

# Package ‘MarkedPointProcess’

November 1, 2009

**Version** 0.2.13

**Title** Analysis of Marks of Marked Point Processes

**Author** Martin Schlather <martin.schlather@math.uni-goettingen.de>

**Maintainer** Martin Schlather <martin.schlather@math.uni-goettingen.de>

**Depends** R (>= 2.6.0), RandomFields (>= 1.3.30)

**Description** Non-parametric Analysis of the Marks of Marked Point Processes

**License** GPL

**URL** <http://www.stochastik.math.uni-goettingen.de/institute>

**Repository** CRAN

**Date/Publication** 2009-11-01 17:32:56

## R topics documented:

BITOEKforests . . . . .	2
get.mpp.names . . . . .	3
MarkedPointProcess . . . . .	4
mpp.characteristics . . . . .	5
rfm.test . . . . .	9
simulateMPP . . . . .	14
splitmodel . . . . .	15
srd . . . . .	17

<b>Index</b>	<b>19</b>
--------------	-----------

---

`BITOEKforests`*Forestry areas of intensif measurement by BITOEK*

---

## Description

Two forestry areas of intensive measurement maintained by BITOEK

## Usage

```
data(BITOEK)
```

## Format

The two variables, `steigerwald` and `coulissenhieb` are lists, where the first component, `coord`, is a matrix of two columns giving the positions of the trees, and the second component gives the stem diameters. The third component, `author`, gives a short description of the data set.

## Details

Both forests can be found in the the North of Bavaria, Germany; ‘Steigerwald’ is a forest near Bamberg, ‘Coulissenhieb’ is in the ‘Fichtelgebirge’.

## Source

The data were collected by Pedro Gerstberger, BITOEK, <http://www.bitoeck.uni-bayreuth.de>, University of Bayreuth, Germany.

## References

Schlather, M., Ribeiro, P. and Diggle, P. (2004) Detecting Dependence Between Marks and Locations of Marked Point Processes *J. R. Statist. Soc., Ser. B* **66**, 79-83.

## Examples

```
data(BITOEK)

plotWithCircles(cbind(coulissenhieb$coord, coulissenhieb$diam), factor=3)

plotWithCircles(cbind(steigerwald$coord, steigerwald$diam), factor=2)
```

---

get.mpp.names      *Model names for marked point process*

---

## Description

get.mpp.names returns the names of implemented marked point processes

## Usage

```
get.mpp.names()
```

## Details

currently implemented models are

- nearest neighbour  
the points are given by a stationary Poisson point process, and a mark is the distance to the nearest neighbour within the process; nearest neighbour has one parameters, multiplied to the calculated distance.
- random coins  
We start in  $R^d$ , here  $d = 2$ , with a marked Poisson process where the points are given by a stationary Poisson point process  $\Phi$  and the marks are i.i.d. random objects (disks or cones) of dimension  $d + 1$ . At a point  $x$  of  $\Phi$  the sum of the heights of the objects that cover  $x$  is the mark of this model; random coins has three parameters, the first parameter chooses the kind of coin function, the second the scale, the third the height. Currently the available coin functions are disk (1) and cone (2).
- variance by coins  
The marks are independent Gaussian random variables with mean 0 and standard deviation equal to the mark of random coin model.

## Value

get.mpp.names returns a vector of names.

## Author(s)

Martin Schlather, (martin.schlather@math.uni-goettingen.de) <http://www.stochastik.math.uni-goettingen.de/institute>

## See Also

[simulateMPP](#)

---

MarkedPointProcess *Simulation study for marked point processes*

---

## Description

This package allows for simulating and analysing marked point processes

## Details

The following functionalities are provided:

- `get.mpp.names` : returns the names of the implemented models
- `mpp.characteristics` : returns characteristics for the marks of marked point processes such as the mark variogram, Stoyan's kmm function, the E function and the V function
- `rfm.test` : MC test whether E or V is a constant. If any of these hypotheses are rejected, the investigated marked point process cannot be considered as random field model, i.e. a model where the marks are independent of the locations (however the random field model allows that the marks themselves are spatially dependent)
- `simulateMPP` : simulation of marked point processes
- `srd.jrssb` : function that generates the results published by Schlather, Ribeiro, Diggle (2004)
- `splitmodel` : auxilliary function that splits a user defined model in a pure Gaussian random field part and a pure marked point process part

Further, a forestry data set is provided, see [BITOEK](#).

## Acknowledgement

The work has been financially supported by the German Federal Ministry of Research and Technology (BMFT) grant PT BEO 51-0339476C during 2000-03.

## Author(s)

Martin Schlather, [martin.schlather@math.uni-goettingen.de](mailto:martin.schlather@math.uni-goettingen.de) <http://www.stochastik.math.uni-goettingen.de/institute>

## References

Schlather, M., Ribeiro, P. and Diggle, P. (2004) Detecting Dependence Between Marks and Locations of Marked Point Processes *J. R. Statist. Soc., Ser. B* **66**, 79-83.

---

 mpp.characteristics

*Characteristics of the marks of a marked point process*


---

## Description

mpp.characteristics returns summary statistics for the marks of a marked point process

## Usage

```
mpp.characteristics(..., bin=NULL, rep=1, p=0.8, name="", normalize=TRUE,
  show=FALSE, model=NULL, param=NULL, summarize=TRUE,
  PrintLevel=RFparameters()$Print,
  dev=if (name=="") 2 else FALSE,
  rdline=if (is.logical(dev)) NULL else readline,
  staticchoice=FALSE)
```

## Arguments

...	coordinates and the data for the species, see <a href="#">Details</a>
bin	sequence of increasing bin margins for the functions $E$ , $V$ , and $S$ ; see <a href="#">Details</a>
rep	number of independent measurements of the marks (at each point); usually 1
p	in $[0, 1]$ ; outlier threshold for the robustified distance function of the test statistics
name	character; if show=FALSE this parameter is ignored. Otherwise, if name="" then plots are printed on the screen else name is the main name of the postscript files
normalize	logical; if TRUE the data are (marginally) transformed to Gaussian variables (for each species, each kind of mark, and each realisation, seperately) before being analysed
show	logical; if TRUE the results are also shown graphically
model	the variogram model to compare with the mark variogram; see <a href="#">CovarianceFct</a>
param	the parameters for the variogram model; see <a href="#">CovarianceFct</a>
summarize	logical; if FALSE results are for each realisation seperately (instead of being averaged over the realisations).
PrintLevel	0,1 or 2. The function gives some short messages if PrintLevel is 1 or 2.
rdline	NULL or function. if not NULL then the function is called after each plot with a string parameter that gives file or image information
dev	the graphical device for the output, see <a href="#">Dev</a>
staticchoice	logical. The calculation of some variances requires the splitting of the data into group. If staticchoice=FALSE this is done in a random way. staticchoice=TRUE is only used for internal testing.

## Details

`bin`: analogously to the variogram in geostatistics, the characteristics  $E$ ,  $V$ , and  $S$  of a stationary and isotropic marked point process depend on the distance  $r$ . Instead of returning a cloud of values, binned values are calculated in the same way the binned variogram is calculated. `bin` gives the margins of the bins (left open, right closed ones) as an increasing sequence. The first bin must include the zero, i.e., `bin=c(-1, 0, ...)`.

... : data for mark point processes typically split up into different species (ill/healthy cells; beaches/oaks/pines); furthermore, multivariate data are measured for each individual (size of the cell; diameter of the stem, height of the tree). The function calculates many cross-statistics; for example the cross variogram of mark  $A$  of species  $B$  and mark  $C$  of species  $D$  given species  $B$  and species  $D$  are a distance  $r$  apart.

Denote by  $S_i$  species  $i$ ,  $i = 1, \dots, s$ . Due to the potential complexity of the data, the data are passed to `mpp.characteristics` in the following way:

(coordinates of species  $S_i$ ), (marks of species  $A$ ), ..., (coordinates of species  $Z$ ), (marks of species  $Z$ )

The coordinates are  $(n_i \times 2)$  matrices; the data are  $(n_i \times m \times \text{rep})$  matrices, if the data are  $m$ -variate, and `rep` independent observations of the data exist. In case `rep > 1` the sequence for the data is:

columns 1:m : first set of the  $m$ -variate data, ...,

columns (m \* rep - m + 1) : (m \* rep) : last set of the  $m$ -variate data.

Note that  $m$  and `rep` must be identical for all species.

The function returns the following values if `summarize=TRUE`. Denote by  $M_k(S_i)$  the  $k$ th mark of species  $S_i$ , and by  $d_{ij}$  the distance of two individuals of species  $S_i$  and  $S_j$ . Denote by  $s$  the number of species.

- $E$ :  $(\text{length}(\text{bin}) - 1) \times (ms^2)$  matrix. function  $E$  in the following ordering of the columns. The index for the marks runs fastest, then the index of the conditioning species, then the index for the species the marks belong to. That is,
 
$$E(M_1(S_1)|d_{11} = r), E(M_2(S_1)|d_{11} = r), \dots, E(M_m(S_1)|d_{11} = r), \dots,$$

$$E(M_1(S_1)|d_{12} = r), \dots, E(M_m(S_1)|d_{12} = r), \dots,$$

$$E(M_1(S_1)|d_{1s} = r), \dots, E(M_m(S_1)|d_{1s} = r),$$
 ...,
 
$$E(M_1(S_2)|d_{21} = r), \dots, E(M_m(S_2)|d_{21} = r), \dots,$$

$$E(M_1(S_2)|d_{2s} = r), \dots, E(M_m(S_2)|d_{2s} = r),$$
 ...,
 
$$E(M_m(S_s)|d_{ss} = r).$$
- `ETest` : matrix of 66 rows and the same structure of columns as  $E$ . The rows correspond to different algorithms for calculating the deviance of  $E$  from a horizontal line. Let denote by  $E(i)$  the value of the  $i$ th bin,  $i = 0, \dots, b$ , where  $E(0)$  is the bin that includes 0.
  - 1 :  $\max E(i) - \min E(i)$
  - 2-36 : see below
  - 37 :  $\sum |E_{i,\text{unbinned}} - E(0)|^2$
  - 38 :  $\sum |E_{i,\text{unbinned}} - E(0)|$

### 2-37:

results of (5 “norms”) x (7 set of weights)

**“norms”**

- 1 :  $\max w_i |E(i) - E(0)|$
- 2 :  $\sum w_i |E(i) - E(0)|^2$
- 3 :  $\sum w_i |E(i) - E(0)|$
- 4 :  $\sum w_i * r(E(i) - E(0))$
- 5 :  $\sum w_i * s(E(i) - E(0))$

where  $r$  is a function that first increases quadratically, then linearly:  $r(x) = x^2$  if  $x < q$ , and  $2qx - q^2$  otherwise, where  $q$  is the  $p$ th quantile of  $|E(i) - E(0)| : i=0, \dots, b$ . The functions  $s$  first increase linearly, then quadratically:  $s(x) = x$  if  $x < a$  and,  $b(x + a)^2$  otherwise. Here,  $a = 1/10$  and  $b_k = 0.25/a$ .

**Weights**

- a :  $w_i \sim 1$
- b :  $w_i \sim 1 / \sum_{j=0}^{i-1} \text{Ebin}(j)$
- c :  $w_i \sim \sqrt{1 / \sum_{j=0}^{i-1} \text{Ebin}(j)}$
- d :  $w_i \sim 1 / \sum_{j=0}^{i-1} \sqrt{\text{Ebin}(j)}$
- e :  $w_i \sim \text{Ebin}(i)$
- f :  $w_i \sim \sqrt{\text{Ebin}(i)}$
- g :  $w_i \sim 1 / \sqrt{\text{var}(E_j, \text{unbinned}; \text{distanceinithbin})}$

The sequence is 1a, 2a, ..., 9a, 1b, ..., 9f.

- VAR :  $(\text{length}(\text{bin}) - 1) \times (m(m - 1)s^2/2)$  matrix. function  $V$  if  $m=1$ . Otherwise, the covariances are returned in the following ordering. The index for the lower triangle of the covariance matrix (including the diagonal) runs fastest, then the index of the conditioning species, then the index for the species the marks belong to. That is,
 
$$\begin{aligned} & \text{Cov}(M_1(S_1), M_1(S_1)|d_{11} = r), \text{Cov}(M_1(S_1), M_2(S_1)|d_{11} = r), \dots, \text{Cov}(M_1(S_1), M_m(S_1)|d_{11} = \\ & r), \text{Cov}(M_2(S_1), M_2(S_1)|d_{11} = r), \dots, \text{Cov}(M_2(S_1), M_m(S_1)|d_{11} = r), \dots, \text{Cov}(M_m(S_1), M_m(S_1)|d_{11} = \\ & r), \dots, \\ & \text{Cov}(M_1(S_1), M_1(S_1)|d_{12} = r), \dots, \text{Cov}(M_m(S_1), M_m(S_1)|d_{12} = r), \dots, \\ & \text{Cov}(M_1(S_1), M_1(S_1)|d_{1s} = r), \dots, \text{Cov}(M_m(S_1), M_m(S_1)|d_{1s} = r), \\ & \dots \\ & \text{Cov}(M_1(S_2), M_1(S_2)|d_{21} = r), \dots, \text{Cov}(M_m(S_2), M_m(S_2)|d_{21} = r), \dots, \\ & \text{Cov}(M_1(S_2), M_1(S_2)|d_{2s} = r), \dots, \text{Cov}(M_m(S_2), M_m(S_2)|d_{2s} = r), \\ & \dots \\ & \text{Cov}(M_m(S_s), M_m(S_s)|d_{ss} = r). \end{aligned}$$
- VARTest : matrix of 66 rows and the same structure of columns as VAR. See VAR and ETest for details.
- SQ :  $(\text{length}(\text{bin}) - 1) \times (m(m - 1)s^2/2)$  matrix. Equals  $\text{sign}(V)\sqrt{|V|}$ .
- SQTest : Test results for SQ. Same matrix dimensions as VARTest.
- KMM : Stoyan's  $k_{mm}$  function
 
$$(\text{length}(\text{bin}) - 1) \times (ms(ms + 1)/2)$$
 matrix. The columns of KMM are returned in the following ordering: It is the lower triangle (including the diagonal) of expectation of the matrix  $vv^T$ . Here  $v$  is vector where first  $m$  components are the marks of species  $S_1$  the next components are the marks of  $S_2$  and so on. That is,
 
$$KMM(M_1(S_1), M_1(S_1)|d_{11} = r), \dots, KMM(M_m(S_1), M_1(S_1)|d_{11} = r), KMM(M_1(S_2), M_1(S_1)|d_{21} =$$

$r), \dots, KMM(M_m(S_2), M_1(S_1)|d_{21} = r), \dots, KMM(M_1(S_s), M_1(S_1)|d_{s1} = r), \dots, KMM(M_m(S_s), M_1(S_1)|d_{s1} = r),$   
 $\dots$   
 $KMM(M_2(S_1), M_2(S_1)|d_{11} = r), \dots, KMM(M_m(S_1), M_2(S_1)|d_{11} = r), KMM(M_1(S_2), M_2(S_1)|d_{21} = r), \dots, KMM(M_m(S_2), M_2(S_1)|d_{21} = r), \dots, KMM(M_1(S_s), M_2(S_1)|d_{s1} = r), \dots, KMM(M_m(S_s), M_2(S_1)|d_{s1} = r),$   
 $\dots$   
 $KMM(M_m(S_1), M_m(S_1)|d_{11} = r), KMM(M_1(S_2), M_m(S_1)|d_{21} = r), \dots, KMM(M_m(S_s), M_m(S_1)|d_{s1} = r),$   
 $\dots$   
 $KMM(M_1(S_2), M_1(S_2)|d_{22} = r), \dots, KMM(M_m(S_2), M_1(S_2)|d_{22} = r), KMM(M_1(S_3), M_1(S_2)|d_{32} = r), \dots, KMM(M_m(S_3), M_1(S_2)|d_{32} = r), \dots, KMM(M_1(S_s), M_1(S_2)|d_{s2} = r), \dots, KMM(M_m(S_s), M_1(S_2)|d_{s2} = r),$   
 $\dots$   
 $KMM(M_2(S_2), M_2(S_2)|d_{22} = r), \dots, KMM(M_m(S_2), M_2(S_2)|d_{22} = r), KMM(M_1(S_3), M_2(S_2)|d_{32} = r), \dots, KMM(M_m(S_3), M_2(S_2)|d_{41} = r), \dots, KMM(M_1(S_s), M_2(S_2)|d_{s2} = r), \dots, KMM(M_m(S_s), M_2(S_2)|d_{s2} = r),$   
 $\dots$   
 $KMM(M_m(S_2), M_m(S_2)|d_{22} = r), KMM(M_1(S_3), M_m(S_2)|d_{21} = r), \dots, KMM(M_m(S_s), M_m(S_2)|d_{s2} = r),$   
 $\dots$   
 $KMM(M_1(S_s), M_1(S_s)|d_{ss} = r), \dots, KMM(M_m(S_s), M_1(S_s)|d_{ss} = r), KMM(M_2(S_s), M_2(S_s)|d_{ss} = r), \dots, KMM(M_m(S_s), M_2(S_s)|d_{ss} = r), \dots, KMM(M_m(S_s), M_m(S_s)|d_{ss} = r),$

- **GAM**: mark variogramm  
(length(bin) - 1) × (ms(ms + 1)/2) matrix; see **KMM** for details.
- **Ebin**: number of values a binned value of  $E$  is based on.
- **VARbin**: number of values a binned value of  $V$  or  $SQ$  is based on.
- **KMMbin**: number of values a binned value of Stoyan's  $k_{mm}$  function is based on.
- **GAMbin**: number of values a binned value of the mark variogram is based on.
- **midbin**: centers of the bins
- **call**: match.call() of mpp.characteristics

summarize=FALSE: the column structure of the above matrices is rep times repeated.

## Value

mpp.characteristics returns list(E, ETest, VAR, VARTest, SQ, SQTest, KMM, GAM, Ebin, VARbin, KMMbin, GAMbin, midbin, call = match.call()); see **Details**. The return is invisible if show=TRUE.

## Author(s)

Martin Schlather, <martin.schlather@math.uni-goettingen.de> <http://www.stochastik.math.uni-goettingen.de/institute>

## References

Schlather, M., Ribeiro, P., and Diggle, P. (2004) Detecting Dependence Between Marks and Locations of Marked Point Processes *J. R. Statist. Soc., Ser. B*, .

Schlather, M. (2001) On the second order characteristics of marked point processes, *Bernoulli* **7**, 99-117.

## See Also

[rfm.test](#), [simulateMPP](#)

## Examples

```
data(BITOEK)

bin <- c(-1, seq(0, 50, 2))
normalize <- TRUE
mpp <- mpp.characteristics(bin=bin, normalize=normalize, show=TRUE,
                           coord=steigerwald$coord, diam=steigerwald$diam)
str(mpp)
```

---

rfm.test

*MC test on random field model*

---

## Description

`rfm.test` performs MC tests which enables the user to decide whether a marked point process may be considered as a random field model, i.e., as a model where the marks are independent of the locations

## Usage

```
rfm.test(coord=NULL, data, normalize=TRUE, MCrepetitions=99,
         MCmodel=list(model="exponential",
                      param=c(mean=0, variance=NA, nugget=0, scale=NA)),
         method=NULL,
         bin=c(-1, seq(0, 1.2, l=15)), MCreregister=1, n.hypo=1000,
         pvalue=c(10, 5, 1), tests="l1 & w3",
         tests.lp=NULL, tests.weight=NULL, Barnard=FALSE,
         PrintLevel=RFparameters()$Print, ...
         )
```

## Arguments

<code>coord</code>	matrix with 2 columns; the coordinates of the points
<code>data</code>	vector or matrix; the univariate marks that correspond to the locations; if <code>data</code> is a matrix then each column is interpreted as an independent observation given the locations <code>coord</code> ; see Details for further possibilities

<code>normalize</code>	logical; if TRUE the data are transformed to standard normal data before analysed; if data is a matrix this is done for each column separately
<code>MCrepetitions</code>	usually 19 or 99; number of simulations that are compared with the data
<code>MCmodel</code>	variogram model to be fitted, see <code>fitvario</code> .
<code>method</code>	method used to simulate Gaussian random fields; see <code>GaussRF</code>
<code>bin</code>	sequence of increasing bin margins for calculating the function E, V, etc in analogy to the binning for variograms; see Details
<code>MRegister</code>	0:9; the register to which intermediate results are stored when the random fields are generated for the MC test
<code>n.hypo</code>	number of repeated MC tests to determine the <code>pvalue</code> -position for the Null-hypothesis. If the variogram were not estimated, this position would be $(1 - pvalue)(MCrepetitions + 1)$ . see Details
<code>pvalue</code>	test levels in percent. Only values below 50 are accepted; otherwise 100-pvalue is regarded as <code>pvalue</code> (to be consistent with the former definition)
<code>tests</code>	vector of characters, see Details.
<code>tests.lp</code>	vector of characters, see Details.
<code>tests.weight</code>	vector of characters, see Details.
<code>Barnard</code>	test by Barnard (1963) on the independence of marks
<code>PrintLevel</code>	If zero then no messages are printed. The higher the value the more tracing information is given.
<code>...</code>	any parameter for <code>fitvario</code> can be passed, except for <code>x</code> , <code>y</code> , <code>z</code> , <code>T</code> , <code>data</code> , <code>model</code> , <code>param</code> , <code>mle.methods</code> and <code>cross.methods</code>

## Details

`data`: there are three possibilities to pass the data

- `data` a vector or matrix, `coord` contains the coordinates, as described above
- `data=list(coord=, data=)` and `coord=NULL`
- `data=list(list(coord=, data=), ..., list(coord=, data=))`; several data sets are analysed and all the results are summed up, and returned in a single matrix E (or V or SQ)

`bin`: as the variogram in geostatistics, the characteristics for the marks of a marked point process depend on a distance (vector)  $r$ . Instead of returning a cloud of values, binned values are calculated in the same way the binned variogram is obtained. `bin` gives the margins of the bins (left open, right closed ones) as an increasing sequence. The first bin must include the zero, i.e., `bin=c(-1, 0, ...)`.

`n.hypo`: for correct appreciation of the relative position of the statistic for the data set w.r.t. the simulations, the reference values for the estimated `pvalue` level must be determined:

- the parameters of a Gaussian random field are adapted to the data set
- `n.hypo` realisations of the Gaussian random field are simulated

- for each realisation, an MC test based on `MCrepetitions` is performed (estimation of the parameters of the random field, and test statistics for 99 realisations)
- the position of the statistic for the simulation within the statistics of the 99 realisations are calculated.
- The position of the largest `pvalue` positions is determined, which have values around  $(1 - pvalue)(MCrepetitions + 1)$ ; this position is returned in `null.sl` as reference values for the estimated `pvalue` level

`tests`, `tests.lp`, `tests.weight`:

- If `tests="all"` then the results for all test variants are returned, independently of the values of `tests.lp` and `tests.weight`
- else the results of all tests given by `tests` and the combinations of `tests.lp` and `tests.weight` are given.
- The values for `tests.lp` are “max” (maximum norm), “l2” (l2 norm), “l1” (l1 norm), “robust” (the distance is squared for small distances only), “anti” (the distance is square for large distances only)
- The values for `tests.weight` are “const” (constant weight), “1/sum#” (‘sum#’ is the cumulative sum of the number of points in all bins to the left, and the considered bin itself), “sqrt(1/sum#)” (sqrt of ‘1/sum#’), “1/sumsqrt#” (similar to ‘1/sum#’, but square root of the number of points is summed up), “#” (number of points within a bin), “sqrt#” (square root of the number of points), “1/sd” (sd=estimated standard deviation within a bin)

or, equivalently,

“w1”, “w2”, “w3”, “w4”, “w5”, “w6”, “w7”.

- The values for `tests` are “max & const”, “l2 & const”, “l1 & const”, “robust & const”, “anti & const”, “max & 1/sum#”, “l2 & 1/sum#”, “l1 & 1/sum#”, “robust & 1/sum#”, “anti & 1/sum#”, “max & sqrt(1/sum#)”, “l2 & sqrt(1/sum#)”, “l1 & sqrt(1/sum#)”, “robust & sqrt(1/sum#)”, “anti & sqrt(1/sum#)”, “max & 1/sumsqrt#”, “l2 & 1/sumsqrt#”, “l1 & 1/sumsqrt#”, “robust & 1/sumsqrt#”, “anti & 1/sumsqrt#”, “max & #”, “l2 & #”, “l1 & #”, “robust & #”, “anti & #”, “max & sqrt#”, “l2 & sqrt#”, “l1 & sqrt#”, “robust & sqrt#”, “anti & sqrt#”, “max & 1/sd”, “l2 & 1/sd”, “l1 & 1/sd”, “robust & 1/sd”, “anti & 1/sd”,

or, equivalently,

“max & w1”, “l2 & w1”, “l1 & w1”, “robust & w1”, “anti & w1”, “max & w2”, “l2 & w2”, “l1 & w2”, “robust & w2”, “anti & w2”, “max & w3”, “l2 & w3”, “l1 & w3”, “robust & w3”, “anti & w3”, “max & w4”, “l2 & w4”, “l1 & w4”, “robust & w4”, “anti & w4”, “max & w5”, “l2 & w5”, “l1 & w5”, “robust & w5”, “anti & w5”, “max & w6”, “l2 & w6”, “l1 & w6”, “robust & w6”, “anti & w6”, “max & w7”, “l2 & w7”, “l1 & w7”, “robust & w7”, “anti & w7”

and “range” (difference largest positive and largest negative deviation for all bins), “no.bin.sq” (l2 norm where the bins are chosen so that they contain only 1 point), “no.bin.abs” (l1 norm where the bins are chosen so that they contain only 1 point)

## Value

Let  $n$  be the number of MC tests chosen by the user. Then `rfm.test` returns a list of the following elements:

**E (E function)**

matrix of  $n$  columns. The number of rows depends on the input parameters: If only one realisation of the data is given then the absolute test positions of the MC test is returned, i.e. a value between 1 and `MCrepetition + 1`, inclusively.

If several realisations of the data (and the `coord`) are given, then the number of rows equals `MCrepetition + 1`, and the  $k$ th row gives the number of test statistics with position  $k$ . The first situation is the standard one for the user. The second situation appears when `rfm.test` is recalled to calculate the intermediate result `null.hypo`, see below.

**VAR (V function)**

matrix of  $n$  columns. See E above.

**SD (the square root of the V function)**

matrix of  $n$  columns. See E above.

**reject.null**

list of logical matrices that indicate whether E, VAR or SQ should be rejected at the given levels, i.e. whether the positions of the tests statistics for E, VAR or SQ are at least as large as the estimated reference values given by `null.sl`.

**est**

list of variogram models according to `MCmodel` estimated from the data.

**normalize**

The input parameter `normalize`.

**MCrepetitions**

The input parameter `MCrepetitions`.

**MCmodel**

The input parameter `MCmodel`.

**null.hypo**

`null.hypo` stores intermediate results that are usually not of interest for the user. `n.hypo` simulations have been performed under the null hypothesis to determine the `pvalue` test positions. (The explicite determination is necessary, since parameters of the variogram have to be estimated within the null hypothesis.) For these `n.hypo` simulations, `rfm.test` is run and `null.hypo` gives the results. Note that here, all test variants are considered.

**null.sl**

List of matrices. They give the reference values for the estimated `pvalue` level. The values are around  $(1-pvalue) * (MCrepetitions + 1)$ , but can range between 1 and `MCrepetitions + 2`. If a value of `MCrepetitions + 2` occurs, usually, `MCrepetitions` and/or `n.hypo` have been chosen too small.

bin  
the binning used to calculate E, VAR and SQ

### Note

In comparison to version 0.1 of `MarkedPointProcess` and the paper by Schlather et al. (2004), the announced positions of the test statistics for E, VAR, SD and `null.sl` are all increased by 1, now ranging from 1 to 100 instead of 0 to 99, for the standard settings.

### Author(s)

Martin Schlather, ([martin.schlather@math.uni-goettingen.de](mailto:martin.schlather@math.uni-goettingen.de)) <http://www.stochastik.math.uni-goettingen.de/institute>

### References

- Barnard, G. (1963) Discussion paper to M.S. Barlett on “The spectral analysis of point processes”, *J. R. Statist. Soc. Ser. B*, **25**, 294.
- Besag, J. and Diggle, P. (1977) Simple Monte Carlo tests for spatial pattern. *J. R. Statist. Soc. Ser. C*, **26**, 327–333.
- Schlather, M., Ribeiro, P. and Diggle, P. (2004) Detecting Dependence Between Marks and Locations of Marked Point Processes *J. R. Statist. Soc., Ser. B* **66**, 79-83.

### See Also

[mpp.characteristics](#), [simulateMPP](#)

### Examples

```
data(BITOEK)
d <- steigerwald
plotWithCircles(cbind(d$coord, d$diam), factor=2)
mpp.characteristics(x=d$coord, data=d$diam,
                   bin=c(-1, seq(0, 50, 2)), show=interactive())

## testing for E=const, V=const or SD=const (this takes several minutes!)
res <- rfm.test(d$coord, d$diam, MCrep=if (interactive()) 99 else 9,
               n.hypo=if (interactive()) 100 else 2)

## test statistics for the data
res$E
res$VAR

## reference values for the estimated 10%, 5% and 1% level
res$null.sl

## should E=const, V=const or SD=const be rejected at the given levels?
res$reject.null
```

---

 simulateMPP

*Simulation of marked point processes*


---

## Description

simulateMPP generates realisations of marked point processes

## Usage

```
simulateMPP(coordmodel=c("given", "uniform", "Poisson"),
            coord=NULL, npoints=NULL, lambda=NULL,
            window=NULL, edgcorrection=0.0,
            repetitions=1, coordrepet=1, model=NULL,
            register=0, method=NULL)
```

## Arguments

coordmodel	if coordmodel="given" then coord are expected to be given and not simulated; if coordmodel="uniform" then npoints uniformly distributed points are created; if coordmodel="Poisson" then a conditional Poisson point process is simulated with intensity lambda
coord	matrix with 2 columns; coordinates of the points; coord is given only if coordmodel="given"
npoints	number of coordinates; npoints must be given if coordmodel="uniform".
lambda	intensity of the Poisson process; lambda must be given if coordmodel="Poisson".
window	= c(xlim, ylim). window must be given if coordmodel equals "uniform" or "Poisson".
edgcorrection	double. If edgcorrection > 0 then a Poisson process is simulated with intensity lambda in a frame of thickness edgcorrection around the window. If window is not given, the range of the x values and the range of the y values are taken to define the window. If lambda is not given, the intensity within the window is used instead.
repetitions	integer; number of independent drawings of the marks for a given set of coordinates
coordrepet	number of independent drawing of the coordinates; this parameter is ignored in case of coordmodel="given"
model	list of lists; model for the marks; see Details and <a href="#">get.mpp.names</a> .
register	the register where intermediate results in the Gaussian random field simulation are stored, see <a href="#">GaussRF</a>
method	the method by which the Gaussian random field is simulated; if is.null(method) then the method is chosen automatically, see <a href="#">GaussRF</a>

## Details

The definition of a model is of the form `model = list(l.1, OP.1, l.2, OP.2, ..., l.n)`. The lists `l.i` are all either of the form `l.i = list(model=, var=, kappas=, scale=)` or of the form `l.i = list(model=, var=, kappas=, aniso=)` in case of random field parts, or of the form `l.i = list(model=, param=)` in case of marked point process parts. `l.i$model` is a string; `var` gives the variance; `scale` is a scalar whereas `aniso` is a  $d \times d$  matrix, which is multiplied from left to the points, and at the transformed points the values of the random field are calculated. The dimension  $d$  of matrix must match the number of rows of `x`. `param` is vector of real values whose length depends on the specified `model`. The models for the random field part can be combined by `OP.i="+"` or `OP.i="*"`, those for the marked point process parts only by `OP.i="+"`.

## Value

`coordrepet=1`  
the function returns `list(coord, data)`, `data` contains the independent drawing of the marks (as columns)

`coordrepet>1`  
the function returns `list(list(coord, data), ..., list(coord, data))`

## Author(s)

Martin Schlather, [martin.schlather@math.uni-goettingen.de](mailto:martin.schlather@math.uni-goettingen.de) <http://www.stochastik.math.uni-goettingen.de/institute>

## See Also

[get.mpp.names](#), [rfm.test](#), [simulateMPP](#), [splitmodel](#), [MarkedPointProcess](#)

## Examples

```
xlim <- c(0, if (interactive()) 200 else 20)
mpp <- simulateMPP(coordmodel="Poisson", lambda=1,
  window=c(xlim=xlim, ylim=c(20, 70)),
  repet=3, coordrepet=4,
  model=list(list(model="exp", var=1, scale=10),
    "+",
    list(model="nearest neighbour", p=1)))
str(mpp)
```

---

splitmodel

*Split between marked point processes and random fields*

---

## Description

`splitmodel` splits a model given in form of a list (the third variant of model definition for random fields, see [CovarianceFct](#)) into a random field part and a marked point process part

**Usage**

```
splitmodel(model)
```

**Arguments**

`model`            The definition of a model is of the form `model = list(l.1, OP.1, l.2, OP.2, ..., l.n)`. The lists `l.i` are all either of the form `l.i = list(model=, var=, kappas=)` or of the form `l.i = list(model=, var=, kappas=, aniso=)` in case of random field parts, or of the form `l.i = list(model=, param=)` in case of marked point process parts. `l.i$model` is a string; `var` gives the variance; `scale` is a scalar whereas `aniso` is a  $d \times d$  matrix, which is multiplied from the right to the points, and at the transformed points the values of the (isotropic) random field (with scale 1) are calculated. The dimension  $d$  of matrix must match the number of rows of `x`. `param` is vector of real values whose length depends on the specified `model`. The models for the random field part can be combined by `OP.i="+"` or `OP.i="*"`, those for the marked point process parts only by `OP.i="+"`.

**Value**

`list(RF=RF, mpp=mpp)` where `RF` is a usual model definition for a random field. Further, `mpp=list(mpp.1, ..., mpp.n)`, where `mpp.i=list(model=model, param=param, mnr=)` and `mnr` is the internal C code for `model`.

**Author(s)**

Martin Schlather, [martin.schlather@math.uni-goettingen.de](mailto:martin.schlather@math.uni-goettingen.de) <http://www.stochastik.math.uni-goettingen.de/institute>

**See Also**

[simulateMPP](#)

**Examples**

```
str(splitmodel(list(list(model="exp", var=5, scale=3))))
str(splitmodel(list(list(model="nearest neighbour", param=4))))
str(splitmodel(list(list(model="exp", var=5, scale=3),
                    "+",
                    list(model="nearest neighbour", param=4)
                    )))
str(splitmodel(list(list(model="exp", var=5, scale=3),
                    "*",
                    list(model="spherical", var=1, scale=2),
                    "+",
                    list(model="nearest neighbour", param=4),
                    "+",
```

```
list(model="random coin",
      param=c(fct=1, scale=7, height=8))
)))
```

srd

*Simulation study for marked point processes***Description**

The function reproduces the simulation study published by Schlather, Ribeiro and Diggle (2004)

**Usage**

```
srd.jrssb(input=NULL, repet=500, dev=2, PrintLevel=2, readlines=TRUE,
          ps.path="ps/", data.save.path="data/", simu.path="simu/",
          tex.path="tex/", biondi.etal=NULL, final=TRUE)
```

**Arguments**

<code>input</code>	vector of values for automatic choice of the submenus
<code>repet</code>	number of realisations investigated for each parameter set. <code>2 * repet</code> gives also roughly the number of hours needed to do all the simulations!
<code>dev</code>	device, see <a href="#">Dev</a>
<code>PrintLevel</code>	If <code>PrintLevel &gt; 1</code> some information on the course of the algorithm is printed
<code>readlines</code>	logical. Only used if <code>is.numeric(dev)</code> . If the operating system is unix and <code>readlines=FALSE</code> then the system waits a second before the next plot is calculated or shown. Otherwise the system waits for return.
<code>ps.path</code>	path used for all postscript files in <code>srd.jrssb</code>
<code>data.save.path</code>	directory where intermediate results in the analysis of the data are stored
<code>simu.path</code>	directory used to store intermediate results for the simulations
<code>tex.path</code>	path used for all created tex files containing tables
<code>biondi.etal</code>	list of two components: <ul style="list-style-type: none"> <li><code>coord</code>: a matrix of two columns for the coordinates and</li> <li><code>diameter</code>: a vector for the marks</li> </ul> <p>The data set by Biondi et al., used in Schlather, Ribeiro and Diggle (2004), is not freely available.</p>
<code>final</code>	logical. If <code>FALSE</code> then several parameters are changed so that the algorithms run faster. Especially <code>repet</code> is limited to 3, <code>optim</code> is very sloppy, and the parameter sets for the simulation study are remarkably reduced. <code>final=FALSE</code> is used for debugging or demonstration, but the results are not reliable.

## Details

The simulation study is written in a way that `srd.jrssb()` might be called several times on a parallel CPU system or PVD from the same directory. The non-interaction of the different instances is managed by means of lock files. It is ensured that no simulation study for a specific parameter set is performed twice. However, the programming has been kept simple so that a lock files might be created but the simulation is not performed by any process because of strange interaction. In this case all remaining lock files have to be deleted after the first run of the simulation study, and the study has to be re-run. Then, `srd.jrssb()` does the simulation study only for remaining cases.

For both the simulation studies and the data analyses, intermediate stored results are re-used if run again.

## Value

NULL. Side effect is the creation of postscript files and tex files.

## Author(s)

Martin Schlather, [martin.schlather@math.uni-goettingen.de](mailto:martin.schlather@math.uni-goettingen.de) <http://www.stochastik.math.uni-goettingen.de/institute>

## References

Schlather, M., Ribeiro, P. and Diggle, P. (2004) Detecting Dependence Between Marks and Locations of Marked Point Processes. *J. R. Statist. Soc., Ser. B* **66**, 79-83.

Biondi, F., Myers, D.E., and Avery, C.C. (1994) Geostatistically modeling stem size and increment in an old-growth forest. *Can. J. Forest Res.* **24**, 1354-1368.

## See Also

[rfm.test](#), [simulateMPP](#), [MarkedPointProcess](#)

## Examples

```
# repet=2 is for demonstration of functionality
# repet=50 gives more reliable results
# repet=500 and final=TRUE have been used in the simulation study

printlevel <- 1 + !interactive()
srd.jrssb(input=if(!interactive()) c(1:2,0), repet=2, dev=TRUE,
          final=FALSE, Pr=printlevel)
```

# Index

## \*Topic **datasets**

BITOEKforests, 1

## \*Topic **spatial**

get.mpp.names, 2

MarkedPointProcess, 3

mpp.characteristics, 4

rfm.test, 8

simulateMPP, 13

splitmodel, 14

srd, 16

BITOEK, 3

BITOEK (*BITOEKforests*), 1

BITOEKforests, 1

coulissenhieb (*BITOEKforests*), 1

CovarianceFct, 4, 14

Dev, 5, 16

fitvario, 9

forests (*BITOEKforests*), 1

GaussRF, 9, 13

get.mpp.names, 2, 3, 13, 14

MarkedPointProcess, 3, 14, 17

mpp.characteristics, 3, 4, 12

optim, 16

rfm.test, 3, 8, 8, 14, 17

simulateMPP, 3, 8, 12, 13, 14, 15, 17

splitmodel, 3, 14, 14

srd, 16

srd.jrssb, 3

steigerwald (*BITOEKforests*), 1