

# Package ‘dbmss’

December 11, 2023

**Type** Package

**Title** Distance-Based Measures of Spatial Structures

**Version** 2.9-0

**Description** Simple computation of spatial statistic functions of distance to characterize the spatial structures of mapped objects, following Marcon, Traissac, Puech, and Lang (2015) <[doi:10.18637/jss.v067.c03](https://doi.org/10.18637/jss.v067.c03)>. Includes classical functions (Ripley's K and others) and more recent ones used by spatial economists (Duranton and Overman's Kd, Marcon and Puech's M). Relies on 'spatstat' for some core calculation.

**URL** <https://github.com/EricMarcon/dbmss>

**BugReports** <https://github.com/EricMarcon/dbmss/issues>

**License** GNU General Public License

**Depends** R (>= 3.5.0), Rcpp (>= 0.12.14), spatstat.explore

**Imports** cubature, ggplot2, RcppParallel, reshape2, rlang, spatstat.utils, stats, tibble, spatstat.geom, spatstat.random

**Suggests** testthat, knitr, pkgdown, rmarkdown

**LinkingTo** Rcpp, RcppParallel

**VignetteBuilder** knitr

**SystemRequirements** pandoc, GNU make

**Encoding** UTF-8

**LazyData** true

**NeedsCompilation** yes

**Author** Eric Marcon [aut, cre] (<<https://orcid.org/0000-0002-5249-321X>>), Gabriel Lang [aut] (<<https://orcid.org/0000-0002-4325-6044>>), Stephane Traissac [aut] (<<https://orcid.org/0000-0001-9255-1616>>), Florence Puech [aut] (<<https://orcid.org/0000-0002-5279-6878>>)

**Maintainer** Eric Marcon <[eric.marcon@agroparistech.fr](mailto:eric.marcon@agroparistech.fr)>

**Repository** CRAN

**Date/Publication** 2023-12-11 20:30:02 UTC

**R topics documented:**

dbmss-package	3
as.Dtable	3
as.wmppp	4
autoplot	5
dbmssEnvelope.object	7
DEnvelope	8
Dhat	9
Dtable	11
envelope.Dtable	12
gEnvelope	13
ghat	14
GoFtest	16
is.wmppp	17
KdEnvelope	18
Kdhat	20
KEnvelope	23
Khat	24
KinhomEnvelope	26
Kinhomhat	28
KmmEnvelope	29
Kmmhat	31
Ktest	32
LEnvelope	33
Lhat	35
LmmEnvelope	36
Lmmhat	38
MEnvelope	39
mEnvelope	41
Mhat	43
mhat	45
paracou16	47
print.dbmssEnvelope	48
rPopulationIndependenceK	49
rPopulationIndependenceM	50
rRandomLabeling	51
rRandomLabelingM	52
rRandomLocation	54
rRandomPositionK	55
Smooth.wmppp	56
spatstat generic functions	58
summary.dbmssEnvelope	59
wmppp	60
wmppp.object	61

**Description**

Simple computation of spatial statistic functions of distance to characterize the spatial structures of mapped objects, including classical ones (Ripley's  $K$  and others) and more recent ones used by spatial economists (Duranton and Overman's  $Kd$ , Marcon and Puech's  $M$ ). Relies on spatstat for some core calculation.

**Author(s)**

Eric Marcon, Gabriel Lang, Stephane Traissac, Florence Puech

Maintainer: Eric Marcon <Eric.Marcon@agroparistech.fr>

**References**

Marcon, E., and Puech, F. (2003). Evaluating the Geographic Concentration of Industries Using Distance-Based Methods. *Journal of Economic Geography*, 3(4), 409-428.

Marcon, E. and Puech, F. (2010). Measures of the Geographic Concentration of Industries: Improving Distance-Based Methods. *Journal of Economic Geography* 10(5): 745-762.

Marcon, E., Puech F. and Traissac, S. (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.

Lang G., Marcon E. and Puech F. (2014) Distance-Based Measures of Spatial Concentration: Introducing a Relative Density Function. *HAL* 01082178, 1-18.

Marcon, E., Traissac, S., Puech, F. and Lang, G. (2015). Tools to Characterize Point Patterns: dbmss for R. *Journal of Statistical Software*. 67(3): 1-15.

Marcon, E. and Puech, F. (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

**Description**

Creates an object of class "Dtable" representing a set of points with weights and labels and the distances between them.. This is a generic method.

**Usage**

```
as.Dtable(X, ...)  
## S3 method for class 'ppp'  
as.Dtable(X, ...)  
## S3 method for class 'data.frame'  
as.Dtable(X, ...)
```

**Arguments**

X                    Data to be converted into a "Dtable".  
 ...                  Extra arguments.

**Details**

This is a generic method, implemented for `ppp` and `data.frame`.

Data is first converted to a (`wmppp.object`). Then, the distance matrix between points is calculated and the marks are kept.

**Value**

An object of class "Dtable".

**See Also**

[as.wmppp](#)

---

as.wmppp

*Converts data to class wmppp*

---

**Description**

Creates a Weighted, Marked, Planar Point Pattern, *i.e.* an object of class "wmppp" representing a two-dimensional point pattern with weights and labels. This is a generic method.

**Usage**

```
as.wmppp(X, ...)
## S3 method for class 'ppp'
as.wmppp(X, ...)
## S3 method for class 'data.frame'
as.wmppp(X, window = NULL, unitname = NULL, ...)
```

**Arguments**

X                    Data to be converted into a weighted, marked, planar point pattern (`wmppp.object`)  
 window             An object of calls "owin" (`owin.object`).  
 unitname           Name of unit of length. Either a single character string, or a vector of two character strings giving the singular and plural forms, respectively.  
 ...                  Extra arguments.

**Details**

This is a generic method, implemented for [ppp](#) and [data.frame](#):

- If the dataset  $X$  is an object of class "ppp" ([ppp.object](#)), the marks are converted to point weights if they are numeric or to point types if they are factors. Default weights are set to 1, default types to "All". If marks are a dataframe with column names equal to `PointType` and `PointWeight`, they are not modified. Row names of the dataframe are preserved as row names of the marks, to identify points.
- If the dataset  $X$  is a dataframe, see [wmppp](#).

**Value**

An object of class "wmppp".

**See Also**

[wmppp.object](#)

---

 autoplot

*ggplot methods to plot dbmss objects*


---

**Description**

S3 methods for the [autoplot](#) generic.

**Usage**

```
## S3 method for class 'envelope'
autoplot(object, fmla, ..., ObsColor = "black",
         H0Color = "red", ShadeColor = "grey75", alpha = 0.3, main = NULL,
         xlab = NULL, ylab = NULL, LegendLabels = NULL)
## S3 method for class 'fv'
autoplot(object, fmla, ..., ObsColor = "black",
         H0Color = "red", ShadeColor = "grey75", alpha = 0.3, main = NULL,
         xlab = NULL, ylab = NULL, LegendLabels = NULL)
## S3 method for class 'wmppp'
autoplot(object, ..., show.window = TRUE,
         MaxPointTypes = 6, Other = "Other",
         main = NULL, xlab = NULL, ylab = NULL, LegendLabels = NULL,
         labelSize = "Weight", labelColor = "Type", palette="Set1",
         windowColor = "black", windowFill = "transparent", alpha = 1)
```

**Arguments**

object	An object to be plotted.
fm1a	An R language formula determining which variables or expressions are plotted. Either a formula object, or a string that can be parsed as a formula. See <a href="#">plot.fv</a> .
...	Extra arguments, currently unused.
ObsColor	The color of the line representing observed values of the function.
H0Color	The color of the line representing the null hypothesis values of the function.
ShadeColor	The color of the confidence envelope.
alpha	The opacity of the confidence envelope (in function values) or the points (in maps), between 0 and 1.
main	The title of the plot.
xlab, ylab	The axes labels.
LegendLabels	A vector of characters. The first two items describe the observed and null-hypothesis curves, the third and last item the confidence interval. To be used only in plots with two curves (typically observed and expected values). The default is 'NULL' to display the full description of functions.
show.window	if 'TRUE', the borders of the window containing the points are shown on the point map.
MaxPointTypes	The maximum number of different point types to show. If the point set contains more of them, the less frequent ones are gathered as "Other". This number must be limited for readability and not to exceed the number of colors offered by the palette.
Other	The name of the point types gathered as "Other".
labelSize	The guide of the point size legend in point maps, i.e. what the 'PointSize' mark represents.
labelColor	The guide of the point color legend in point maps, i.e. what the 'PointType' mark represents.
palette	The color palette used to display point types in maps. See <a href="#">scale_colour_brewer</a>
windowColor	The color used to draw the limits of the windows in point maps.
windowFill	The color used to fill the windows in point maps.

**Details**

Plots of 'wmppp' objects are a single representation of both point types and point weights. Rectangular and polygonal windows (see [owin.object](#)) are supported but mask windows are ignored (use the 'plot' method if necessary).

**Value**

A [ggplot](#) object.

**Author(s)**

Eric Marcon <[Eric.Marcon@agroparistech.fr](mailto:Eric.Marcon@agroparistech.fr)>, parts of the code from `spatstat.explore::plot.fv`.

## Examples

```
data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X)

# Plot the envelope (should be 1000 simulations, reduced to 20 to save time)
autoplot(KdEnvelope(X, ReferenceType="Q. Rosea", NumberOfSimulations=20))

# With a formula and a compact legend
autoplot(KEnvelope(X, NumberOfSimulations=20),
         ./(pi*r^2) ~ r,
         LegendLabels=c("Observed", "Expected", "Confidence\n envelope"))
```

---

dbmssEnvelope.object    *Class of envelope of function values (fv)*

---

## Description

A class "dbmssEnvelope", *i.e.* a particular type of see [envelope](#) to represent several estimates of the same function and its confidence envelope.

## Details

"dbmssEnvelope" objects are similar to envelope objects. The differences are that the risk level is chosen (instead of the simulation rank to use as the envelope), so the rank is calculated (interpolation is used if necessary), and a global envelope can be calculated following Durantón and Overman (2005).

## References

Durantón, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106

## See Also

[summary.dbmssEnvelope](#), [KdEnvelope](#), [MEnvelope](#)

---

DEnvelope	<i>Estimation of the confidence envelope of the D function under its null hypothesis</i>
-----------	--

---

### Description

Simulates point patterns according to the null hypothesis and returns the envelope of  $D$  according to the confidence level.

### Usage

```
DEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05,
          Cases, Controls, Intertype = FALSE, Global = FALSE,
          verbose = interactive())
```

### Arguments

<code>X</code>	A point pattern ( <a href="#">wmppp.object</a> ).
<code>r</code>	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
<code>NumberOfSimulations</code>	The number of simulations to run, 100 by default.
<code>Alpha</code>	The risk level, 5% by default.
<code>Cases</code>	One of the point types
<code>Controls</code>	One of the point types.
<code>Intertype</code>	Logical; if TRUE, $D$ is computed as $D_i$ in Marcon and Puech (2012).
<code>Global</code>	Logical; if TRUE, a global envelope sensu Durantou and Overman (2005) is calculated.
<code>verbose</code>	Logical; if TRUE, print progress reports during the simulations.

### Details

The only null hypothesis is random labeling: marks are distributed randomly across points.

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until  $Alpha / Number\ of\ simulations$  simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

### Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The `f` contains the observed value of the function, its average simulated value and the confidence envelope.



## References

Durantón, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.

Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.

Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.

Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

## See Also

[Dhat](#)

## Examples

```
data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X,
  labelSize = expression("Basal area ( ~cm^2~ ") ),
  labelColor = "Species")

# Calculate confidence envelope (should be 1000 simulations, reduced to 20 to save time)
r <- 0:30
NumberOfSimulations <- 20
Alpha <- .05
# Plot the envelope (after normalization by pi.r^2)
autoplot(DEnvelope(X, r, NumberOfSimulations, Alpha,
  "V. Americana", "Q. Rosea", Intertype = TRUE), ./(pi*r^2) ~ r)
```

---

Dhat

*Estimation of the D function*

---

## Description

Estimates the *D* function

## Usage

```
Dhat(X, r = NULL, Cases, Controls = NULL, Intertype = FALSE, CheckArguments = TRUE)
```

**Arguments**

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
Cases	One of the point types.
Controls	One of the point types. If NULL, controls are all types except for cases.
Intertype	Logical; if TRUE, $D$ is computed as $D_i$ in Marcon and Puech (2012).
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

The  $D_i$  function allows comparing the structure of the cases to that of the controls around cases, that is to say the comparison is made around the same points. This has been advocated by Arbia et al. (2008) and formalized by Marcon and Puech (2012).

**Value**

An object of class `fv`, see [fv.object](#), which can be plotted directly using [plot.fv](#).

**Note**

The computation of `Dhat` relies on `spatstat` functions [Kest](#) and [Kcross](#).

**References**

- Arbia, G., Espa, G. and Quah, D. (2008). A class of spatial econometric methods in the empirical analysis of clusters of firms in the space. *Empirical Economics* 34(1): 81-103.
- Diggle, P. J. and Chetwynd, A. G. (1991). Second-Order Analysis of Spatial Clustering for Inhomogeneous Populations. *Biometrics* 47(3): 1155-1163.
- Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

**See Also**

[Khat](#), [DEnvelope](#), [Kest](#), [Kcross](#)

**Examples**

```
data(paracou16)
autoplot(paracou16)

# Calculate D
r <- 0:30
(Paracou <- Dhat(paracou16, r, "V. Americana", "Q. Rosea", Intertype = TRUE))

# Plot (after normalization by pi*r^2)
autoplot(Paracou, ./(pi*r^2) ~ r)
```

---

Dtable	<i>Create a Distance table object.</i>
--------	--

---

**Description**

Creates an object of class "Dtable" representing a set of points with weights and labels and the distances between them.

**Usage**

```
Dtable(Dmatrix, PointType = NULL, PointWeight = NULL)
```

**Arguments**

Dmatrix	A square matrix containing distances.
PointType	A vector describing the point types. Its length must correspond to the number of points.
PointWeight	A vector describing the point weights. Its length must correspond to the number of points.

**Details**

The distance matrix is not necessarily symmetric, so distances are understood in the common sense, not in the mathematical sense. Asymmetric distances are appropriate when paths between points are one-way only.

The points of origin are in lines, the targets in columns. The diagonal of the matrix must contain zeros (the distance between a point and itself is 0), and all other distances must be positive (they can be 0).

**Value**

An object of class "Dtable". It is a list:

Dmatrix	The distance matrix.
n	The number of points.
marks	A list of two items: PointType, a vector of factors containing the point types and PointWeight, the numeric vector of weights.

**See Also**

[as.Dtable](#)

**Examples**

```
# A Dtable containing two points
Dmatrix <- matrix(c(0,1,1,0), nrow=2)
PointType <- c("Type1", "Type2")
PointWeight <- c(2,3)
Dtable(Dmatrix, PointType, PointWeight)
```

---

envelope.Dtable	<i>Computes simulation envelopes of a summary function.</i>
-----------------	---

---

**Description**

Prints a useful summary of a confidence envelope of class "dbmssEnvelope"

**Usage**

```
## S3 method for class 'Dtable'
envelope(Y, fun = Kest, nsim = 99, nrank = 1, ...,
         funargs = list(), funYargs = funargs, simulate = NULL,
         verbose = TRUE, savefuns = FALSE, Yname = NULL, envir.simul = NULL)
```

**Arguments**

Y	An object of class <a href="#">Dtable</a> .
fun	Function that computes the desired summary statistic for Y.
nsim	Number of simulated point patterns to be generated when computing the envelopes.
nrank	Integer. Rank of the envelope value amongst the nsim simulated values. A rank of 1 means that the minimum and maximum simulated values will be used.
...	Extra arguments passed to fun.
funargs	A list, containing extra arguments to be passed to fun.
funYargs	Optional. A list, containing extra arguments to be passed to fun when applied to the original data Y only.
simulate	Optional. Specifies how to generate the simulated point patterns.
verbose	Logical flag indicating whether to print progress reports during the simulations.
savefuns	Logical flag indicating whether to save all the simulated function values.
Yname	Character string that should be used as the name of the data Y when printing or plotting the results.
envir.simul	Environment in which to evaluate the expression simulate, if not the current environment.

**Details**

This is the S3 method [envelope](#) for [Dtable](#) objects.

**Author(s)**

Eric Marcon <Eric.Marcon@agroparistech.fr>. Relies on the [envelope](#) engine of **spatstat**.

---

gEnvelope	<i>Estimation of the confidence envelope of the g function under its null hypothesis</i>
-----------	--

---

**Description**

Simulates point patterns according to the null hypothesis and returns the envelope of  $g$  according to the confidence level.

**Usage**

```
gEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05,
          ReferenceType = "", NeighborType = "", SimulationType = "RandomPosition",
          Global = FALSE, verbose = interactive())
```

**Arguments**

X	A point pattern ( <a href="#">wppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <b>spatstat</b> .
NumberOfSimulations	The number of simulations to run, 100 by default.
Alpha	The risk level, 5% by default.
ReferenceType	One of the point types. Default is all point types.
NeighborType	One of the point types. Default is all point types.
SimulationType	A string describing the null hypothesis to simulate. The null hypothesis may be " <i>RandomPosition</i> ": points are drawn in a Poisson process (default); " <i>RandomLabeling</i> ": randomizes point types, keeping locations unchanged; " <i>PopulationIndependence</i> ": keeps reference points unchanged, shifts other point locations.
Global	Logical; if TRUE, a global envelope sensu Durantou and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

**Details**

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

**Value**

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

**References**

Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.

Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.

Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.

Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

**See Also**

[ghat](#), [rRandomPositionK](#), [rRandomLocation](#), [rPopulationIndependenceK](#)

**Examples**

```
data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Calculate confidence envelope (should be 1000 simulations, reduced to 10 to save time)
r <- 0:40
NumberOfSimulations <- 10
# Plot the envelope
autoplot(gEnvelope(X, r, NumberOfSimulations))
```

---

ghat

*Estimation of the g function*


---

**Description**

Estimates the  $g$  function

**Usage**

```
ghat(X, r = NULL, ReferenceType = "", NeighborType = "", CheckArguments = TRUE)
```

## Arguments

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
ReferenceType	One of the point types. Default is all point types.
NeighborType	One of the point types. Default is all point types.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

## Details

The computation of `ghat` relies on `spatstat` function [sewpcf](#).

## Value

An object of class `fv`, see [fv.object](#), which can be plotted directly using `plot.fv`.

## References

Stoyan, D. and Stoyan, H. (1994) *Fractals, random shapes and point fields: methods of geometrical statistics*. John Wiley and Sons.

## See Also

[gEnvelope](#)

## Examples

```
data(paracou16)
autoplot(paracou16)

# Calculate g
r <- 0:30
(Paracou <- ghat(paracou16, r, "Q. Rosea", "V. Americana"))

# Plot
autoplot(Paracou)
```

---

GoFtest	<i>Goodness of Fit test between a distance based measure of spatial structure and simulations of its null hypothesis</i>
---------	--

---

**Description**

Calculates the risk to reject the null hypothesis erroneously, based on the distribution of the simulations.

**Usage**

```
GoFtest(Envelope)
```

**Arguments**

Envelope	An envelope object ( <a href="#">envelope</a> ) containing simulations in its <code>simfuns</code> attribute. It may be the result of any estimation function of the <code>dbmss</code> package or obtained by the <a href="#">envelope</a> function with argument <code>savefuns=TRUE</code> .
----------	---

**Details**

This test was introduced by Diggle(1983) and extensively developed by Loosmore and Ford (2006) for  $K$ , and applied to  $M$  by Marcon et al. (2012).

**Value**

A p-value.

**Note**

No support exists in the literature to apply the GoF test to non-cumulative functions ( $g$ ,  $Kd...$ ).

[Ktest](#) is a much better test (it does not rely on simulations) but it is limited to the  $K$  function against complete spatial randomness (CSR) in a rectangle window.

**References**

- Diggle, P. J. (1983). *Statistical analysis of spatial point patterns*. Academic Press, London. 148 p.
- Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.
- Marcon, E., F. Puech and S. Traissac (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.

**See Also**

[Ktest](#)



**Examples**

```

# Simulate a Matern (Neyman Scott) point pattern
nclust <- function(x0, y0, radius, n) {
  return(runifdisc(n, radius, centre=c(x0, y0)))
}
X <- rNeymanScott(20, 0.2, nclust, radius=0.3, n=10)
autoplot(as.wmppp(X))

# Calculate confidence envelope (should be 1000 simulations, reduced to 50 to save time)
r <- seq(0, 0.3, 0.01)
NumberOfSimulations <- 50
Alpha <- .10
Envelope <- KEnvelope(as.wmppp(X), r, NumberOfSimulations, Alpha)
autoplot(Envelope, ./(pi*r^2) ~ r)

# GoF test. Power is correct if enough simulations are run (say >1000).
paste("p-value =", GoFtest(Envelope))

```

is.wmppp

*Test whether an object is a weighted, marked, planar point pattern***Description**

Check whether its argument is an object of class "wmppp" ([wmppp.object](#)).

**Usage**

```
is.wmppp(X)
```

**Arguments**

X                   Any object

**Value**

TRUE if X is a weighted, marked, planar point pattern, otherwise FALSE.

**See Also**

[wmppp.object](#)

---

KdEnvelope	<i>Estimation of the confidence envelope of the Kd function under its null hypothesis</i>
------------	---

---

### Description

Simulates point patterns according to the null hypothesis and returns the envelope of *Kd* according to the confidence level.

### Usage

```
KdEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05, ReferenceType,
           NeighborType = ReferenceType, Weighted = FALSE, Original = TRUE,
           Approximate = ifelse(X$n < 10000, 0, 1), Adjust = 1, MaxRange = "ThirdW",
           StartFromMinR = FALSE,
           SimulationType = "RandomLocation", Global = FALSE,
           verbose = interactive())
```

### Arguments

X	A point pattern ( <a href="#">wmppp.object</a> ) or a <a href="#">Dtable</a> object.
r	A vector of distances. If NULL, a default value is set: 512 equally spaced values are used, and the first 256 are returned, corresponding to half the maximum distance between points (following Duranton and Overman, 2005).
NumberOfSimulations	The number of simulations to run, 100 by default.
Alpha	The risk level, 5% by default.
ReferenceType	One of the point types.
NeighborType	One of the point types. By default, the same as reference type.
Weighted	Logical; if TRUE, estimates the <i>Kemp</i> function.
Original	Logical; if TRUE (by default), the original bandwidth selection by Duranton and Overman (2005) following Silverman (2006: eq 3.31) is used. If FALSE, it is calculated following Sheather and Jones (1991), <i>i.e.</i> the state of the art. See <a href="#">bw.SJ</a> for more details.
Approximate	if not 0 (1 is a good choice), exact distances between pairs of points are rounded to 1024 times Approximate single values equally spaced between 0 and the largest distance. This technique (Scholl and Brenner, 2015) allows saving a lot of memory when addressing large point sets (the default value is 1 over 10000 points). Increasing Approximate allows better precision at the cost of proportional memory use. Ignored if X is a <a href="#">Dtable</a> object.
Adjust	Force the automatically selected bandwidth (following Silverman, 1986) to be multiplied by Adjust. Setting it to values lower than one (1/2 for example) will sharpen the estimation. If not 1, Original is ignored.

MaxRange	The maximum value of $r$ to consider, ignored if $r$ is not NULL. Default is "ThirdW", one third of the diameter of the window. Other choices are "HalfW", and "QuarterW" and "D02005". "HalfW", and "QuarterW" are for half or the quarter of the diameter of the window. "D02005" is for the median distance observed between points, following Duranton and Overman (2005). "ThirdW" should be close to "D02005" but has the advantage to be independent of the point types chosen as ReferenceType and NeighborType, to simplify comparisons between different types. "D02005" is approximated by "ThirdW" if Approximate is not 0. if X is a <a href="#">Dtable</a> object, the diameter of the window is taken as the max distance between points.
StartFromMinR	Logical; if TRUE, points are assumed to be further from each other than the minimum observed distance, So $Kd$ will not be estimated below it: it is assumed to be 0. If FALSE, by default, distances are smoothed down to $r = 0$ . Ignored if Approximate is not 0: then, estimation always starts from $r = 0$ .
SimulationType	A string describing the null hypothesis to simulate. The null hypothesis may be "RandomLocation": points are redistributed on the actual locations (default); "RandomLabeling": randomizes point types, keeping locations and weights unchanged; "PopulationIndependence": keeps reference points unchanged, randomizes other point locations.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

## Details

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

## Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

## References

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.
- Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.

Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

Scholl, T. and Brenner, T. (2015) Optimizing distance-based methods for large data sets, *Journal of Geographical Systems* 17(4): 333-351.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman and Hall, London.

### See Also

[Kdhat](#)

### Examples

```
data(paracou16)
autoplot(paracou16[paracou16$marks$PointType=="Q. Rosea"])

# Calculate confidence envelope
plot(KdEnvelope(paracou16, , ReferenceType="Q. Rosea", Global=TRUE))

# Center of the confidence interval
Kdhat(paracou16, ReferenceType="") -> kd
lines(kd$Kd ~ kd$r, lty=3, col="green")
```

---

Kdhat

*Estimation of the Kd function*

---

### Description

Estimates the *Kd* function

### Usage

```
Kdhat(X, r = NULL, ReferenceType, NeighborType = ReferenceType, Weighted = FALSE,
      Original = TRUE, Approximate = ifelse(X$n < 10000, 0, 1), Adjust = 1,
      MaxRange = "ThirdW", StartFromMinR = FALSE, CheckArguments = TRUE)
```

### Arguments

X	A weighted, marked planar point pattern ( <a href="#">wmppp.object</a> ) or a <a href="#">Dtable</a> object.
r	A vector of distances. If NULL, a default value is set: 512 equally spaced values are used, from the smallest distance between points to half the diameter of the window.
ReferenceType	One of the point types. If "", all points are considered (this is not the default value; NeighborType is ignored then) to estimate the average value of simulated <i>Kd</i> values under the null hypothesis of <i>RandomLocation</i> (Marcon and Puech, 2012).
NeighborType	One of the point types. By default, the same as reference type.

Weighted	Logical; if TRUE, estimates the <i>Kemp</i> function.
Original	Logical; if TRUE (by default), the original bandwidth selection by Duranton and Overman (2005) following Silverman (1986: eq 3.31) is used. If FALSE, it is calculated following Sheather and Jones (1991), <i>i.e.</i> the state of the art. See <a href="#">bw.SJ</a> for more details.
Approximate	if not 0 (1 is a good choice), exact distances between pairs of points are rounded to 1024 times Approximate single values equally spaced between 0 and the largest distance. This technique (Scholl and Brenner, 2015) allows saving a lot of memory when addressing large point sets (the default value is 1 over 10000 points). Increasing Approximate allows better precision at the cost of proportional memory use. Ignored if <i>X</i> is a <a href="#">Dtable</a> object.
Adjust	Force the automatically selected bandwidth (following Original) to be multiplied by Adjust. Setting it to values lower than one (1/2 for example) will sharpen the estimation.
MaxRange	The maximum value of <i>r</i> to consider, ignored if <i>r</i> is not NULL. Default is "ThirdW", one third of the diameter of the window. Other choices are "HalfW", and "QuarterW" and "D02005". "HalfW", and "QuarterW" are for half or the quarter of the diameter of the window. "D02005" is for the median distance observed between points, following Duranton and Overman (2005). "ThirdW" should be close to "D02005" but has the advantage to be independent of the point types chosen as ReferenceType and NeighborType, to simplify comparisons between different types. "D02005" is approximated by "ThirdW" if Approximate is not 0. if <i>X</i> is a <a href="#">Dtable</a> object, the diameter of the window is taken as the max distance between points.
StartFromMinR	Logical; if TRUE, points are assumed to be further from each other than the minimum observed distance, So <i>Kd</i> will not be estimated below it: it is assumed to be 0. If FALSE, distances are smoothed down to $r = 0$ . Ignored if Approximate is not 0: then, estimation always starts from $r = 0$ .
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

## Details

*Kd* is a density, absolute measure of a point pattern structure. *Kd* is computed efficiently by building a matrix of distances between point pairs and calculating the density of their distribution (the default values of *r* are those of the [density](#) function). The kernel estimator is Gaussian.

The weighted *Kd* function has been named *Kemp* (*emp* is for employees) by Duranton and Overman (2005).

If *X* is not a [Dtable](#) object, the maximum value of *r* is obtained from the geometry of the window rather than calculating the median distance between points as suggested by Duranton and Overman (2005) to save (a lot of) calculation time.

## Value

An object of class *fv*, see [fv.object](#), which can be plotted directly using [plot.fv](#).

**Note**

Estimating *Kd* relies on calculating distances, exactly or approximately (if `Approximate` is not 0). Then distances are smoothed by estimating their probability density. Reflection is used to estimate density close to the lowest distance, that is the minimum observed distance (if `StartFromMinR` is TRUE) or 0: all distances below 4 times the estimation kernel bandwidth apart from the lowest distance are duplicated (symmetrically with respect to the lowest distance) to avoid edge effects (underestimation of the density close to the lowest distance).

Density estimation heavily relies on the bandwidth. Starting from version 2.7, the optimal bandwidth is computed from the distribution of distances between pairs of points up to twice the maximum distance considered. The consequence is that choosing a smaller range of distances in argument `r` results in less smoothed *Kd* values. The default values (`r = NULL`, `MaxRange = "ThirdW"`) are such that almost all the pairs of points (except those more than 2/3 of the window diameter apart) are taken into account to determine the bandwidth.

**References**

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.
- Scholl, T. and Brenner, T. (2015) Optimizing distance-based methods for large data sets, *Journal of Geographical Systems* 17(4): 333-351.
- Sheather, S. J. and Jones, M. C. (1991) A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society series B*, 53, 683-690.
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman and Hall, London.

**See Also**

[KdEnvelope](#), [Mhat](#)

**Examples**

```
data(paracou16)
autoplot(paracou16)

# Calculate Kd
(Paracou <- Kdhat(paracou16, , "Q. Rosea", "V. Americana"))
# Plot
autoplot(Paracou)
```

---

KEnvelope	<i>Estimation of the confidence envelope of the K function under its null hypothesis</i>
-----------	--

---

### Description

Simulates point patterns according to the null hypothesis and returns the envelope of  $K$  according to the confidence level.

### Usage

```
KEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05,
          ReferenceType = "", NeighborType = ReferenceType,
          SimulationType = "RandomPosition", Precision = 0, Global = FALSE,
          verbose = interactive())
```

### Arguments

X	A point pattern ( <a href="#">wppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
NumberOfSimulations	The number of simulations to run, 100 by default.
Alpha	The risk level, 5% by default.
ReferenceType	One of the point types. Default is all point types.
NeighborType	One of the point types. By default, the same as reference type.
SimulationType	A string describing the null hypothesis to simulate. The null hypothesis may be " <i>RandomPosition</i> ": points are drawn in a Poisson process (default); " <i>RandomLabeling</i> ": randomizes point types, keeping locations unchanged; " <i>PopulationIndependence</i> ": keeps reference points unchanged, shifts other point locations.
Precision	Accuracy of point coordinates, measured as a part of distance unit. See <a href="#">rRandomPositionK</a> . Default is 0 for no approximation.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

### Details

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

**Value**

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

**References**

Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.

Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.

Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.

Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman and Hall, London.

**See Also**

[Khat](#), [rRandomPositionK](#), [rRandomLocation](#), [rPopulationIndependenceK](#)

**Examples**

```
data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Calculate confidence envelope (should be 1000 simulations, reduced to 20 to save time)
r <- 0:30
NumberOfSimulations <- 20
# Plot the envelope
autoplot(KEnvelope(X, r, NumberOfSimulations), ./(pi*r^2) ~ r)
```

---

Khat

*Estimation of the K function*

---

**Description**

Estimates the  $K$  function

**Usage**

```
Khat(X, r = NULL, ReferenceType = "", NeighborType = ReferenceType, CheckArguments = TRUE)
```



### Arguments

<code>X</code>	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
<code>r</code>	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <b>spatstat</b> .
<code>ReferenceType</code>	One of the point types. Default is all point types.
<code>NeighborType</code>	One of the point types. By default, the same as reference type.
<code>CheckArguments</code>	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

### Details

$K$  is a cumulative, topographic measure of a point pattern structure.

### Value

An object of class `fv`, see [fv.object](#), which can be plotted directly using `plot.fv`.

### Note

The computation of `Khat` relies on spatstat functions [Kest](#) and [Kcross](#).

### References

- Ripley, B. D. (1976). The Foundations of Stochastic Geometry. *Annals of Probability* 4(6): 995-998.
- Ripley, B. D. (1977). Modelling Spatial Patterns. *Journal of the Royal Statistical Society B* 39(2): 172-212.

### See Also

[Lhat](#), [KEnvelope](#), [Ktest](#)

### Examples

```
data(paracou16)
autoplot(paracou16)

# Calculate K
r <- 0:30
(Paracou <- Khat(paracou16, r))

# Plot (after normalization by pi.r^2)
autoplot(Paracou, ./(pi*r^2) ~ r)
```

---

KinhomEnvelope	<i>Estimation of the confidence envelope of the Kinhom function under its null hypothesis</i>
----------------	---

---

### Description

Simulates point patterns according to the null hypothesis and returns the envelope of *Kinhom* according to the confidence level.

### Usage

```
KinhomEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05,
               ReferenceType = "", lambda = NULL, SimulationType = "RandomPosition",
               Global = FALSE, verbose = interactive())
```

### Arguments

X	A point pattern ( <a href="#">wmppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
NumberOfSimulations	The number of simulations to run.
Alpha	The risk level.
ReferenceType	One of the point types. Default is all point types.
lambda	An estimation of the point pattern density, obtained by the <a href="#">density.ppp</a> function.
SimulationType	A string describing the null hypothesis to simulate. The null hypothesis, may be " <i>RandomPosition</i> ": points are drawn in an inhomogenous Poisson process (intensity is either lambda or estimated from X); " <i>RandomLocation</i> ": points are redistributed across actual locations; " <i>RandomLabeling</i> ": randomizes point types, keeping locations unchanged; " <i>PopulationIndependence</i> ": keeps reference points unchanged, redistributes others across actual locations.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

### Details

The random location null hypothesis is that of Duranton and Overman (2005). It is appropriate to test the univariate *Kinhom* function of a single point type, redistributing it over all point locations. It allows fixing lambda along simulations so the warning message can be ignored.

The random labeling hypothesis is appropriate for the bivariate *Kinhom* function.

The population independence hypothesis is that of Marcon and Puech (2010).

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

## Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

## References

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.
- Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.
- Marcon, E. and Puech, F. (2010). Measures of the Geographic Concentration of Industries: Improving Distance-Based Methods. *Journal of Economic Geography* 10(5): 745-762.
- Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

## See Also

[Kinhomhat](#)

## Examples

```
data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Density of all trees
lambda <- density.ppp(X, bw.diggle(X))
plot(lambda)
V.americana <- X[X$marks$PointType=="V. Americana"]
plot(V.americana, add=TRUE)

# Calculate Kinhom according to the density of all trees
# and confidence envelope (should be 1000 simulations, reduced to 4 to save time)
r <- 0:30
NumberOfSimulations <- 4
```

```
Alpha <- .10
autoplot(KinhomEnvelope(X, r,NumberOfSimulations, Alpha, ,
  SimulationType="RandomPosition", lambda=lambda), ./(pi*r^2) ~ r)
```

---

 Kinhomhat

*Estimation of the inhomogenous K function*


---

## Description

Estimates the *Kinhom* function

## Usage

```
Kinhomhat(X, r = NULL, ReferenceType = "", lambda = NULL, CheckArguments = TRUE)
```

## Arguments

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
ReferenceType	One of the point types. Default is all point types.
lambda	An estimation of the point pattern density, obtained by the <a href="#">density.ppp</a> function.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

## Details

*Kinhom* is a cumulative, topographic measure of an inhomogenous point pattern structure.

By default, density estimation is performed at points by [density.ppp](#) using the optimal bandwidth ([bw.diggle](#)). It can be calculated separately (see example), including at pixels if the point pattern is too large for the default estimation to succeed, and provided as the argument `lambda`: Arbia et al. (2012) for example use another point pattern as a reference to estimate density.

Bivariate *Kinhom* is not currently supported.

## Value

An object of class `fv`, see [fv.object](#), which can be plotted directly using [plot.fv](#).

## Note

The computation of `Kinhomhat` relies on `spatstat` functions [Kinhom](#), [density.ppp](#) and [bw.diggle](#).

## References

- Baddeley, A. J., J. Møller, et al. (2000). Non- and semi-parametric estimation of interaction in inhomogeneous point patterns. *Statistica Neerlandica* 54(3): 329-350.
- Arbia, G., G. Espa, et al. (2012). Clusters of firms in an inhomogeneous space: The high-tech industries in Milan. *Economic Modelling* 29(1): 3-11.

## See Also

[KinhomEnvelope](#), [Kinhom](#)

## Examples

```
data(paracou16)

# Density of all trees
lambda <- density.ppp(paracou16, bw.diggle(paracou16))
plot(lambda)
# Reduce the point pattern to one type of trees
V.americana <- paracou16[paracou16$marks$PointType=="V. Americana"]
plot(V.americana, add=TRUE)

# Calculate Kinhom according to the density of all trees
r <- 0:30
autoplot(Kinhomhat(paracou16, r, "V. Americana", lambda), ./(pi*r^2) ~ r)
```

---

KmmEnvelope

*Estimation of the confidence envelope of the Lmm function under its null hypothesis*

---

## Description

Simulates point patterns according to the null hypothesis and returns the envelope of *Lmm* according to the confidence level.

## Usage

```
KmmEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05, ReferenceType = "",
            Global = FALSE, verbose = interactive())
```

## Arguments

- X** A point pattern ([wppp](#) object).
- r** A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following [spatstat](#).
- NumberOfSimulations** The number of simulations to run, 100 by default.
- Alpha** The risk level, 5% by default.

ReferenceType	One of the point types. Others are ignored. Default is all point types.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

### Details

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

### Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

### References

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.
- Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.
- Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

### See Also

[Kmmhat](#)

### Examples

```
data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Calculate confidence envelope (should be 1000 simulations, reduced to 4 to save time)
r <- seq(0, 30, 2)
NumberOfSimulations <- 4
Alpha <- .10
autoplot(KmmEnvelope(X, r, NumberOfSimulations, Alpha), ./(pi*r^2) ~ r)
```

---

Kmmhat *Estimation of the Kmm function*

---

**Description**

Estimates of the *Kmm* function

**Usage**

```
Kmmhat(X, r = NULL, ReferenceType = "", CheckArguments = TRUE)
```

**Arguments**

**X** A weighted, marked, planar point pattern ([wmppp.object](#)).

**r** A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following **spatstat**.

**ReferenceType** One of the point types. Others are ignored. Default is all point types.

**CheckArguments** Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

The *Kmm* function is used to test the independence of marks.

**Value**

An object of class fv, see [fv.object](#), which can be plotted directly using [plot.fv](#).

**Note**

The function is computed using [markcorrint](#) in spatstat.

**References**

Penttinen, A., Stoyan, D. and Henttonen, H. M. (1992). Marked Point Processes in Forest Statistics. *Forest Science* 38(4): 806-824.

Penttinen, A. (2006). Statistics for Marked Point Patterns. in *The Yearbook of the Finnish Statistical Society*. The Finnish Statistical Society, Helsinki: 70-91.

**See Also**

[Lmmhat](#), [LmmEnvelope](#), [markcorrint](#)

**Examples**

```

data(paracou16)
# Keep only 50% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.5))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Calculate Kmm
r <- seq(0, 30, 2)
(Paracou <- Kmmhat(X, r))

# Plot
autoplot(Paracou, ./(pi*r^2) ~ r)

```

---

Ktest

*Test of a point pattern against Complete Spatial Randomness*


---

**Description**

Tests the point pattern against CSR using values of the  $K$  function

**Usage**

```
Ktest(X, r)
```

**Arguments**

X	A point pattern ( <a href="#">ppp.object</a> ). Marks are ignored. The window must be a rectangle sensu spatstat (tested by <a href="#">is.rectangle</a> ).
r	A vector of distances.

**Details**

The test returns the risk to reject CSR erroneously, i.e. the p-value of the test, based on the distribution of the  $K$  function.

If  $r$  includes 0, it will be silently removed because no neighbor point can be found at distance 0. The longer  $r$ , the more accurate the test is in theory but at the cost of computation time first, and of computation accuracy then because a matrix of size the length of  $r$  must be inverted. 10 values in  $r$  seems to be a reasonable choice.

**Value**

A p-value.

**Author(s)**

Gabriel Lang <Gabriel.Lang@agroparistech.fr>, Eric Marcon <Eric.Marcon@agroparistech.fr>



## References

- Lang, G. and Marcon, E. (2013). Testing randomness of spatial point patterns with the Ripley statistic. *ESAIM: Probability and Statistics*. 17: 767-788.
- Marcon, E., S. Traissac, and Lang, G. (2013). A Statistical Test for Ripley's Function Rejection of Poisson Null Hypothesis. *ISRN Ecology* 2013(Article ID 753475): 9.

## See Also

[Khat](#), [GoFtest](#)

## Examples

```
# Simulate a Matern (Neyman Scott) point pattern
nclust <- function(x0, y0, radius, n) {
  return(runifdisc(n, radius, centre=c(x0, y0)))
}
X <- rNeymanScott(20, 0.1, nclust, radius=0.2, n=5)
autoplot(as.wmppp(X))

# Test it
Ktest(X, r=seq(0.1, .5, .1))
```

---

LEnvelope	<i>Estimation of the confidence envelope of the L function under its null hypothesis</i>
-----------	--

---

## Description

Simulates point patterns according to the null hypothesis and returns the envelope of  $L$  according to the confidence level.

## Usage

```
LEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05,
  ReferenceType = "", NeighborType = "", SimulationType = "RandomPosition",
  Precision = 0, Global = FALSE, verbose = interactive())
```

## Arguments

X	A point pattern ( <a href="#">wmppp</a> object).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
NumberOfSimulations	The number of simulations to run, 100 by default.
Alpha	The risk level, 5% by default.
ReferenceType	One of the point types. Default is all point types.

NeighborType	One of the point types. Default is all point types.
SimulationType	A string describing the null hypothesis to simulate. The null hypothesis may be "RandomPosition": points are drawn in a Poisson process (default); "RandomLabeling": randomizes point types, keeping locations unchanged; "PopulationIndependence": keeps reference points unchanged, randomizes other point locations.
Precision	Accuracy of point coordinates, measured as a part of distance unit. See <a href="#">rRandomPositionK</a> . Default is 0 for no approximation.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

### Details

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

### Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The `fv` contains the observed value of the function, its average simulated value and the confidence envelope.

### References

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.
- Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.
- Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

### See Also

[Khat](#)

**Examples**

```

data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Calculate confidence envelope (should be 1000 simulations, reduced to 20 to save time)
r <- 0:30
NumberOfSimulations <- 20
# Plot the envelope
autoplot(LEnvelope(X, r, NumberOfSimulations))

```

Lhat

*Estimation of the L function***Description**

Estimates the  $L$  function

**Usage**

```
Lhat(X, r = NULL, ReferenceType = "", NeighborType = "", CheckArguments = TRUE)
```

**Arguments**

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <b>spatstat</b> .
ReferenceType	One of the point types. Default is all point types.
NeighborType	One of the point types. Default is all point types.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

$L$  is the normalized version of  $K$ :  $L(r) = \sqrt{\frac{K}{\pi}} - r$ .

**Value**

An object of class `fv`, see [fv.object](#), which can be plotted directly using [plot.fv](#).

**Note**

$L$  was originally defined as  $L(r) = \sqrt{\frac{K}{\pi}}$ . It has been used as  $L(r) = \sqrt{\frac{K}{\pi}} - r$  in a part of the literature because this normalization is easier to plot.

## References

Besag, J. E. (1977). Comments on Ripley's paper. *Journal of the Royal Statistical Society B* 39(2): 193-195.

## See Also

[Khat](#), [LEnvelope](#)

## Examples

```
data(paracou16)
autoplot(paracou16)

# Calculate L
r <- 0:30
(Paracou <- Lhat(paracou16, r))

# Plot
autoplot(Paracou)
```

---

LmmEnvelope	<i>Estimation of the confidence envelope of the Lmm function under its null hypothesis</i>
-------------	--

---

## Description

Simulates point patterns according to the null hypothesis and returns the envelope of *Lmm* according to the confidence level.

## Usage

```
LmmEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05, ReferenceType = "",
            Global = FALSE, verbose = interactive())
```

## Arguments

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
r	A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following <a href="#">spatstat</a> .
NumberOfSimulations	The number of simulations to run, 100 by default.
Alpha	The risk level, 5% by default.
ReferenceType	One of the point types. Others are ignored. Default is all point types.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

## Details

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

## Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

## References

Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.

Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.

Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.

Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

## See Also

[Lmmhat](#)

## Examples

```
data(paracou16)
# Keep only 20% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.2))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Calculate confidence envelope (should be 1000 simulations, reduced to 4 to save time)
r <- seq(0, 30, 2)
NumberOfSimulations <- 4
Alpha <- .10
autoplot(LmmEnvelope(X, r, NumberOfSimulations, Alpha))
```

**Description**

Estimates the *Lmm* function

**Usage**

```
Lmmhat(X, r = NULL, ReferenceType = "", CheckArguments = TRUE)
```

**Arguments**

**X** A weighted, marked, planar point pattern ([wmppp.object](#)).

**r** A vector of distances. If NULL, a sensible default value is chosen (512 intervals, from 0 to half the diameter of the window) following [spatstat](#).

**ReferenceType** One of the point types. Others are ignored. Default is all point types.

**CheckArguments** Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

*Lmm* is the normalized version of *Kmm*:  $Lmm(r) = \sqrt{\frac{Kmm}{\pi}} - r$ .

**Value**

An object of class `fv`, see [fv.object](#), which can be plotted directly using [plot.fv](#).

**References**

Penttinen, A., Stoyan, D. and Henttonen, H. M. (1992). Marked Point Processes in Forest Statistics. *Forest Science* 38(4): 806-824.

Espa, G., Giuliani, D. and Arbia, G. (2010). Weighting Ripley's K-function to account for the firm dimension in the analysis of spatial concentration. *Discussion Papers*, 12/2010. Universita di Trento, Trento: 26.

**See Also**

[Kmmhat](#), [LmmEnvelope](#)

**Examples**

```

data(paracou16)
# Keep only 50% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.5))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")

# Calculate Lmm
r <- seq(0, 30, 2)
(Paracou <- Lmmhat(X, r))

# Plot
autoplot(Paracou)

```

---

MEnvelope	<i>Estimation of the confidence envelope of the M function under its null hypothesis</i>
-----------	--

---

**Description**

Simulates point patterns according to the null hypothesis and returns the envelope of  $M$  according to the confidence level.

**Usage**

```

MEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05,
  ReferenceType, NeighborType = ReferenceType,
  CaseControl = FALSE, SimulationType = "RandomLocation", Global = FALSE,
  verbose = interactive())

```

**Arguments**

$X$	A point pattern ( <a href="#">wmppp.object</a> ) or a <a href="#">Dtable</a> object.
$r$	A vector of distances. If NULL, a default value is set: 32 unequally spaced values are used up to half the maximum distance between points $d_m$ . The first value is 0, first steps are small ( $d_m/200$ ) then increase progressively up to $d_m/20$ .
NumberOfSimulations	The number of simulations to run, 100 by default.
Alpha	The risk level, 5% by default.
ReferenceType	One of the point types.
NeighborType	One of the point types, equal to the reference type by default to calculate univariate $M$ .
CaseControl	Logical; if TRUE, the case-control version of $M$ is computed. <i>ReferenceType</i> points are cases, <i>NeighborType</i> points are controls.

SimulationType	A string describing the null hypothesis to simulate. The null hypothesis may be "RandomLocation": points are redistributed on the actual locations (default); "RandomLabeling": randomizes point types, keeping locations and weights unchanged; "PopulationIndependence": keeps reference points unchanged, randomizes other point locations.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

### Details

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

### Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

### References

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.
- Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.
- Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

### See Also

[Mhat](#)

### Examples

```
data(paracou16)
# Keep only 50% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.5))
autoplot(X,
  labelSize = expression("Basal area (" ~cm^2~ ")"),
  labelColor = "Species")
```



```
# Calculate confidence envelope (should be 1000 simulations, reduced to 4 to save time)
NumberOfSimulations <- 4
Alpha <- .10
autoplot(MEnvelope(X, , NumberOfSimulations, Alpha,
  "V. Americana", "Q. Rosea", FALSE, "RandomLabeling"))
```

---

mEnvelope	<i>Estimation of the confidence envelope of the m function under its null hypothesis</i>
-----------	--

---

## Description

Simulates point patterns according to the null hypothesis and returns the envelope of  $m$  according to the confidence level.

## Usage

```
mEnvelope(X, r = NULL, NumberOfSimulations = 100, Alpha = 0.05,
  ReferenceType, NeighborType = ReferenceType, CaseControl = FALSE,
  Original = TRUE, Approximate = ifelse(X$n < 10000, 0, 1), Adjust = 1,
  MaxRange = "ThirdW", SimulationType = "RandomLocation", Global = FALSE,
  verbose = interactive())
```

## Arguments

X	A point pattern ( <a href="#">wmppp.object</a> ).
r	A vector of distances. If NULL, a default value is set: 512 equally spaced values are used up to the median distance between points (following Duranton and Overman, 2005).
NumberOfSimulations	The number of simulations to run, 100 by default.
Alpha	The risk level, 5% by default.
ReferenceType	One of the point types.
NeighborType	One of the point types, equal to the reference type by default to calculate univariate $M$ .
CaseControl	Logical; if TRUE, the case-control version of $M$ is computed. <i>ReferenceType</i> points are cases, <i>NeighborType</i> points are controls.
Original	Logical; if TRUE (by default), the original bandwidth selection by Duranton and Overman (2005) following Silverman (1986: eq 3.31) is used. If FALSE, it is calculated following Sheather and Jones (1991), <i>i.e.</i> the state of the art. See <a href="#">bw.SJ</a> for more details.
Approximate	if not 0 (1 is a good choice), exact distances between pairs of points are rounded to 1024 times <i>Approximate</i> single values equally spaced between 0 and the largest distance. This technique (Scholl and Brenner, 2015) allows saving a lot of memory when addressing large point sets (the default value is 1 over 10000 points). Increasing <i>Approximate</i> allows better precision at the cost of proportional memory use.

Adjust	Force the automatically selected bandwidth (following Original) to be multiplied by Adjust. Setting it to values lower than one (1/2 for example) will sharpen the estimation.
MaxRange	The maximum value of r to consider, ignored if r is not NULL. Default is "ThirdW", one third of the diameter of the window. Other choices are "HalfW", and "QuarterW" and "D02005". "HalfW", and "QuarterW" are for half or the quarter of the diameter of the window. "D02005" is for the median distance observed between points, following Duranton and Overman (2005). "ThirdW" should be close to "D02005" but has the advantage to be independent of the point types chosen as ReferenceType and NeighborType, to simplify comparisons between different types. "D02005" is approximated by "ThirdW" if Approximate is not 0.
SimulationType	A string describing the null hypothesis to simulate. The null hypothesis may be "RandomLocation": points are redistributed on the actual locations (default); "RandomLabeling": randomizes point types, keeping locations and weights unchanged; "PopulationIndependence": keeps reference points unchanged, randomizes other point locations.
Global	Logical; if TRUE, a global envelope sensu Duranton and Overman (2005) is calculated.
verbose	Logical; if TRUE, print progress reports during the simulations.

### Details

This envelope is local by default, that is to say it is computed separately at each distance. See Loosmore and Ford (2006) for a discussion.

The global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

### Value

An envelope object ([envelope](#)). There are methods for print and plot for this class.

The fv contains the observed value of the function, its average simulated value and the confidence envelope.

### References

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Kenkel, N. C. (1988). Pattern of Self-Thinning in Jack Pine: Testing the Random Mortality Hypothesis. *Ecology* 69(4): 1017-1024.
- Lang G., Marcon E. and Puech F. (2014) Distance-Based Measures of Spatial Concentration: Introducing a Relative Density Function. *HAL* 01082178, 1-18.
- Loosmore, N. B. and Ford, E. D. (2006). Statistical inference using the G or K point pattern spatial statistics. *Ecology* 87(8): 1925-1931.

Marcon, E. and F. Puech (2017). A typology of distance-based measures of spatial concentration. *Regional Science and Urban Economics*. 62:56-67.

Scholl, T. and Brenner, T. (2015) Optimizing distance-based methods for large data sets, *Journal of Geographical Systems* 17(4): 333-351.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman and Hall, London.

## See Also

[mhat](#)

## Examples

```
data(paracou16)
# Keep only 50% of points to run this example
X <- as.wmppp(rthin(paracou16, 0.5))
autoplot(X,
  labelSize = expression("Basal area ( ~cm^2~ ")",
  labelColor = "Species")

# Calculate confidence envelope (should be 1000 simulations, reduced to 4 to save time)
NumberOfSimulations <- 4
Alpha <- .10
autoplot(mEnvelope(X, , NumberOfSimulations, Alpha,
  "V. Americana", "Q. Rosea", Original = FALSE, SimulationType = "RandomLabeling"))
```

---

Mhat

*Estimation of the M function*

---

## Description

Estimates the  $M$  function

## Usage

```
Mhat(X, r = NULL, ReferenceType, NeighborType = ReferenceType,
  CaseControl = FALSE, Individual = FALSE, CheckArguments = TRUE)
```

## Arguments

<code>X</code>	A weighted, marked planar point pattern ( <a href="#">wmppp.object</a> ) or a <a href="#">Dtable</a> object.
<code>r</code>	A vector of distances. If NULL, a default value is set: 64 unequally spaced values are used up to half the maximum distance between points $d_m$ . The first value is 0, first steps are small ( $d_m/800$ ) then increase progressively up to $d_m/40$ .
<code>ReferenceType</code>	One of the point types.
<code>NeighborType</code>	One of the point types. By default, the same as reference type.

CaseControl	Logical; if TRUE, the case-control version of $M$ is computed. <i>ReferenceType</i> points are cases, <i>NeighborType</i> points are controls.
Individual	Logical; if TRUE, values of the function around each individual point are returned.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

### Details

$M$  is a weighted, cumulative, relative measure of a point pattern structure. Its value at any distance is the ratio of neighbors of the *NeighborType* to all points around *ReferenceType* points, normalized by its value over the windows.

If *CaseControl* is TRUE, then *ReferenceType* points are cases and *NeighborType* points are controls. The univariate concentration of cases is calculated as if *NeighborType* was equal to *ReferenceType*, but only controls are considered when counting all points around cases (Marcon et al., 2012). This makes sense when the sampling design is such that all points of *ReferenceType* (the cases) but only a sample of the other points (the controls) are recorded. Then, the whole distribution of points is better represented by the controls alone.

### Value

An object of class `fv`, see `fv.object`, which can be plotted directly using `plot.fv`.

If *Individual* is set to TRUE, the object also contains the value of the function around each individual *ReferenceType* point taken as the only reference point. The column names of the `fv` are "M\_" followed by the point names, i.e. the row names of the marks of the point pattern.

### References

- Marcon, E. and Puech, F. (2010). Measures of the Geographic Concentration of Industries: Improving Distance-Based Methods. *Journal of Economic Geography* 10(5): 745-762.
- Marcon, E., F. Puech and S. Traissac (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.
- Marcon, E., and Puech, F. (2017). A Typology of Distance-Based Measures of Spatial Concentration. *Regional Science and Urban Economics* 62:56-67

### See Also

[MEnvelope](#), [Kdhat](#)

### Examples

```
data(paracou16)
autoplot(paracou16)

# Calculate M
autoplot(Mhat(paracou16, , "V. Americana", "Q. Rosea"))
```

---

mhat *Estimation of the m function*

---

### Description

Estimates the  $m$  function

### Usage

```
mhat(X, r = NULL, ReferenceType, NeighborType = ReferenceType,
      CaseControl = FALSE, Original = TRUE, Approximate = ifelse(X$n < 10000, 0, 1),
      Adjust = 1, MaxRange = "ThirdW", Individual = FALSE, CheckArguments = TRUE)
```

### Arguments

X	A weighted, marked planar point pattern ( <a href="#">wppp.object</a> ) or a <a href="#">Dtable</a> object.
r	A vector of distances. If NULL, a default value is set: 512 equally spaced values are used, from the smallest distance to the range defined by MaxRange. the between points to half the diameter of the window.
ReferenceType	One of the point types.
NeighborType	One of the point types. By default, the same as reference type.
CaseControl	Logical; if TRUE, the case-control version of $M$ is computed. <i>ReferenceType</i> points are cases, <i>NeighborType</i> points are controls.
Original	Logical; if TRUE (by default), the original bandwidth selection by Duranton and Overman (2005) following Silverman (1986: eq 3.31) is used. If FALSE, it is calculated following Sheather and Jones (1991), <i>i.e.</i> the state of the art. See <a href="#">bw.SJ</a> for more details.
Approximate	if not 0 (1 is a good choice), exact distances between pairs of points are rounded to 1024 times Approximate single values equally spaced between 0 and the largest distance. This technique (Scholl and Brenner, 2015) allows saving a lot of memory when addressing large point sets (the default value is 1 over 10000 points). Increasing Approximate allows better precision at the cost of proportional memory use. Ignored if X is a <a href="#">Dtable</a> object.
Adjust	Force the automatically selected bandwidth (following Original) to be multiplied by Adjust. Setting it to values lower than one (1/2 for example) will sharpen the estimation.
MaxRange	The maximum value of r to consider, ignored if r is not NULL. Default is "ThirdW", one third of the diameter of the window. Other choices are "HalfW", and "QuarterW" and "D02005". "HalfW", and "QuarterW" are for half or the quarter of the diameter of the window. "D02005" is for the median distance observed between points, following Duranton and Overman (2005). "ThirdW" should be close to "D02005" but has the advantage to be independent of the point types chosen as ReferenceType and NeighborType, to simplify comparisons between different types. "D02005" is approximated by "ThirdW" if Approximate is not 0. If X is a <a href="#">Dtable</a> object, the diameter of the window is taken as the max distance between points.

Individual	Logical; if TRUE, values of the function around each individual point are returned.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

## Details

$m$  is a weighted, density, relative measure of a point pattern structure (Lang *et al.*, 2014). Its value at any distance is the ratio of neighbors of the *NeighborType* to all points around *ReferenceType* points, normalized by its value over the windows.

The number of neighbors at each distance is estimated by a Gaussian kernel whose bandwidth is chosen optimally according to Silverman (1986: eq 3.31). It can be sharpened or smoothed by multiplying it by *Adjust*. The bandwidth of Sheather and Jones (1991) would be better but it is very slow to calculate for large point patterns and it sometimes fails. It is often sharper than that of Silverman.

If  $X$  is not a `Dtable` object, the maximum value of  $r$  is obtained from the geometry of the window rather than calculating the median distance between points as suggested by Durantou and Overman (2005) to save (a lot of) calculation time.

If *CaseControl* is TRUE, then *ReferenceType* points are cases and *NeighborType* points are controls. The univariate concentration of cases is calculated as if *NeighborType* was equal to *ReferenceType*, but only controls are considered when counting all points around cases (Marcon *et al.*, 2012). This makes sense when the sampling design is such that all points of *ReferenceType* (the cases) but only a sample of the other points (the controls) are recorded. Then, the whole distribution of points is better represented by the controls alone.

## Value

An object of class `fv`, see [fv.object](#), which can be plotted directly using `plot.fv`.

If *Individual* is set to TRUE, the object also contains the value of the function around each individual *ReferenceType* point taken as the only reference point. The column names of the `fv` are "m\_" followed by the point names, i.e. the row names of the marks of the point pattern.

## Note

Estimating  $m$  relies on calculating distances, exactly or approximately (if *Approximate* is not 0). Then distances are smoothed by estimating their probability density. In contrast with `Kdhat`, reflection is not used to estimate density close to the lowest distance. The same kernel estimation is applied to the distances from reference points of neighbor points and of all points. Since  $m$  is a relative function, a ratio of densities is calculated, that makes the features of the estimation vanish.

Density estimation heavily relies on the bandwidth. Starting from version 2.7, the optimal bandwidth is computed from the distribution of distances between pairs of points up to twice the maximum distance considered. The consequence is that choosing a smaller range of distances in argument  $r$  results in less smoothed  $m$  values. The default values ( $r = \text{NULL}$ , *MaxRange* = "ThirdW") are such that almost all the pairs of points (except those more than 2/3 of the window diameter apart) are taken into account to determine the bandwidth.

## References

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Lang G., Marcon E. and Puech F. (2014) Distance-Based Measures of Spatial Concentration: Introducing a Relative Density Function. *HAL* 01082178, 1-18.
- Marcon, E., F. Puech and S. Traissac (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.
- Scholl, T. and Brenner, T. (2015) Optimizing distance-based methods for large data sets, *Journal of Geographical Systems* 17(4): 333-351.
- Sheather, S. J. and Jones, M. C. (1991) A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society series B*, 53, 683-690.
- Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Chapman and Hall, London.

## See Also

[mEnvelope](#), [Kdhat](#)

## Examples

```
data(paracou16)
autoplot(paracou16)

# Calculate M
autoplot(mhat(paracou16, , "V. Americana", "Q. Rosea"))
```

---

paracou16

*Paracou field station plot 16, partial map*

---

## Description

This point pattern is from Paracou field station, French Guiana, managed by [Cirad](#).

## Usage

```
data(paracou16)
```

## Format

An object of class `ppp.object` representing the point pattern of tree locations in a 250 x 300 meter sampling region. Each tree is marked with its species ("Q. Rosea", "V. Americana" or "Other"), and basal area (square centimeters).

## Source

Permanent data census of Paracou and Marcon et al. (2012).

## References

Gourlet-Fleury, S., Guehl, J. M. and Laroussinie, O., Eds. (2004). *Ecology & management of a neotropical rainforest. Lessons drawn from Paracou, a long-term experimental research site in French Guiana*. Paris, Elsevier.

Marcon, E., F. Puech and S. Traissac (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.

## Examples

```
data(paracou16)
# Plot (second column of marks is Point Types)
autoplot(paracou16, which.marks=2, leg.side="right")
```

---

```
print.dbmssEnvelope    Print a confidence envelope
```

---

## Description

Prints useful information of a confidence envelope of class "dbmssEnvelope"

## Usage

```
## S3 method for class 'dbmssEnvelope'
print(x, ...)
```

## Arguments

x	An object of class "dbmssEnvelope".
...	Ignored.

## Details

"dbmssEnvelope" objects are similar to [envelope](#) objects. The way they are printed is different to take into account the possibility of building global envelope following Duranton and Overman (2005): the global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

## References

Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106



**Examples**

```

data(paracou16)
autoplot(paracou16)

# Calculate intertype K envelope
Envelope <- KEnvelope(paracou16, NumberOfSimulations = 20, Global = TRUE,
  ReferenceType = "V. Americana", NeighborType = "Q. Rosea")
autoplot(Envelope)
# print
print(Envelope)

```

---

rPopulationIndependenceK

*Simulations of a point pattern according to the null hypothesis of population independence defined for K*

---

**Description**

Simulates of a point pattern according to the null hypothesis of population independence defined for  $K$ .

**Usage**

```
rPopulationIndependenceK(X, ReferenceType, NeighborType, CheckArguments = TRUE)
```

**Arguments**

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
ReferenceType	One of the point types.
NeighborType	One of the point types.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

Reference points are kept unchanged, neighbor type point positions are shifted by `rshift`. Other points are lost and point weights are not kept (they are set to 1) since the  $K$  function ignores them.

**Value**

A new weighted, marked, planar point pattern (an object of class `wmppp`, see [wmppp.object](#)).

**References**

Goreaud, F. et Pelissier, R. (2003). Avoiding misinterpretation of biotic interactions with the inter-type  $K_{12}$  fonction: population independence vs random labelling hypotheses. *Journal of Vegetation Science* 14(5): 681-692.

**See Also**

[rPopulationIndependenceM](#), [rRandomLabeling](#)

**Examples**

```
# Simulate a point pattern with three types
X <- rpoispp(50)
PointType <- sample(c("A", "B", "C"), X$n, replace=TRUE)
PointWeight <- runif(X$n, min=1, max=10)
X$marks <- data.frame(PointType, PointWeight)
X <- as.wmppp(X)

# Plot the point pattern, using PointType as marks
autoplot(X, main="Original pattern")

# Randomize it
Y <- rPopulationIndependenceK(X, "A", "B")
# Points of type "A" are unchanged, points of type "B" have been moved altogether
# Other points are lost and point weights are set to 1
autoplot(Y, main="Randomized pattern")
```

---

rPopulationIndependenceM

*Simulations of a point pattern according to the null hypothesis of population independence defined for  $M$*

---

**Description**

Simulates of a point pattern according to the null hypothesis of population independence defined for  $M$

**Usage**

```
rPopulationIndependenceM(X, ReferenceType, CheckArguments = TRUE)
```

**Arguments**

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
ReferenceType	One of the point types.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

Reference points are kept unchanged, other points are redistributed randomly across locations.

**Value**

A new weighted, marked, planar point pattern (an object of class `wmppp`, see [wmppp.object](#)).

**References**

Marcon, E. and Puech, F. (2010). Measures of the Geographic Concentration of Industries: Improving Distance-Based Methods. *Journal of Economic Geography* 10(5): 745-762.

Marcon, E., F. Puech and S. Traissac (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.

**See Also**

[rPopulationIndependenceK](#), [rRandomLabelingM](#)

**Examples**

```
# Simulate a point pattern with five types
X <- rpoispp(50)
PointType <- sample(c("A", "B", "C", "D", "E"), X$n, replace=TRUE)
PointWeight <- runif(X$n, min=1, max=10)
X$marks <- data.frame(PointType, PointWeight)
X <- as.wmppp(X)
```

```
autoplot(X, main="Original pattern")
```

```
# Randomize it
Y <- rPopulationIndependenceM(X, "A")
# Points of type "A" are unchanged,
# all other points have been redistributed randomly across locations
autoplot(Y, main="Randomized pattern")
```

---

rRandomLabeling	<i>Simulations of a point pattern according to the null hypothesis of random labeling</i>
-----------------	---

---

**Description**

Simulates of a point pattern according to the null hypothesis of random labeling.

**Usage**

```
rRandomLabeling(X, CheckArguments = TRUE)
```

**Arguments**

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

Marks are redistributed randomly across the original point pattern.

**Value**

A new weighted, marked, planar point pattern (an object of class `wmppp`, see [wmppp.object](#)).

**References**

Goreaud, F. et Pelissier, R. (2003). Avoiding misinterpretation of biotic interactions with the inter-type K12 fonction: population independence vs random labelling hypotheses. *Journal of Vegetation Science* 14(5): 681-692.

**See Also**

[rRandomLabelingM](#), [rPopulationIndependenceK](#)

**Examples**

```
# Simulate a point pattern with five types
X <- rpoispp(50)
PointType <- sample(c("A", "B", "C", "D", "E"), X$n, replace=TRUE)
PointWeight <- runif(X$n, min=1, max=10)
X$marks <- data.frame(PointType, PointWeight)
X <- as.wmppp(X)

autoplot(X, main="Original pattern")

# Randomize it
Y <- rRandomLabeling(X)
# Types and weights have been redistributed randomly across locations
autoplot(Y, main="Randomized pattern")
```

---

rRandomLabelingM	<i>Simulations of a point pattern according to the null hypothesis of random labelling defined for M</i>
------------------	--

---

**Description**

Simulates of a point pattern according to the null hypothesis of random labelling defined for  $M$

**Usage**

```
rRandomLabelingM(X, CheckArguments = TRUE)
```

**Arguments**

- X** A weighted, marked, planar point pattern ([wmppp.object](#)) or a [Dtable](#) object.
- CheckArguments** Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

Point types are randomized. Locations and weights are kept unchanged. If both types and weights must be randomized together (Duranton and Overman, 2005; Marcon and Puech, 2010), use [rRandomLocation](#).

**Value**

A new weighted, marked, planar point pattern (an object of class `wmppp`, see [wmppp.object](#)).

**References**

- Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.
- Marcon, E. and Puech, F. (2010). Measures of the Geographic Concentration of Industries: Improving Distance-Based Methods. *Journal of Economic Geography* 10(5): 745-762.
- Marcon, E., F. Puech and S. Traissac (2012). Characterizing the relative spatial structure of point patterns. *International Journal of Ecology* 2012(Article ID 619281): 11.

**See Also**

[rRandomLabeling](#), [rPopulationIndependenceM](#)

**Examples**

```
# Simulate a point pattern with five types
X <- rpoispp(50)
PointType <- sample(c("A", "B", "C", "D", "E"), X$n, replace=TRUE)
PointWeight <- runif(X$n, min=1, max=10)
X$marks <- data.frame(PointType, PointWeight)
X <- as.wmppp(X)

autoplot(X, main="Original pattern")

# Randomize it
Y <- rRandomLabelingM(X)
# Labels have been redistributed randomly across locations
# But weights are unchanged
autoplot(Y, main="Randomized pattern")
```

---

rRandomLocation	<i>Simulations of a point pattern according to the null hypothesis of random location</i>
-----------------	---

---

### Description

Simulates of a point pattern according to the null hypothesis of random location.

### Usage

```
rRandomLocation(X, ReferenceType = "", CheckArguments = TRUE)
```

### Arguments

X	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
ReferenceType	One of the point types.
CheckArguments	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

### Details

Points are redistributed randomly across the locations of the original point pattern. This randomization is equivalent to random labeling, considering the label is both point type and point weight. If ReferenceType is specified, then only reference type points are kept in the original point pattern before randomization.

### Value

A new weighted, marked, planar point pattern (an object of class wmppp, see [wmppp.object](#)).

### References

Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106.

Marcon, E. and Puech, F. (2010). Measures of the Geographic Concentration of Industries: Improving Distance-Based Methods. *Journal of Economic Geography* 10(5): 745-762.

### See Also

[rRandomPositionK](#)

**Examples**

```
# Simulate a point pattern with five types
X <- rpoispp(50)
PointType <- sample(c("A", "B", "C", "D", "E"), X$n, replace=TRUE)
PointWeight <- runif(X$n, min=1, max=10)
X$marks <- data.frame(PointType, PointWeight)
X <- as.wmppp(X)

autoplot(X, main="Original pattern")

# Randomize it
Y <- rRandomLocation(X)
# Points have been redistributed randomly across locations
autoplot(Y, main="Randomized pattern")
```

---

rRandomPositionK	<i>Simulations of a point pattern according to the null hypothesis of random position defined for <math>K</math>.</i>
------------------	---

---

**Description**

Simulations of a point pattern according to the null hypothesis of random position defined for  $K$ .

**Usage**

```
rRandomPositionK(X, Precision = 0, CheckArguments = TRUE)
```

**Arguments**

<code>X</code>	A weighted, marked, planar point pattern ( <a href="#">wmppp.object</a> ).
<code>Precision</code>	Accuracy of point coordinates, measured as a part of distance unit. See notes. Default is 0 for no approximation.
<code>CheckArguments</code>	Logical; if TRUE, the function arguments are verified. Should be set to FALSE to save time in simulations for example, when the arguments have been checked elsewhere.

**Details**

Points marks are kept unchanged and their position is drawn in a binomial process by [runifpoint](#).

**Value**

A new weighted, marked, planar point pattern (an object of class `wmppp`, see [wmppp.object](#)).

**Note**

Simulations in a binomial process keeps the same number of points, so that marks can be redistributed. If a real CSR simulation is needed and marks are useless, use [rpoispp](#).

Actual data coordinates are often rounded. Use the Precision argument to simulate point patterns with the same rounding procedure. For example, if point coordinates are in meters and rounded to the nearest half meter, use Precision = 0.5 so that the same approximation is applied to the simulated point patterns.

**See Also**

[rRandomLocation](#)

**Examples**

```
# Simulate a point pattern with two types
X <- rpoispp(5)
PointType <- sample(c("A", "B"), X$n, replace=TRUE)
PointWeight <- runif(X$n, min=1, max=10)
X$marks <- data.frame(PointType, PointWeight)
X <- as.wmppp(X)

autoplot(X, main="Original pattern")

# Randomize it
Y <- rRandomPositionK(X)
# Points are randomly distributed
autoplot(Y, main="Randomized pattern")
```

---

Smooth.wmppp

*Spatial smoothing of individual dbmss's*

---

**Description**

Performs spatial smoothing of the individual values of distance-based measures computed in the neighborhood of each point (Marcon and Puech, 2023).

**Usage**

```
## S3 method for class 'wmppp'
Smooth(X, fwind, distance = NULL, Quantiles = FALSE,
       sigma = bw.scott(X, isotropic = TRUE), Weighted = TRUE, Adjust = 1,
       Nbx = 128, Nby = 128, ..., CheckArguments = TRUE)
```



**Arguments**

X	A point pattern ( <code>wmppp.object</code> ).
fvind	An object of class <code>fv</code> , see <code>fv.object</code> , obtained a distance-based method, such as <code>Mhat</code> with individual values (argument <code>Individual = TRUE</code> ).
distance	The distance at which the function value must be considered. The default value is the median distance used to calculate the function values.
Quantiles	If <code>FALSE</code> (default), the dbmss is smoothed to produce a map of the measure. If <code>TRUE</code> , its quantiles (computed by <code>Mhat</code> with argument <code>Quantiles = TRUE</code> ) are smoothed to produce a map of the confidence level of the measure.
Weighted	If <code>TRUE</code> (default), the point weights are taken into account for smoothing.
sigma	The bandwidth used for smoothing. A Gaussian kernel is used (see <code>Smooth.ppp</code> ). Its bandwidth is chosen by default according to Scott's rule (see <code>bw.scott</code> ).
Adjust	Force the selected bandwidth ( <code>sigma</code> ) to be multiplied by <code>Adjust</code> . Setting it to values smaller than one (1/2 for example) will sharpen the estimation.
Nbx, Nby	The number of columns and rows (pixels) of the resulting map, 128 by default. Increase it for quality, paid by increasing computing time.
...	Extra arguments, passed to <code>Smooth.ppp</code> .
CheckArguments	If <code>TRUE</code> (default), the function arguments are verified. Should be set to <code>FALSE</code> to save time in simulations for example, when the arguments have been checked elsewhere.

**Value**

An image that can be plotted. If quantiles have been computed in `fvind`, attributes "High" and "Low" contain logical vectors to identify significantly high and low quantiles.

**References**

Marcon, E. and Puech, F. (2023). Mapping distributions in non-homogeneous space with distance-based methods. *Journal of Spatial Econometrics* 4(1), 13.

**Examples**

```
ReferenceType <- "V. Americana"
NeighborType <- "Q. Rosea"
# Calculate individual intertype M(distance) values
fvind <- Mhat(paracou16, r=c(0, 30), ReferenceType, NeighborType, Individual=TRUE)
# Plot the point pattern with values of M(30 meters)
p16_map <- Smooth(paracou16, fvind, distance=30)
plot(p16_map, main = "")
# Add the reference points to the plot
is.ReferenceType <- paracou16$marks$PointType == ReferenceType
points(x=paracou16$x[is.ReferenceType], y=paracou16$y[is.ReferenceType], pch=20)
# Add contour lines
contour(p16_map, nlevels = 5, add = TRUE)
```

---

 spatstat generic functions

*Methods for weighted, marked planar point patterns (of class wmppp)  
from spatstat*

---

## Description

**spatstat** methods for a [ppp.object](#) applied to a [wmppp.object](#).

## Usage

```
## S3 method for class 'wmppp'
sharpen(X, ...)
## S3 method for class 'wmppp'
split(...)
## S3 method for class 'wmppp'
superimpose(...)
## S3 method for class 'wmppp'
unique(x, ...)
## S3 method for class 'wmppp'
i[j, drop=FALSE, ..., clip=FALSE]
```

## Arguments

<code>X, x</code>	A two-dimensional point pattern. An object of class "wmppp".
<code>...</code>	Arguments passed to the <a href="#">ppp.object</a> method.
<code>i</code>	Subset index. Either a valid subset index in the usual R sense, indicating which points should be retained, or a window (an object of class "owin") delineating a subset of the original observation window, or a pixel image with logical values defining a subset of the original observation window.
<code>j</code>	Redundant. Included for backward compatibility.
<code>drop</code>	Logical value indicating whether to remove unused levels of the marks, if the marks are a factor.
<code>clip</code>	Logical value indicating how to form the window of the resulting point pattern, when <code>i</code> is a window. If <code>clip=FALSE</code> (the default), the result has window equal to <code>i</code> . If <code>clip=TRUE</code> , the resulting window is the intersection between the window of <code>x</code> and the window <code>i</code> .

## Details

**spatstat** methods for ppp objects returning a ppp object can be applied to a wmppp and return a wpppp with these methods which just call the [ppp.object](#) method and change the class of the result for convenience.

Some **spatstat** functions such as [rthin](#) are not generic so they always return a [ppp.object](#) when applied to a [wmppp.object](#). Their result may be converted by [as.wmppp](#).

**Value**

An object of class "wmppp".

**See Also**

[sharpen.ppp](#), [split.ppp](#), [superimpose.ppp](#), [unique.ppp](#)

---

summary.dbmssEnvelope *Summary of a confidence envelope*

---

**Description**

Prints a useful summary of a confidence envelope of class "dbmssEnvelope"

**Usage**

```
## S3 method for class 'dbmssEnvelope'  
summary(object, ...)
```

**Arguments**

object	An object of class "dbmssEnvelope".
...	Ignored.

**Details**

"dbmssEnvelope" objects are similar to [envelope](#) objects. Their summary is different to take into account the possibility of building global envelope following Duranton and Overman (2005): the global envelope is calculated by iteration: the simulations reaching one of the upper or lower values at any distance are eliminated at each step. The process is repeated until *Alpha / Number of simulations* simulations are dropped. The remaining upper and lower bounds at all distances constitute the global envelope. Interpolation is used if the exact ratio cannot be reached.

**References**

Duranton, G. and Overman, H. G. (2005). Testing for Localisation Using Micro-Geographic Data. *Review of Economic Studies* 72(4): 1077-1106

**Examples**

```
data(paracou16)  
autoplot(paracou16)  
  
# Calculate intertype K envelope  
Envelope <- KEnvelope(paracou16, NumberOfSimulations = 20, Global = TRUE,  
  ReferenceType = "V. Americana", NeighborType = "Q. Rosea")  
autoplot(Envelope)  
summary(Envelope)
```

---

`wmppp`*Create a Weighted, Marked, Planar Point Pattern*

---

## Description

Creates an object of class "wmppp" representing a two-dimensional point pattern with weights and labels.

## Usage

```
wmppp(df, window = NULL, unitname = NULL)
```

## Arguments

<code>df</code>	A dataframe with at least two columns containing point coordinates.
<code>window</code>	An object of class "owin" ( <a href="#">owin.object</a> ).
<code>unitname</code>	Name of unit of length. Either a single character string, or a vector of two character strings giving the singular and plural forms, respectively. Ignored if <code>window</code> is not NULL.

## Details

Columns named "X", "Y", "PointType", "PointWeight" (capitalization is ignored) are searched to build the "wmppp" object and set the point coordinates, type and weight. If they are not found, columns are used in this order. If columns are missing, PointType is set to "All" and PointWeight to 1. If a "PointName" column is found, it is used to set the row names of the marks, else the original row names are used.

If the window is not specified, a rectangle containing all points is used, and `unitname` is used.

## Value

An object of class "wmppp".

## See Also

[wmppp.object](#),

## Examples

```
# Draw the coordinates of 10 points
X <- runif(10)
Y <- runif(10)
# Draw the point types.
PointType <- sample(c("A", "B"), 10, replace=TRUE)
# Plot the point pattern. Weights are set to 1 and the window is adjusted.
plot(wmppp(data.frame(X, Y, PointType)), , which.marks=2)
```

---

`wmppp.object`*Class of Weighted, Marked, Planar Point Patterns*

---

## Description

A class "wmppp" to represent a two-dimensional point pattern of class `ppp` whose marks are a dataframe with two columns:

- `PointType`: labels, as factors
- `PointWeight`: weights.

## Details

This class represents a two-dimensional point pattern dataset. `wmppp` objects are also of class `ppp`.

Objects of class `wmppp` may be created by the function `wmppp` and converted from other types of data by the function `as.wmppp`.

## See Also

[ppp.object](#), [wmppp](#), [as.wmppp](#) [autoplot.wmppp](#)

## Examples

```
# Draw the coordinates of 10 points
X <- runif(10)
Y <- runif(10)
# Draw the point types and weights
PointType <- sample(c("A", "B"), 10, replace=TRUE)
PointWeight <- runif(10)
# Build the point pattern
X <- wmppp(data.frame(X, Y, PointType, PointWeight), owin())

# Plot the point pattern. which.marks=1 for point weights, 2 for point types
par(mfrow=c(1,2))
plot(X, which.marks=1, main="Point weights")
plot(X, which.marks=2, main="Point types")

# Or use autoplot for a ggplot
autoplot(X)
```

# Index

- \* **datasets**
  - paracou16, 47
- \* **package**
  - dbmss-package, 3
- [.wmppp (spatstat generic functions), 58
  
- as.Dtable, 3, 11
- as.wmppp, 4, 4, 58, 61
- autoplot, 5, 5
- autoplot.wmppp, 61
  
- bw.diggle, 28
- bw.scott, 57
- bw.SJ, 18, 21, 41, 45
  
- data.frame, 4, 5
- dbmss (dbmss-package), 3
- dbmss-package, 3
- dbmssEnvelope.object, 7
- density, 21
- density.ppp, 26, 28
- DEnvelope, 8, 10
- Dhat, 9, 9
- Dtable, 11, 12, 18–21, 39, 43, 45, 46, 53
  
- envelope, 7, 8, 12–14, 16, 19, 24, 27, 30, 34, 37, 40, 42, 48, 59
- envelope.Dtable, 12
  
- fv.object, 10, 15, 21, 25, 28, 31, 35, 38, 44, 46, 57
  
- gEnvelope, 13, 15
- ggplot, 6
- ghat, 14, 14
- GoFtest, 16, 33
  
- is.rectangle, 32
- is.wmppp, 17
  
- Kcross, 10, 25
  
- KdEnvelope, 7, 18, 22
- Kdhat, 20, 20, 44, 46, 47
- KEnvelope, 23, 25
- Kest, 10, 25
- Khat, 10, 24, 24, 33, 34, 36
- Kinhom, 28, 29
- KinhomEnvelope, 26, 29
- Kinhomhat, 27, 28
- KmmEnvelope, 29
- Kmmhat, 30, 31, 38
- Ktest, 16, 25, 32
  
- LEnvelope, 33, 36
- Lhat, 25, 35
- LmmEnvelope, 31, 36, 38
- Lmmhat, 31, 37, 38
  
- markcorrint, 31
- MEnvelope, 7, 39, 44
- mEnvelope, 41, 47
- Mhat, 22, 40, 43, 57
- mhat, 43, 45
  
- owin.object, 4, 6, 60
  
- paracou16, 47
- plot.fv, 6, 10, 15, 21, 25, 28, 31, 35, 38, 44, 46
- ppp, 4, 5, 61
- ppp.object, 5, 32, 47, 58, 61
- print.dbmssEnvelope, 48
  
- rpoispp, 56
- rPopulationIndependenceK, 14, 24, 49, 51, 52
- rPopulationIndependenceM, 50, 50, 53
- rRandomLabeling, 50, 51, 53
- rRandomLabelingM, 51, 52, 52
- rRandomLocation, 14, 24, 53, 54, 56
- rRandomPositionK, 14, 23, 24, 34, 54, 55
- rshift, 49

rthin, [58](#)  
runifpoint, [55](#)

scale\_colour\_brewer, [6](#)  
sewpcf, [15](#)  
sharpen.ppp, [59](#)  
sharpen.wmppp (spatstat generic functions), [58](#)  
Smooth.ppp, [57](#)  
Smooth.wmppp, [56](#)  
spatstat generic functions, [58](#)  
split.ppp, [59](#)  
split.wmppp (spatstat generic functions), [58](#)  
summary.dbmssEnvelope, [7](#), [59](#)  
superimpose.ppp, [59](#)  
superimpose.wmppp (spatstat generic functions), [58](#)

unique.ppp, [59](#)  
unique.wmppp (spatstat generic functions), [58](#)

wmppp, [5](#), [60](#), [61](#)  
wmppp.object, [4](#), [5](#), [8](#), [10](#), [13](#), [15](#), [17](#), [18](#), [20](#),  
[23](#), [25](#), [26](#), [28](#), [29](#), [31](#), [33](#), [35](#), [36](#), [38](#),  
[39](#), [41](#), [43](#), [45](#), [49–55](#), [57](#), [58](#), [60](#), [61](#)