampling window
swrt<-spp(BP$trees,win=BP$rect,tri=BP$tril,marks=BP$species)
summary(swrt)
plot(swrt)
plot(swrt, out=TRUE) # plot points outside the sampling window
```

swin

*Creating a sampling window***Description**

Function `swin` creates an object of class "swin", which represents the sampling window (or study region) in which a spatial point pattern was observed. The `ads` library supports simple (rectangular or circular) and complex sampling windows.

**Usage**

```
swin(window=c(xmin, ymin, xmax, ymax))
swin(window=c(x0, y0, r0))
swin(window, triangles)
```

**Arguments**

<code>window</code>	a vector defining the limits of a simple sampling window: <code>c(xmin, ymin, xmax, ymax)</code> for a rectangle ; <code>c(x0, y0, r0)</code> for a circle.
<code>triangles</code>	(optional) a list of triangles removed from a simple initial window to define a complex sampling window (see Details).

**Details**

A sampling window may be of simple or complex type. A simple sampling window may be a rectangle or a circle. A complex sampling window is defined by removing triangular surfaces from a simple (rectangular or circular) initial sampling window.

- **rectangular window:** `window=c(xmin, ymin, xmax, ymax)` a vector of length 4 giving the coordinates  $(x_{min}, y_{min})$  and  $(x_{max}, y_{max})$  of the origin and the opposite corner of a rectangle.
- **circular window:** `window=c(x0, y0, r0)` a vector of length 3 giving the coordinates  $(x_0, y_0)$  of the centre and the radius  $r_0$  of a circle.
- **complex window:** `triangles` is a list of 6 variables giving the vertices coordinates  $(ax, ay, bx, by, cx, cy)$  of the triangles to remove from a simple (rectangular or circular) initial window. The triangles may be removed near the boundary of a rectangular window in order to design a polygonal sampling window, or within a rectangle or a circle, to delineating holes in the initial sampling window (see Examples). The triangles do not overlap each other, nor overlap boundary of the initial sampling window. Any polygon (possibly with holes) can be decomposed into contiguous triangles using `triangulate`.

**Value**

An object of class "swin" describing the sampling window. It may be of four different types with different arguments:

```

this-is-escaped-codenormal-bracket39bracket-normal
    a vector of two character strings defining the type of sampling window among
    c("simple", "rectangle"), c("simple", "circle"), c("complex", "rectangle")
    or c("complex", "circle").
this-is-escaped-codenormal-bracket46bracket-normal
    (optional) coordinates of the origin and the opposite corner for a rectangular
    sampling window (see details).
this-is-escaped-codenormal-bracket49bracket-normal
    (optional) coordinates of the center and radius for a circular sampling window
    (see details).
this-is-escaped-codenormal-bracket52bracket-normal
    (optional) vertices coordinates of triangles for a complex sampling window (see
    details).

```

### Note

There are printing, summary and plotting methods for "swin" objects.  
 Function `swin2owin(swin.object)` coerces an object of class "swin" into an `owin.object` suitable for use with package `spatstat`.

### Author(s)

⟨Raphael.Pelissier@ird.fr⟩

### References

Goreaud, F. and Pélissier, R. 1999. On explicit formula of edge effect correction for Ripley's  $K$ -function. *Journal of Vegetation Science*, 10:433-438.

### See Also

[area.swin](#), [inside.swin](#), [swin2owin](#), [spp](#)

### Examples

```

#rectangle of size [0,110] x [0,90]
wr<-swin(c(0,0,110,90))
summary(wr)
plot(wr)

#circle with radius 50 centred on (55,45)
wc<-swin(c(55,45,50))
summary(wc)
plot(wc)

# polygon (diamond shape)
t1<-c(0,0,55,0,0,45)
t2<-c(55,0,110,0,110,45)
t3<-c(0,45,0,90,55,90)
t4<-c(55,90,110,90,110,45)
wp<-swin(wr, rbind(t1,t2,t3,t4))

```

```

summary(wp)
plot(wp)

#rectangle with a hole
h1<-c(25,45,55,75,85,45)
h2<-c(25,45,55,15,85,45)
wrh<-swin(wr, rbind(h1,h2))
summary(wrh)
plot(wrh)

#circle with a hole
wch<-swin(wc, rbind(h1,h2))
summary(wch)
plot(wch)

```

---

triangulate

*Triangulate polygon*


---

### Description

Function `triangulate` decomposes a simple polygon (optionally having holes) into contiguous triangles.

### Usage

```

triangulate(outer.poly)
triangulate(outer.poly, holes)

```

### Arguments

<code>outer.poly</code>	a list with two component vectors $x$ and $y$ giving vertice coordinates of the polygon or a vector $(x_{min}, y_{min}, x_{max}, y_{max})$ giving coordinates $(x_{imn}, y_{min})$ and $(x_{max}, y_{max})$ of the origin and the opposite corner of a rectangle sampling window (see <code>swin</code> ).
<code>holes</code>	(optional) a list (or a list of list) with two component vectors $x$ and $y$ giving vertics coordinates of inner polygon(s) delineating hole(s) within the <code>outer.poly</code> .

### Details

In argument `outer.poly`, the vertices must be listed following boundary of the polygon without any repetition (i.e. do not repeat the first vertex). Argument `holes` may be a list of vertices coordinates of a single hole (i.e. with  $x$  and  $y$  component vectors) or a list of list for multiple holes, where each `holes[[i]]` is a list with  $x$  and  $y$  component vectors. Holes' vertices must all be inside the `outer.poly` boundary (vertices on the boundary are considered outside). Multiple holes do not overlap each others.

**Value**

A list of 6 variables, suitable for using in `swin` and `spp`, and giving the vertices coordinates  $(ax, ay, bx, by, cx, cy)$  of the triangles that pave the polygon. For a polygon with  $t$  holes totalling  $n$  vertices (outer contour + holes), the number of triangles produced is  $(n - 2) + 2t$ , with  $n < 200$  in this version of the program.

**Author(s)**

`<Raphael.Pelissier@ird.fr>`

**References**

Goreaud, F. and Pélissier, R. 1999. On explicit formula of edge effect correction for Ripley's  $K$ -function. *Journal of Vegetation Science*, 10:433-438.

Narkhede, A. & Manocha, D. 1995. Fast polygon triangulation based on Seidel's algorithm. Pp 394-397 In A.W. Paeth (Ed.) *Graphics Gems V*. Academic Press. <http://www.cs.unc.edu/~dm/CODE/GEM/chapter.html>.

**See Also**

`spp`, `swin`

**Examples**

```
data(BPoirier)
BP<-BPoirier
plot(BP$poly1$x, BP$poly1$y)

# a single polygon triangulation
tri1<-triangulate(BP$poly1)
plot(swin(BP$rect, tri1))

# a single polygon with a hole
tri2<-triangulate(c(-10, -10, 120, 100), BP$poly1)
plot(swin(c(-10, -10, 120, 100), tri2))
```

# Index

## \*Topic **datasets**

Allogny, 1  
BPoirier, 3  
Couepia, 4

## \*Topic **spatial**

area.swin, 2  
dval, 5  
inside.swin, 7  
k12fun, 8  
k12val, 11  
kfun, 13  
ki.fun, 16  
kijfun, 18  
kmfun, 19  
kval, 22  
plot.fads, 24  
plot.spp, 25  
plot.vads, 28  
spp, 29  
swin, 32  
triangulate, 34

Allogny, 1  
area.swin, 2, 33

BPoirier, 3

Couepia, 4

dval, 5, 13, 23, 28, 29

inside.swin, 7, 33

k12fun, 8, 12, 13, 15–19, 21, 25  
k12val, 11, 11, 23, 29  
kfun, 11, 13, 17–19, 21–25  
ki.fun, 11, 15, 16, 19, 21, 25  
kijfun, 11, 15, 17, 18, 18, 21, 25  
kmfun, 11, 15, 19, 25  
kval, 13, 15, 22, 29

owin.object, 33

plot.default, 24, 26, 28  
plot.fads, 11, 15, 17, 19, 21, 24  
plot.fads.ki.fun (*plot.fads*), 24  
plot.fads.kijfun (*plot.fads*), 24  
plot.fads.kmfun (*plot.fads*), 24  
plot.spp, 25, 31  
plot.swin, 24, 26–28  
plot.swin (*swin*), 32  
plot.vads, 6, 13, 23, 28  
points, 24, 26, 28  
ppp.object, 31  
print.dval (*dval*), 5  
print.k12val (*k12val*), 11  
print.kval (*kval*), 22  
print.spp (*spp*), 29  
print.summary.dval (*dval*), 5  
print.summary.k12val (*k12val*), 11  
print.summary.kval (*kval*), 22  
print.summary.spp (*spp*), 29  
print.summary.swin (*swin*), 32  
print.swin (*swin*), 32

seq, 6

spp, 5, 6, 8, 11–13, 15–22, 25–27, 29, 33, 35

spp2ppp (*spp*), 29

summary.dval (*dval*), 5

summary.k12val (*k12val*), 11

summary.kval (*kval*), 22

summary.spp (*spp*), 29

summary.swin (*swin*), 32

swin, 2, 3, 6–8, 27, 30, 31, 32, 35

swin2owin, 33

swin2owin (*swin*), 32

symbols, 24, 26, 28

triangulate, 33, 34