Package 'RandomFields'

October 1, 2014

Version 3.0.44

Title Simulation and Analysis of Random Fields

Author Martin Schlather [aut, cre], Alexander Malinowski [aut], Marco Oesting [aut], Daphne Boecker [aut], Kirstin Strokorb [aut], Sebastian Engelke [aut], Johannes Martini [aut], Felix Ballani [aut], Peter Menck [aut], Sebastian Gross [aut], Ulrike Ober [ctb], Katharina Burmeister [ctb], Juliane Manitz [ctb], Paulo Ribeiro [ctb], Richard Singleton [ctb], Ben Pfaff [ctb], R Core Team [ctb]

Maintainer Martin Schlather <schlather@math.uni-mannheim.de>

Depends R (>= 3.0.2), sp

Imports colorspace, graphics, methods, spam, tcltk2, tcltk, tkrplot

Suggests raster

Description Simulation of Gaussian and extreme value random fields; conditional simulation; kriging; maximum likelihood estimation

License GPL (>= 3)

URL http://ms.math.uni-mannheim.de/de/publications/software

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-10-01 11:03:13

R topics documented:

`RandomFields-package` *Simulation and Analysis of Random Fields*

Description

The package `RandomFields` offers various tools for

1. **simulation** of different kinds of random fields, including
   - multivariate, spatial, spatio-temporal Gaussian random fields,
   - Poisson fields, binary fields, Chi2 fields, t fields and
   - max-stable fields.

   It can also deal with non-stationarity and anisotropy of these processes and conditional simulation (for Gaussian random fields, currently).
2. **model estimation (ML) and inference (tests)** for regionalizd variables and data analysis,
3. **model estimation** for (geostatistical) linear (mixed) models

See `http://ms.math.uni-mannheim.de/de/publications/software/` for **intermediate updates**.

Details

The following features are provided by the package:

1. **Simulation**
   - RFsimulate: Simulation of random fields, including conditional simulation. For a list of all covariance functions and variogram models see RMmodel. Use plot for visualisation of the result.
2. **Estimation of parameters (for second-order random fields)**
   - RFfit : general function for estimating parameters; (for Gaussian random fields)
   - RFhurst : estimation of the Hurst parameter
   - RFfractaldim : estimation of the fractal dimension
   - RFempiricalvariogram : calculates the empirical variogram
3. **Prediction (for second-order random fields)**
   - RFinterpolate : kriging, including imputing
4. **Inference (for Gaussian random fields)**
   - RFcrossvalidate : cross validation
   - RFratiotest : likelihood ratio test
   - AIC, AICc, BIC, anova, logLik
5. **Models**
   - For a list of covariance and variogram models –e.g. for **geostatistical** purposes– see RMmodel. More sophisticated models and covariance function operators are included.
   - To apply the offered package procedures to **mixed models** – e.g. appearing in genetical data analysis– see RFformula.
   - models are evaluated by RFcov, RFvariogram and RFcovmatrix. For a quick impression use plot(model).
6. **Data and example studies**: Some data sets and published code are provided to illustrate the syntax and structure of the package functions.
   - soil : soil physical data
   - weather : UWME weather data
   - papers : code used in the papers published by the author(s)
7. **Graphics**
   - Fitting a covariance function manually RFgui
   - the generic function plot
8. **S3 and S4 objects**
   - The functions return S4 objects based on the package **sp**, if spConform=TRUE. This is the default.
     If spConform=FALSE, simple objects as in version 2 are returned. These simple objects are frequently provided with an S3 class. This options makes the returning procedure much faster, but currently does not allow for the comfortable use of plot.
   - plot, print, summary, sometimes also str recognise these S3 and S4 objects
   - use sp2RF for an explicite transformation of **sp** objects to S4 objects of **RandomFields**.

- Further generic functions are available for fitted models, see 'Inference' above.

9. **Advanced** users, package programmers
   - might decide on a large variety of arguments of the simulation and estimation procedures using the function `RFoptions`
   - may use './configure –with-tcl-config=/usr/lib/tcl8.5/tclConfig.sh –with-tk-config=/usr/lib/tk8.5/tkConfig.sh' to configure R

Changings

A list of major changings from Version 2 to Version 3 can be found in MajorRevisions.

Changings lists some further changings, in particular of argument and argument names.

Update

Current updates are available through `http://ms.math.uni-mannheim.de/de/publications/software`.

Contributions

- Contributions to version 3.0 and following:
  Felix Ballani (TU Bergakademie Freiberg; Poisson Polygons, 2014)
  Daphne Boecker (Univ. Goettingen; RFgui, 2011)
  Katharina Burmeister (Univ. Goettingen; testing, 2012)
  Sebastian Engelke (Univ. Goettingen; RFempiricalvariogram, 2011-12)
  Sebastian Gross (Univ. Goettingen; tilde formulae, 2011)
  Alexander Malinowski (Univ. Mannheim; S3, S4 classes 2011-13)
  Juliane Manitz (Univ. Goettingen; testing, 2012)
  Johannes Martini (Univ. Goettingen; RFempiricalvariogram, 2011-12)
  Ulrike Ober (Univ. Goettingen; help pages, testing, 2011-12)
  Marco Oesting (Univ. Mannheim; Brown-Resnick processes, Kriging, Trend, 2011-13)
  Paulo Ribeiro (Unversidade Federal do Parana; code adopted from **geoR**, 2014)
  Kirstin Strokorb (Univ. Mannheim; help pages, 2011-13)

- Contributions to version 2.0 and following:
  Peter Menck (Univ. Goettingen; multivariate circulant embedding)
  R Core Team, Richard Singleton (fft.c and advice)

- Contributions to version 1 and following:
  Ben Pfaff, 12167 Airport Rd, DeWitt MI 48820, USA making available an algorithm for AVL trees (avltr*)

Thanks

Patrick Brown : comments on Version 3
Paulo Riberio : comments on Version 1
Martin Maechler : advice for Version 1

Financial support

- V3.0 has been financially supported by the German Science Foundation (DFG) through the Research Training Group 1953 'Statistical Modeling of Complex Systems and Processes — Advanced Nonparametric Approaches' (2013-2018).
- V3.0 has been financially supported by Volkswagen Stiftung within the project 'WEX-MOP' (2011-2014).
- Alpha versions for V3.0 have been financially supported by the German Science Foundation (DFG) through the Research Training Groups 1644 'Scaling problems in Statistics' and 1023 'Identification in Mathematical Models' (2008-13).
- V1.0 has been financially supported by the German Federal Ministry of Research and Technology (BMFT) grant PT BEO 51-0339476C during 2000-03.
- V1.0 has been financially supported by the EU TMR network ERB-FMRX-CT96-0095 on "Computational and statistical methods for the analysis of spatial data" in 1999.

Note

The following packages enable further features in RandomFields: **optimx**, **soma**, **GenSA**, **minqa**, **pso**, **DEoptim**, **nloptr**, **RColorBrewer**, **colorspace**

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Singleton, R.C. (1979). In *Programs for Digital Signal Processing* Ed.: Digital Signal Processing Committee and IEEE Acoustics, Speech, and Signal Processing Committe (1979) IEEE press.
- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Strokorb, K. (2013) Analysis, simulation and prediction of multivariate Random Fields with package **RandomFields**. *Submitted to JSS*, see the corresponding vignette.

See Also

See also RF, RM, RP, RR, RC

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

# simulate some data first (Gaussian random field with exponential
# covariance; 6 realisations)
model <- RMexp()
x <- seq(0, 10, if (interactive()) 0.1 else 1)
z <- RFsimulate(model, x, x, n=6)
```

```
## select some data from the simulated data
xy <- as.matrix(expand.grid(x=x, y=x))
pts <- sample(nrow(xy), 100)
data <- matrix(ncol=6, as.vector(z))[pts, ]
data <- cbind(xy[pts, ], data)
plot(z, data)

## re-estimate the parameter (true values are 1)
estmodel <- RMexp(var=NA, scale=NA)
(fit <- RFfit(estmodel, data=data))

## show a kriged field based on the estimated parameters
kriged <- RFinterpolate(fit, x, x, data=data)
plot(kriged, data)
```

`Advanced Max-stable random fields`
Simulation examples of advanced Max-Stable Random Fields

Description

Here, an advanced example is given used to test whether the algorithms work correctly.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Strokorb, K. (2013) Ph.D. thesis.

See Also

RPmaxstable

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again


n <- if (interactive()) 5000 else 3
step <- if (interactive()) 0.2 else 2


model <- RMexp(var=1.62 / 2)
x <- seq(0, 5, step)
y <- seq(0, 10, step)
```

```
auswertung <- function(simu, model, threshold=2) {
  simu <- as.array(simu)
  below <- simu <= threshold
  freq <- rowMeans(below)
  meanfreq <- mean(freq)
  Print(freq, meanfreq, exp(-1/threshold)) ## univariate kontrolle
  both <- t(below) & below[1, ]
  ecf <-  2-log(colMeans(both)) / log(meanfreq)
  plot(x, ecf)

  ## alle 3 Linien ergeben das Gleiche:
  lines(x, m1 <- RFcov(RMbrownresnick(model), x), col="yellow")
  lines(x, m2 <- RFcov(RMschlather(RMbr2eg(model)), x), col="red", lty=2) # OK
  m3 <- RFcov(RMbernoulli(RMbr2bg(model), centred=FALSE), x)
  lines(x, m3, col="blue", lty=3)

  erfc <- function(x) 2 * pnorm(x, 0, sd=1/sqrt(2), lower=FALSE)
  lines(x, m4 <- erfc(0.45 * sqrt(1-exp(-x))), lty=4)

  ## theoretical curves correct?
  if (!all.equal(m1, m2) || !all.equal(m1, m3) || !all.equal(m1, m4))
    stop("calculation error")

  if ( (n <- ncol(simu)) >= 1000) {
    ## margins correct?
    mar.threshold <- 4 * 0.5 / sqrt(n)
    mmar.threshold <- 3 * 0.5 / sqrt(n)
    Print(abs(freq - exp(-1/threshold)), mar.threshold)
    if (abs(freq[sample(length(freq), 1)] - exp(-1/threshold)) > mar.threshold)
      stop("marginal distribution wrong? (single margin)")
    if (abs(meanfreq - exp(-1/threshold)) > mmar.threshold)
      stop("marginal distribution wrong? (mean margin)")

    ## extremal correlation function correct?
    meanthreshold <- 4 / sqrt(n)
    maxthreshold <- 2 * sqrt(nrow(simu)) / sqrt(n)
    Print(abs(ecf - m1), meanthreshold, maxthreshold)
    if (mean(abs(ecf - m2)) > meanthreshold)
      stop("ecf not correct? (mean deviation too large)")
    if (max(abs(ecf - m2)) > maxthreshold)
      stop("ecf not correct? (max deviation too large)")
  }
}


## Brown-Resnick
z <- RFsimulate(RPbrownresnick(model), y, y)
plot(z)
simu <- RFsimulate(RPbrownresnick(model), x, n=n,  max_gauss=5)
auswertung(simu, model)
```

```
## Extremal Gaussian
z <- RFsimulate(RPschlather(RMbr2eg(model)), y, y)
plot(z)
simu <- RFsimulate(RPschlather(RMbr2eg(model)), x,  n=n)
auswertung(simu, model)


## Extremal Binary Gaussian
binary.model <- RPbernoulli(RMbr2bg(model))
z <- RFsimulate(RPschlather(binary.model), y, y)
plot(z)
simu <- RFsimulate(RPschlather(binary.model), x, n=n, max_gauss=5)
auswertung(simu, model)
```

`Baysian` *Baysian Spatial Modelling*

Description

Baysian modelling is a future project. However, some kind of hierarchical modelling is already available, see RRmodels.

Details

– this is a future project –

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodelsAdvanced

For hierarchical modelling see RR

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## See 'RRmodels'for hierarchical models
```

Brown-Resnick-Specific

Simulation methods for Brown-Resnick processes

Description

These models define the particular way to simulate Brown-Resnick processes

Usage

```
RPbrmixed(phi, tcf, xi, mu, s, meshsize, vertnumber, optim_mixed,
          optim_mixed_tol, optim_mixed_maxpo, lambda, areamat, variobound)

RPbrorig(phi, tcf, xi, mu, s)

RPbrshifted(phi, tcf, xi, mu, s)
```

Arguments

`phi`	object of class `RMmodel`; specifies the covariance model to be simulated.
`tcf`	the extremal correlation function; either `phi` or `tcf` must be given.
`xi, mu, s`	the shape parameter, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.
`lambda`	positive constant factor in the intensity of the Poisson point processused in the M3 representation, cf. Thm. 6 and Remark 7 in Oesting et. al (2012); can be estimated by setting `optim_mixed` if unknown. Default value is 1.
`areamat`	vector or matrix of values in $[0, 1]$ with odd length (odd number of rows and columns, respectively). Each value represents the portion of processes whose maximum is located at a specific location on a grid taken into account for the simulation of the shape function in the M3 representation. The center of `areamat` represents the value for the origin, the other entries belong to the corresponding locations on a 1D or 2D grid. `areamat` can be used for dimensions 1 and 2 only; can be optimized by setting `optim_mixed` if unknown. Default value is 1.
`meshsize, vertnumber, optim_mixed, optim_mixed_tol, optim_mixed_maxpo, variobound`	further arguments for simulation via the mixed moving maxima (M3) representation; see `RFoptions`

Details

The argument `xi` is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical value or given a `RMmodel`, in particular by a `RMtrend` model.

The functions `RPbrorig`, `RPbrshifted` and `RPbrmixed` simulate a Brown-Resnick process, which is defined by

$$Z(x) = \max_{i=1}^{\infty} X_i \exp(W_i(x) - \gamma),$$

where the X_i are the points of a Poisson point process on the positive real half-axis with intensity $x^{-2}dx$, $W_i \sim W$ are iid centered Gaussian processes with stationary increments and variogram γ given by `model`. The functions correspond to the following ways of simulation:

`RPbrorig` simulation via using the original definition (method 0 in Oesting et al., 2012)

`RPbrshifted` simulation using a random shift (similar to method 1 and 2)

`RPbrmixed` simulation using M3 representation (method 4)

Value

The functions return an object of class `RMmodel`

Note

Advanced options for `RPbroriginal` and `RPbrshifted` are `maxpoints` and `max_gauss`, see `RFoptions`.

Author(s)

Marco Oesting, <oesting@math.uni-mannheim.de>, Martin Schlather, <schlather@math.uni-mannheim.de>
http://ms.math.uni-mannheim.de/de/publications/software

References

- Oesting, M., Kabluchko, Z. and Schlather M. (2012) Simulation of Brown-Resnick Processes, *Extremes*, **15**, 89-107.

See Also

`RPbrownresnick`, `RMmodel`, `RPgauss`, `maxstable`, `maxstableAdvanced`

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
  model <- RPbrshifted(RMfbm(alpha=1.5), xi=0)
  z <- RFsimulate(model=model, 0:10, 0:10, n=4)
  plot(z)
```

BrownResnick *Brown-Resnick process*

Description

RPbrownresnick defines a Brown-Resnick process.

Usage

```
RPbrownresnick(phi, tcf, xi, mu, s)
```

Arguments

phi	specifies the covariance model or variogram, see RMmodel and RMmodelsAdvanced.
tcf	the extremal correlation function; either phi or tcf must be given.
xi, mu, s	the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.

Details

The extreme value index xi is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical value or given a RMmodel, in particular by a RMtrend model. The default values of mu and s are 1 and $z\xi$, respectively.

The functions RPbrorig, RPbrshifted and RPbrmixed perform the simulation of a Brown-Resnick process, which is defined by

$$Z(x) = \max_{i=1}^{\infty} X_i \exp(W_i(x) - \gamma^2),$$

where the X_i are the points of a Poisson point process on the positive real half-axis with intensity $x^{-2}dx$, $W_i \sim W$ are iid centered Gaussian processes with stationary increments and variogram γ given by model.

For simulation, internally, one of the methods RPbrorig, RPbrshifted and RPbrmixed is chosen automatically.

Note

Advanced options are maxpoints and max_gauss, see RFoptions.

Further advanced options related to the simulation methods RPbrorig, RPbrshifted and RPbrmixed can be found in the paragraph 'Specific method options for Brown-Resnick Fields' in RFoptions.

Author(s)

Marco Oesting, <oesting@math.uni-mannheim.de>, Martin Schlather, <schlather@math.uni-mannheim.de>
http://ms.math.uni-mannheim.de/de/publications/software

References

- Brown, B.M. and Resnick, S.I. (1977). Extreme values of independent stochastic processes. *J. Appl. Probab.* **14**, 732-739.
- Buishand, T., de Haan , L. and Zhou, C. (2008). On spatial extremes: With application to a rainfall problem. *Ann. Appl. Stat.* **2**, 624-642.
- Kabluchko, Z., Schlather, M. and de Haan, L (2009) Stationary max-stable random fields associated to negative definite functions *Ann. Probab.* **37**, 2042-2065.
- Oesting, M., Kabluchko, Z. and Schlather M. (2012) Simulation of Brown-Resnick Processes, *Extremes*, **15**, 89-107.

See Also

RPbrorig, RPbrshifted, RPbrmixed, RMmodel, RPgauss, maxstable, maxstableAdvanced

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again


## for some more sophisticated models see 'maxstamableAdvanced'
```

`Changings` *Documentation of some further changings*

Description

- Options getting obsolete
  - `oldstyle` is becoming `warn_oldstyle`
  - `newstyle` is becoming `warn_newstyke`
  - `newAniso` is becoming `warn_newAniso`
  - `ambiguous` is becoming `warn_ambiguous`
  - `normal_mode` is becoming `warn_normal_mode`
  - `colour_palette` is becoming `warn_colour_palette`
- **Changings in option names**
  - `pdfnumber` became in version 3.0.42 `filenumber`
  - `pdfonefile` became in version 3.0.42 `onefile`
  - `pdffile` became in version 3.0.42 `file`
  - `tbmdim` became in version 3.0.41 `reduceddim`
  - `coord_units` became in version 3.0.39 `coordunits`
  - `new_coord_units` became in version 3.0.39 `new_coordunits`
  - `variab_units` became in version 3.0.39 `varunits`

See Also

MajorRevisions, RandomFields

Examples

```
## no examples given
```

`Circulant Embedding` *Circulant Embedding methods*

Description

Circulant embedding is a fast simulation method for stationary (possibly anisotropic) Gaussian random fields on regular grids based on Fourier transformations. It is garantueed to be an exact method for covariance functions with finite support, e.g. the spherical model. The method is admissible for any dimension apart from memory restictions.
The simulation is performed on a torus which represents the bended grid. To remove wrong dependencies occuring at different borders of the grid which would be close on the torus, the simulation area is multiplied by a natural number. There is also a multivariate version of circulant embedding.

Cut-off embedding is a fast simulation method for stationary, isotropic Gaussian random fields on square lattices based on the standard `RPcirculant` method, so that exact simulation is garantueed for further covariance models, e.g. the `RMwhittle` model.

In fact, the circulant embedding is called with the cutoff hypermodel, see `RMcutoff`. `Cutoff` halfens the maximum number of elements models used to define the covariance function of interest (from 10 to 5).

Here multiplicative models are not allowed (yet).
For details see `RMcutoff`.

Intrinsic embedding is a fast simulation method for intrinsically stationary, isotropic Gaussian random fields on square lattices based on the standard `RPcirculant` method, for further *variogram* models, e.g. `RMfbm`.

Note that the simulated random field is always *non-stationary*. In fact, the circulant embedding is called with the Intrinsic hypermodel, see `RMintrinsic`.

Here multiplicative models are not allowed (yet).
For details see `RMintrinsic`.

Usage

```
RPcirculant(phi, force, mmin, strategy,
 maxGB, maxmem,  tolIm, tolRe, trials, useprimes, dependent,
 approx_step, approx_maxgrid)

RPcutoff(phi, force, mmin, strategy,
 maxGB, maxmem, tolIm, tolRe, trials, useprimes, dependent,
 approx_step, approx_maxgrid, diameter, a)
```

```
RPintrinsic(phi, force, mmin, strategy,
 maxGB, maxmem, tolIm, tolRe, trials, useprimes, dependent,
 approx_step, approx_maxgrid, diameter, rawR)
```

Arguments

`phi`	See RPgauss
`force`	Logical. Circulant embedding does not work if the constructed circulant matrix has negative eigenvalues. Sometimes it is convenient to replace all the negative eigenvalues by zero (`force=TRUE`) after `trials` number of trials. Default: `FALSE`
`mmin`	Scalar or vector, integer if positive. `CE.mmin` determines the initial size of the circulant matrix. If `CE.mmin=0` the minimal starting size is determined automatically according to the dimensions of the grid. If `CE.mmin>0` then the absolute starting size is given. If `CE.mmin<0` then the automatically determined matrix size is multiplied by $\lvert$`CE.mmin`$\rvert$; here `CE.mmin` must be smaller than -1; the value -1 takes over the minimal starting size. Note: in any cases, the initial size might be increased according to `CE.useprimes`. Default: `0`
`strategy`	Logical. `0`, if the circulant matrix has negative eigenvalues then the size in each direction is doubled; `1`: the size is enhanced only in one direction, namely that one where the covariance function has the largest value at the end point of the grid — note that the default value of `trials` is probably too small in that case. In some cases `strategy=0` works better, in other cases `strategy=1`. Just try. Clearly, if the field is isotropic and a square grid should be simulated, then `strategy=0` is the better choice. Default: `0`
`maxGB`	Maximal memory used for the circulant matrix in units of GB. If this argument is set then `maxmem` is set to MAXINT. Default: 0.5.
`maxmem`	Integer. maximal number of entries in a row of the circulant matrix. The total amount of memory needed for the internal calculations is is 32 (=4 * sizeof(double)) time as large (factor 2 is needed as complex numbers must be considered for calculating the fft of the covariance matrix; another factor 2 is needed for storing the simulated result). The value of `maxmem` must be at least 2^d times as large as the number of points to be simulated. Here d is the space dimension. In some cases even much larger. Note that `maxmem` can be used to control the automatic choice of the simulation algorithm. Namely, in case of huge circulant matrices, other simulation methods (TBM) might be faster and might be preferred by the user. If this argument is set then `maxGB` is set to `Inf`. Default: `MAXINT`

tolIm	If the modulus of the imaginary part is less than tolIm then the eigenvalue is always considered as real (independently of the value of force). Default: 1E-3
tolRe	Eigenvalues between tolRe and 0 are always considered as 0 and set 0 (independently of the value of force). Default: -1E-7
trials	Integer. A larger circulant matrix is likely to make more eigenvalues non-negative. If at least one of the thresholds tolRe and tolIm are missed then the matrix size is doubled according to strategy, and the matrix is checked again. This procedure is repeated up to trials - 1 times. If there are still negative eigenvalues, the simulation method fails if force=FALSE. Default: 3
useprimes	Logical. If FALSE the columns of the circulant matrix have length 2^k for some k. Otherwise the algorithm tries to find a nicely factorizable number close to the size of the given matrix. Default: TRUE
dependent	Logical. If FALSE then independent random fields are created. If TRUE then at least 4 non-overlapping rectangles are taken out of the the expanded grid defined by the circulant matrix. These simulations are dependent. See RFoptionsAdvanced for an example. See trials for some more information on the circulant matrix. Default: FALSE
approx_step	Real value. It gives the grid size of the approximating grid in case circulant embedding is used although the points do not ly on a grid. If NA then approx_step is chosen such that approx_maxgrid is nearly reached. Default: NA
approx_maxgrid	It defaults to maxmem.
diameter	See RMcutoff or RMintrinsic
a	See RMcutoff
rawR	See RMintrinsic

Details

Here, the algorithms by Dietrich and Newsam are implemented.

Value

an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Circulant Embedding

- Chan, G. and Wood, A.T.A. (1997) An Algorithm for Simulating Stationary Gaussian Random Fields. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **46**, 171–181.
- Dietrich, C. R. and G. N. Newsam (1993) A fast and exact method for multidimensional gaussian stochastic simulations. *Water Resour. Res.* **29(8)**, 2861–2869.
- Dietrich, C. R. and G. N. Newsam (1996) A fast and exact method for multidimensional Gaussian stochastic simulations: Extension to realizations conditioned on direct and indirect measurements *Water Resour. Res.* **32(6)**, 1643–1652.
- Dietrich, C. R. and Newsam, G. N. (1997) Fast and Exact Simulation of Stationary Gaussian Processes through Circulant Embedding of the Covariance Matrix. *SIAM J. Sci. Comput.* **18**, 1088–1107.
- Wood, A. T. A. and Chan, G. (1994) Simulation of Stationary Gaussian Processes in $[0, 1]^d$. *Journal of Computational and Graphical Statistics* **3**, 409–432.

Cutoff and Intrinsic

- Gneiting, T., Sevecikova, H, Percival, D.B., Schlather M., Jiang Y. (2006) Fast and Exact Simulation of Large Gaussian Lattice Systems in R^2: Exploring the Limits. *J. Comput. Graph. Stat.* **15**, 483–501.
- Stein, M.L. (2002) Fast and exact simulation of fractional Brownian surfaces. *J. Comput. Graph. Statist.* **11**, 587–599

See Also

RP, RPtbm

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMstable(s=1, alpha=1.8)
x <- seq(-3,3,0.1)
z <- RFsimulate(model=RPcirculant(model), x=x, y=x, n=1)
plot(z)

model <- RMexp(var=10, s=10)
z <- RFsimulate(model=RPcirculant(model), 1:10)
plot(z)

model <- RMfbm(Aniso=diag(c(1,2)), alpha=1.5)
z <- RFsimulate(model, x=1:10, y=1:10)
plot(z)
```

Coins *Random coin method*

Description

The random coin method (or dilution method) is simulation method for stationary Gaussian random fields. It is based on the following procedure: For a stationary Poisson point process on $\mathbf{R}^d$ consider the random field

$$Y(y) = \sum_{x \in X} f(y - x)$$

for a function f. The covariance of Y is proportional to the convolution

$$C(h) = \int f(x) f(x + h) dx$$

If the intensity of the Poisson point process increases, the random field Y is approaches a Gaussian random field with covariance function C.

Usage

```
RPcoins(phi, shape, intensity)

RPaverage(phi, shape, intensity)
```

Arguments

phi	object of class RMmodel; specifies the covariance function of the Poisson process; either phi or shape must be given.
shape	object of class RMmodel; specifies the function which is attached to the Poisson points; note that this is not the covariance function of the simulated random field.
intensity	positive number, intensity of the underlying Poisson point process.

Value

RPcoins returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Lantuejoul, C. (2002) *Geostatistical Simulation: Models and Algorithms.* Springer.

See Also

RFgetMethodNames RP, RPhyperplane, RPspectral, RPtbm.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

Constants *Constants used in RandomFields (RC constants)*

Description

Several constants are provided that might make the use of some functions easier, e.g. RFgetModelNames

Value

```
RC_TYPE = c("tail correlation function", "positive definite",    "negative definite", "process",    "me
RC_DOMAIN = c("single variable", "kernel", "framework dependent", "mismatch")
RC_ISOTROPY = c("isotropic", "space-isotropic", "zero-space-isotropic",    "vector-isotropic", "symme

RC_MONOTONE = c("mismatch in monotonicity",              "submodel dependent monotonicity",
```

RC_ISOTROPIC gives the numerical code for option "isotropic"

RC_SPACEISOTROPIC gives the numerical code for option "space-isotropic"

RC_CARTESIAN_COORD gives the numerical code for option "cartesian system"

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RF, RM, RP, RR, RFgetModelNames, RMmodelgenerator-class, RMtrafo

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

RC_ISOTROPY[1:5]
RFgetModelNames(isotropy=RC_ISOTROPY[1:5])
```

conventional2RFspDataFrame
Coercion to class 'RFsp' objects

Description

Generate an object of class RFsp from conventional objects

Usage

```
conventional2RFspDataFrame(data, coords=NULL, gridTopology=NULL, n=1,
                           vdim=1, vdim_close_together)
```

Arguments

data	array; of dimension c(vdim, space-time-dim, n); contains the values of the random fields
coords	matrix of coordinates
gridTopology	3-row-matrix or of class GridTopology; specifies the grid vectors; either coords or gridTopology must be NULL
n	number of iid copies of the random fields, default is 1
vdim	number of dimensions of the values of the random field, default is 1
vdim_close_together	logical. Currently only vdim_close_together=FALSE is coded. In this case the dimensions of the data follow the order "locations, multivariate, repeated". Otherwise "multivariate, locations, repeated".

Value

object of class RFspatialGridDataFrame, RFspatialPointsDataFrame, RFgridDataFrame or RFpointsDataFrame

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- 1:20
z <- RFsimulate(RMexp(), x, spConform=FALSE)
z2 <- conventional2RFspDataFrame(z, coord=x)
Print(z, z2)
```

`Distribution Families` *Distribution families (RR commands)*

Description

Distribution families to specify random parameters in the model definition.

Details

When simulating Gaussian random fields, the random parameters are drawn only once at the very beginning. So, if the argument `n` in `RFsimulate` is greater than 1 then `n` simulations conditional on a single realisation of the random parameters are performed. See the examples below.

There are (simple) multivariate version and additional version to the distributions families implemented. Further, **any** distribution family defined in R can be used, see the examples below

These function will allow for Baysian modelling. (Future project).

Implemented models

`RRdeterm`	no scattering
`RRdistr`	families of distributions transferred from R
`RRgauss`	a (multivariate) Gaussian random variable
`RRloc`	modification of location and scale
`RRspheric`	random scale for the `RMball` to simulate `RMspheric`, etc.
`RRunif`	a (multivariate) uniform random variable

Note

The allowance of random parameters is a very recent, developing feature of **RandomField**.

Future changings of the behaviour are not unlikely.

Note

A further random element is `RMsign`, which is an operator on shape functions. As an exception its name starts with `RM` and not with `RR`.

Author(s)

Martin Schlather, `<schlather@math.uni-mannheim.de>` `http://ms.math.uni-mannheim.de/de/publications/software`

See Also

RC, RF, RM, RP, `Other models`, `RFdistr`, `RMmodelgenerator`,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## here, the scale is given by an exponential variable:
model <- RMgauss(scale=exp())
for (i in 1:4) {
  RFoptions(seed = i)
  # each leads to a simulation with a different scale parameter
  plot(model) ## random
  plot(RFsimulate(model, x=seq(0,10,0.1)))
  readline("press return")
}

# but here, all 4 simulations have same (but random) scale:
plot(RFsimulate(model, x=seq(0,10,0.1), n=4))


## hierarchical models are also possible:
## here, the scale is given by an exponential variable whose
## rate is given by a uniform variable
model <- RMgauss(scale=exp(rate=unif()))
plot(model)
plot(RFsimulate(model, x=seq(0,10,0.1)))


## HOWEVER, the next model is deterministic with scale \eqn{e=2.718282}.
model <- RMgauss(scale=exp(1))
plot(model)
plot(RFsimulate(model, x=seq(0,10,0.1)))
```

`Extremal t` *Extremal t process*

Description

`RPopitz` defines an extremal t process.

Usage

```
RPopitz(phi, xi, mu, s, alpha)
```

Arguments

`phi`	an `RMmodel`; covariance model for a standardized Gaussian random fields, or the field itself.

xi,mu,s	the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.
alpha	originally referred to the α-Frechet marginal distribution, see the original literature for details.

Details

The argument xi is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical value or given a RMmodel, in particular by a RMtrend model. The default values of mu and s are 1 and $z\xi$, respectively.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Davison, A.C., Padoan, S., Ribatet, M. (2012). Statistical modelling of spatial extremes. *Stat. Science* **27**, 161-186.
- Opitz, T. (2012) A spectral construction of the extremal t process. *arxiv* **1207.2296**.

See Also

RMmodel, RPgauss, maxstable, maxstableAdvanced

Examples

```
RFoptions(seed=0, xi=0)
## seed=0: *ANY* simulation will have the random seed 0; set
##         RFoptions(seed=NA) to make them all random again
## xi=0: any simulated max-staable random field has extreme value index 0

x <- seq(0, 2, if (interactive()) 0.01 else 1)
model <- RPopitz(RMgauss(), alpha=2)
z1 <- RFsimulate(model, x)
plot(z1, type="l")
```

ExtremalGaussian *Extremal Gaussian process*

Description

RPschlather defines an extremal Gaussian process.

Usage

```
RPschlather(phi, tcf, xi, mu, s)
```

Arguments

phi	an RMmodel, see Details.
tcf	an RMmodel specifying the extremal correlation function; either phi or tcf must be given.
xi,mu,s	the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.

Details

The argument xi is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical value or given a RMmodel, in particular by a RMtrend model. The default values of mu and s are 1 and $z\xi$, respectively.

The argument phi can be any random field for which the expectation of the positive part is known at the origin.

It simulates Extremal Gaussian process Z (also called "Schlather model"), which is defined by

$$Z(x) = \max_{i=1}^{\infty} X_i \max(0, Y_i(x)),$$

where the X_i are the points of a Poisson point process on the positive real half-axis with intensity $cx^{-2}dx$, $Y_i \sim Y$ are iid stationary Gaussian processes with a covariance function given by model, and c is chosen such that Z has standard Frechet margins. model must represent a stationary covariance model.

Note

Advanced options are maxpoints and max_gauss, see RFoptions.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RMmodel, RPgauss, maxstable, maxstableAdvanced

Examples

```
RFoptions(seed=0, xi=0)
## seed=0: *ANY* simulation will have the random seed 0; set
##         RFoptions(seed=NA) to make them all random again
## xi=0: any simulated max-staable random field has extreme value index 0

x <- seq(0, 2, if (interactive()) 0.01 else 1)
```

```
## standard use of RPschlather (i.e. a standardized Gaussian field)
model <- RMgauss()
z1 <- RFsimulate(RPschlather(model), x)
plot(z1, type="l")


## the following refers to the generalized use of RPschlather, where
## any random field can be used. Note that 'z1' and 'z2' have the same
## margins and the same .Random.seed (and the same simulation method),
## hence the same values
model <- RPgauss(RMgauss(var=2))
z2 <- RFsimulate(RPschlather(model), x)
plot(z2, type="l")
all.equal(z1, z2) # true

## Note that the the following defintion is incorrect
try(RFsimulate(model=RPschlather(RMgauss(var=2)), x=x))


## check whether the marginal distribution (Gumbel) is indeed correct:
model <- RMgauss()
n <- if (interactive()) 100 else 1
z <- RFsimulate(RPschlather(model, xi=0), x, n=n)
plot(z)
hist(unlist(z@data), 50, freq=FALSE)
curve(exp(-x) * exp(-exp(-x)), from=-3, to=8, add=TRUE)
```

`fitgauss` *Details on fitting Gaussian random fields*

Description

Here some details of `RFfit` are given concerning the fitting of models for Gaussian random fields

This documentation is far from being complete.

Maximum likelihood

The application of the usual maximum likelihood method and reporting the result is the default.

Least squares

The weighted least squares methods minimise

$$\sum_i w_i(\hat{\gamma}(h_i) - \gamma(h_i))^2$$

over all parametrised models of γ. Here, i runs over all N bins of the binned variogram $\hat{\gamma}$ and h_i is the centre of bin i.

The following variants of the least squares methods, passed as `sub.methods` in `RFfit` are implemented.

`'self'` $w_i = (\gamma(h_i))^{-2}$

`'plain'` $w_i = 1$ for all i.

`'sqrt.nr'` w_i^2 equals the number of points n_i in bin i.

`'sd.inv'` $1/w_i$ equals the standard deviation of the variogram cloud within bin i.

`'internal'` Three subvariants are implemented:

- `'internal1'` $w_i^2 = (N - i + 1)n_i$
- `'internal2'` $w_i = N - i + 1$
- `'internal3'` $w_i^2 = N - i + 1$

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

`RFfit`, `RFfit-class`

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
## see 'RFfit'
```

`GaussianFields` *Methods for Gaussian Random Fields*

Description

Hier all the methods (models) for simulating Gaussian random fields are listed

Implemented models

`RPcirculant`	simulation by circulant embedding
`RPcutoff`	simulation by a variant of circulant embedding
`RPcoins`	simulation by random coin / shot noise
`RPgauss`	generic model that chooses automatically among the specific methods
`RPhyperplane`	simulation by hyperplane tessellation
`RPintrinsic`	simulation by a variant of circulant embedding
`RPnugget`	simulation of (anisotropic) nugget effects

RPsequential	sequential method
RPspecific	model specific methods (very advanced)
RPspectral	spectral method
RPtbm	turning bands

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.
- Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.
- Schlather, M. (2010) On some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.
- Schlather, M. (2011) Construction of covariance functions and unconditional simulation of random fields. In Porcu, E., Montero, J.M. and Schlather, M., *Space-Time Processes and Challenges Related to Environmental Problems.* New York: Springer.
- Yaglom, A.M. (1987) *Correlation Theory of Stationary and Related Random Functions I, Basic Results.* New York: Springer.
- Wackernagel, H. (2003) *Multivariate Geostatistics.* Berlin: Springer, 3nd edition.

See Also

RP, Other models, RMmodel, RFsimulateAdvanced

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

x <- seq(0, 10, 0.01)
z <- RFsimulate(RMexp(), x)
RFgetModelInfo(RFsimulate, level=0, which="internal")
# i.e., circulant embedding has been chosen
```

GSPSJ06 *Fast and Exact Simulation of Large Gaussian Lattice Systems in R2*

Description

Here the code of the paper on 'Fast and Exact Simulation of Large Gaussian Lattice Systems in R2' is given.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software;

References

- Gneiting, T., Sevcikova, H., Percival, D.B., Schlather, M., Jiang, Y. (2006) Fast and Exact Simulation of Large Gaussian Lattice Systems in R2: Exploring the Limits. *J. Comput. Graph. Stat.*, **15**, 483-501.

Examples

```
## Figure 1
stabletest <- function(alpha, theta, size=if (interactive()) 512 else 20) {
  RFoptions(trials=1, tolIm = 1e-8, tolRe=0, force = FALSE,
            useprimes=TRUE, strategy=0, skipchecks=!FALSE,
             storing=TRUE)
  model <- RMcutoff(diameter=theta, a=1, RMstable(alpha=alpha))
  RFcov(dist=0, model=model, dim=2, seed=0)
  r <- RFgetModelInfo(RFcov, modelname="RMcutoff", level=3)$internalq[5] # theor R
  x <- seq(0, r, by= r / (size - 1)) * theta
  err <- try(RFsimulate(x, x, model=RPcirculant(model), n=0))
  return(if (class(err) == "try-error") NA else r)
}

alphas <- seq(1.52, 2.0, if (interactive()) 0.02 else 0.5)
thetas <-
  if (interactive()) seq(0.05, 3.5, 0.05) else seq(0.1, 3.5, 2)

m <- matrix(NA, nrow=length(thetas), ncol=length(alphas))
for (it in 1:length(thetas)) {
  theta <- thetas[it]
  for (ia in 1:length(alphas)) {
  alpha <- alphas[ia]
  cat("alpha=", alpha, "theta=", theta,"\n")
  m[it, ia] <- stabletest(alpha=alpha, theta=theta)
  if (is.na(m[it, ia])) break
  }
  if (any(is.finite(m))) image(thetas, alphas, m, col=rainbow(100))
}
```

`Hyperplane` *Hyperplane method*

Description

The Hyperplane method is a simulation method for stationary, isotropic random fields with exponential covariance function. It is based on a tessellation of the space by hyperplanes. Each cell takes a spatially constant value of an i.i.d. random variable. The superposition of several such random fields yields approximatively a Gaussian random field.

Usage

```
RPhyperplane(phi, superpos, maxlines, mar_distr, mar_param)
```

Arguments

`phi`	object of class `RMmodel`; specifies the covariance function to be simulated; only exponential covariance functions and scale mixtures of it are allowed.
`superpos`	integer. number of superposed hyperplane tessellations. Default: `300`.
`maxlines`	integer. Maximum number of allowed lines. Default: `1000`.

`mar_distr`	integer. code for the marginal distribution used in the simulation:
	`0` uniform distribution
	`1` Frechet distribution with form parameter `mar_param`
	`2` Bernoulli distribution (Binomial with $n = 1$) with argument `mar_param`
	This argument should not be changed yet.
	Default: `0`.

`mar_param`	Argument used for the marginal distribution. The argument should not be changed yet. Default: `NA`.

Value

`RPhyperplane` returns an object of class `RMmodel`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Lantuejoul, C. (2002) *Geostatistical Simulation: Models and Algorithms.* Springer.

See Also

RP, RPcoins, RPspectral, RPtbm.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RPhyperplane(RMexp(s=2), superpos=1)
x <- seq(0, 3, if (interactive()) 0.04 else 1)
z <- RFsimulate(x=x, x, model=model, n=1)
plot(z)
```

Independent Variables *Method to simulate the Nugget effect*

Description

Method to simulate the Nugget effect.

Usage

```
RPnugget(phi, tol, vdim)
```

Arguments

phi	object of class RMmodel; specifies the covariance model to be simulated. The only possible model for phi is RMnugget.
tol	points at a distance less than or equal to nugget.tol are considered as being identical. This strategy applies to the simulation method and the covariance function itself. Hence, the covariance function is only positive definite if nugget.tol=0.0. However, if the anisotropy matrix does not have full rank and nugget.tol=0.0 then, the simulations are likely to be odd. The value of nugget.tol should be of order 10^{-15}. Default: 0.0
vdim	positive integer; the model is treated vdim-variate, vdim=1 (default) corresponds to a univariate random field. Mostly, the value of vdim is set automatically. Default is that it takes the value of the submodel phi

Details

This method only allows RMnugget as a submodel.

The method also allows for zonal nugget effects. Only there the argument tol becomes important. For the zonal nugget effect, the anisotropy matrix Aniso should be given in RMnugget. There, only the kernal of the matrix is important.

Value

RPnugget returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

See Also

RP, RPcoins, RPhyperplane, RPspectral, RPtbm.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMnugget()
z <- RFsimulate(model=model, 0:10, 0:10, n=4)
plot(z)

model <- RPnugget(RMnugget(var=0.01, Aniso=matrix(nc=2, rep(1,4))))
z <- RFsimulate(model=model, 0:10, 0:10, n=4)
plot(z)
```

Internal functions *Internal functions*

Description

These functions are internal and should not be used.

Usage

```
rfGenerateModels(assigning,
 RFpath = "~/R/RF/svn/RandomFields",
 RMmodels.file = paste(RFpath, "R/RMmodels.R", sep="/")
 )

checkExamples(exclude = NULL, include=1:length(.fct.list),
               ask=FALSE, echo=TRUE, halt=FALSE, ignore.all = FALSE,
               path="RandomFields", package = "RandomFields",
                          read.rd.files=TRUE)
```

```
ScreenDevice(height, width)

FinalizeExample()

Dependencies(install = length(pkgs) ==
                     length(c(depends, imports, suggests)) &&
                     all(pkgs == c(depends, imports, suggests)),
             check=TRUE,
             pkgs = c(depends, imports, suggests),
             lib.loc ="/usr/local/lib64/R/library")

showManpages(path="/home/schlather/svn/RandomFields/RandomFields/man")

plotWithCircles(data, factor=1.0,
                xlim=range(data[,1])+c(-maxr,maxr),
                ylim=range(data[,2])+c(-maxr,maxr),
                col=1, fill=0, ...)
```

Arguments

`assigning, RFpath, RMmodels.file`	internal
`exclude, include, ask, echo, halt, ignore.all, path, package, read.rd.files`	internal; ignore.all referes to the 'all' export statement in the namespace – whether this should be ignored. if `read.rf.files` is `TRUE` or a path to the Rd files, then the man pages are analysed to get all examples; `ignore.all` is then ignored. If `FALSE` only examples of functions (which are search in the environments) are run.
`install, check, pkgs, lib.loc`	internal
`height,width`	window sizes
`data, factor, xlim, ylim, col, fill, ...`	internal

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## internal functions: no examples given


# for (i in dep.packages) cat(maintainer(i), "\n")
```

jss14 *Covariance models for multivariate and vector valued fields*

Description

Here the code of the paper on 'Analysis, simulation and prediction of multivariate random fields with package RandomFields'

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software;

References

- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Strokorb, K. (2014) Analysis, simulation and prediction of multivariate random fields with package **RandomFields** *JSS Submitted*

See Also

weather, SS12, S10

Examples

```
# This skript is based on version 3.0.28 available at
# http://ms.math.uni-mannheim.de/de/publications/software/



              ##########################################
              ##  SECTION 4: UNCONDITIONAL SIMULATION  ##
              ##########################################

RFoptions(seed = 0, height = 4, always_close_screen = TRUE)
## seed=0:  *ANY* simulation will have the random seed 0; set
##          RFoptions(seed=NA) to make them all random again
## height : height of X11
## always_close_screen=FALSE: the pictures are kept on the screen


## Fig. 1: linear model of coregionalization
M1 <- c(0.9, 0.6)
M2 <- c(sqrt(0.19), 0.8)
```

```
model <- RMmatrix(M = M1, RMwhittle(nu = 0.3)) +
         RMmatrix(M = M2, RMwhittle(nu = 2))
x <- y <- seq(-10, 10, 0.2)
simu <- RFsimulate(model, x, y)
plot(simu)


## Fig. 2: Wackernagel's delay model
model <- RMdelay(RMstable(alpha = 1.9, scale = 2), s = c(4, 4))
simu <- RFsimulate(model, x, y)
plot(simu, zlim = 'joint')


## Fig. 3: extended Wackernagel's delay model
model <- RMdelay(RMstable(alpha = 1.9, scale = 2), s = c(0, 4)) +
         RMdelay(RMstable(alpha = 1.9, scale = 2), s = c(4, 0))
simu <- RFsimulate(model, x, y)
plot(simu, zlim = 'joint')
plot(model, dim = 2, xlim = c(-5, 5), main = "Covariance function",
     cex = 1.5, col = "brown")


## Fig. 4: latent dimension model
## MARGIN.slices has the effect of choosing the third dimension
##               as latend dimension
## n.slices has the effect of choosing a bivariate model
model <- RMgencauchy(alpha = 1.5, beta = 3)
simu <- RFsimulate(model, x, y, z = c(0,1))
plot(simu, MARGIN.slices = 3, n.slices = 2)


## Fig. 5: Gneiting's bivariate Whittle-Matern model
model <- RMbiwm(nudiag = c(1, 2), nured = 1, rhored = 1, cdiag = c(1, 5),
                s = c(1, 1, 2))
simu <- RFsimulate(model, x, y)
plot(simu)


## Fig. 6: anisotropic linear model of coregionalisaton
M1 <- c(0.9, 0.6)
M2 <- c(sqrt(0.19), 0.8)
A1 <- RMangle(angle = pi/4, diag = c(0.1, 0.5))
A2 <- RMangle(angle = 0, diag = c(0.1, 0.5))
model <- RMmatrix(M = M1, RMgengneiting(kappa = 0, mu = 2, Aniso = A1)) +
         RMmatrix(M = M2, RMgengneiting(kappa = 3, mu = 2, Aniso = A2))
x <- y <- seq(-10, 10, 0.2)
simu <- RFsimulate(model, x, y)
plot(simu)


## Fig. 7: random vector field whose path are curl free
## A 4-variate field is simulated, where the first component
## refers to the potential field, the second and third component
```

```
## to the curl free vector field and the forth component to the
## field of sinks and sources
model <- RMcurlfree(RMmatern(nu = 5), scale = 4)
simu <- RFsimulate(model, x, y)
plot(simu, select.variables = list(1, 2 : 3, 4))
plot(model, dim = 2, xlim = c(-3, 3), main = "", cex = 2.3, col="brown")



## Fig. 8: Kolmogorov's model of turbulence
## See the physical literature for its derivation and details
x <- y <- seq(-2, 2, len = 20)
model <- RMkolmogorov()
simu <- RFsimulate(model, x, y, z = 1)
plot(simu, select.variables = list(1 : 2), col = c("red"))
plot(model, dim = 3, xlim = c(-3, 3), MARGIN = 1 : 2, cex = 2.3,
     fixed.MARGIN = 1.0, main = "", col = "brown")




              ##########################################
              ## SECTION 5: DATA ANALYSIS              ##
              ##########################################

## get the data
data("weather")
PT <- weather[ , 1 : 2]
## transformation of earth coordinates to Euclidean coordinates
Dist.mat <- as.vector(RFearth2dist(weather[ , 3 : 4]))


#################################
## model definition            ##
#################################
## bivariate pure nugget effect:
nug <- RMmatrix(M = matrix(nc = 2, c(NA, 0, 0, NA)), RMnugget())
## parsimonious bivariate Matern model
pars.model <- nug + RMbiwm(nudiag = c(NA, NA), scale = NA, cdiag = c(NA, NA),
                           rhored = NA)
## whole bivariate Matern model
whole.model <- nug + RMbiwm(nudiag = c(NA, NA), nured = NA, s = rep(NA, 3),
                            cdiag = c(NA, NA), rhored = NA)



#################################
## model fitting and testing   ##
#################################
## 'parsimoneous model'
## fitting takes a while; 'pars' is already available within 'data(weather)'
pars <- RFfit(pars.model, distances = Dist.mat, dim = 3, data = PT)
```

```
print(pars)
print(pars, full=TRUE)
RFratiotest(pars)
RFcrossvalidate(pars, x = weather[ , 3 : 4], data = PT, full = TRUE)

## 'whole model'
## fitting takes a while; 'whole' is already available within 'data(weather)'
whole <- RFfit(whole.model, distances = Dist.mat, dim = 3, data = PT)
print(whole, full=TRUE)
RFratiotest(whole)
RFcrossvalidate(whole, x = weather[ , 3 : 4], data = PT, full = TRUE)

## compare parsimonous and whole
RFratiotest(nullmodel = pars, alternative = whole)


################################
## kriging                    ##
################################
## First the coordindates are projected on a plane
a <- colMeans(weather[ , 3 : 4]) * pi / 180
P <- cbind(c(-sin(a[1]), cos(a[1]), 0),
           c(-cos(a[1]) * sin(a[2]), -sin(a[1]) * sin(a[2]), cos(a[2])),
           c(cos(a[1]) * cos(a[2]), sin(a[1]) * cos(a[2]), sin(a[2])))
x <- RFearth2cartesian(weather[ , 3 : 4])
plane <- (x %*%P)[ , 1 : 2]

## here, kriging is performed on a rectangular that covers the
## the projected points above. The rectangular is approximated
## by a grid of length 200 in each direction.
n <- 200
r <- apply(plane, 2, range)
data <- cbind(plane, weather[ , 1 : 2])
z <- RFinterpolate(pars, data = data, dim = 2,
                   x = seq(r[1, 1], r[2, 1], length = n),
                   y = seq(r[1, 2], r[2, 2], length = n),
                   varunits = c("Pa", "K"), spConform = TRUE)
plot(z)
```

Major Revisions *Documentation of major changings*

Description

Major Revision: changings from Version 2 to Version 3

- **S4 objects**
  - **RandomFields** is now based on S4 objects using the package **sp**. The functions accept both **sp** objects and simple objects as used in version 2. See also above.
- **Documentation**
  - each model has now its own man page;
  - classes of models and functions are bundled in several pages: Covariance models start with `RM`, distribution families with `RR`, processes with `RP`, user functions with `RF`
  - the man pages of several functions are split into two parts:
    (i) a beginners man page which includes a link to
    (ii) man pages for advanced users
- **Interfaces**
  - The interfaces become simpler, at the same time more powerful then the functions in version 2. E.g., `RFsimulate` can perform unconditional simulation, conditional simulation and random imputing.
  - Only those arguments are kept in the functions that are considered as being absolutely necessary. All the other arguments can be included as options.
  - `RFgui` is an instructive interface based on tcl/tk, replacing the former `ShowModels`
- **Inference for Gaussian random fields**
  - `RFfit` has undergone a major revision. E.g.:
    (i) estimation random effects model with spatial covariance structure
    (ii) automatic estimation of 10 and more arguments in multivariate and/or space-time models
  - `RFempiricalvariogram` is now based on an fft algorithm if the data are on a grid, even allowing for missing values.
  - `RFratiotest` has been added.
- **Processes**
  - Maxstable processes modelling of maxstable processes has been enhanced, including
    (i) the simulation of Brown-Resnick processes
    (ii) initial support of tail correlation functions;
  - Further processes chi2 processes, compound Poisson processes, binary processes added.
- **Models**
  - the formula notation for linear models may now be defined
  - Novel, user friendly definition of the covariance models
  - Multivariate and vector valued random fields are now fully included
  - The user may now define his own functions, to some extend.
  - The trend allows for much more flexibility
  - Distributions may now included which will be extended to Baysian modelling in future.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## S4 vs S3
x <- seq(0, 10, if (interactive()) 0.1 else 2)
model <- RMexp()
plot(RFsimulate(model, x)) ## S4
plot(RFsimulate(model, x, spConform=FALSE)) ## no class
```

`Max-stable random fields`

Simulation of Max-Stable Random Fields

Description

Here, a list of models and methods for simulating max-stable random fields is given.

See also maxstableAdvanced for more advanced examples.

Implemeted models and methods

Models

RPbrownresnick	Brown-Resnick process using an automatic choice of the below 3 RPbr* methods
RPopitz	extremal t process
RPschlather	extremal Gaussian process
RPsmith	M3 processes

Methods

RPbrmixed	simulation of Brown-Resnick processes using M3 representation
RPbrorig	simulation of Brown-Resnick processes using the original definition
RPbrshifted	simulation of Brown-Resnick processes using a random shift

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Kabluchko, Z., Schlather, M. & de Haan, L (2009) Stationary max-stable random fields associated to negative definite functions *Ann. Probab.* **37**, 2042-2065.
- Schlather, M. (2002) Models for stationary max-stable random fields. *Extremes* **5**, 33-44.
- Smith, R.L. (1990) Max-stable processes and spatial extremes Unpublished Manuscript.

See Also

RP, RMmodel, RPgauss, RPbernoulli maxstableAdvanced

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMfbm(alpha=1.5)
x <-  seq(0, 8, if (interactive()) 0.02 else 4)
z <- RFsimulate(RPbrownresnick(model), x=x, n=if (interactive()) 4 else 1)
plot(z)
## Not run: \dontshow{
## to do : seq(0, 10, 0.02) oben ist furchtbar langsam. Warum?
}
## End(Not run)

## Not run: \dontshow{
model <- RMball()
x <- seq(0, 10, 5) # nice for   x <- seq(0, 10, 0.02)
n <- if (interactive()) 1000 else 1
z <- RFsimulate(RPsmith(model, xi=0), x, n=n, every=1000)
plot(z)
hist(unlist(z@data), 150, freq=FALSE) #not correct; to do; sqrt(2) wrong
curve(exp(-x) * exp(-exp(-x)), from=-3, to=8, add=TRUE, col=3)
}
## End(Not run)

model <- RMgauss()
n <- if (interactive()) 100 else 1
x <- seq(0, 10, if (interactive()) 0.05 else 5)
z <- RFsimulate(RPschlather(model, xi=0), x, n=n)
plot(z)
hist(unlist(z@data), 50, freq=FALSE)
curve(exp(-x) * exp(-exp(-x)), from=-3, to=8, add=TRUE)


## for some more sophisticated models see maxstableAdvanced
```

Obsolete Functions *Obsolete functions*

Description

This part gives the obsolete functions of RandomFields Version 2 that are **not maintained** anymore.

Usage

```
Covariance(x, y = NULL, model, param = NULL, dim = if
 (!missing(Distances)) { if (is.matrix(x)) ncol(x) else 1},
 Distances, fctcall = c("Cov", "Variogram", "CovMatrix"))
CovarianceFct(x, y = NULL, model, param = NULL, dim = if
 (!missing(Distances)) { if (is.matrix(x)) ncol(x) else 1},
 Distances, fctcall = c("Cov", "Variogram", "CovMatrix"))
CovMatrix(x, y = NULL, model, param = NULL, dim = if
 (!missing(Distances)) { if (is.matrix(x)) ncol(x) else 1}, Distances)
DeleteAllRegisters()
DeleteRegister(nr=0)
DoSimulateRF(n = 1, register = 0, paired=FALSE, trend=NULL)
InitSimulateRF(x, y = NULL, z = NULL, T=NULL, grid=!missing(gridtriple),
 model, param, trend, method = NULL, register = 0,
 gridtriple, distribution=NA)
InitGaussRF(x, y = NULL, z = NULL, T=NULL, grid=!missing(gridtriple),
 model, param, trend=NULL, method = NULL, register = 0, gridtriple)
GaussRF(x, y = NULL, z = NULL, T=NULL, grid=!missing(gridtriple), model,
 param, trend=NULL, method = NULL, n = 1, register = 0, gridtriple,
 paired=FALSE, PrintLevel=1, Storing=TRUE, ...)
Variogram(x, model, param = NULL, dim = if (!missing(Distances))
 { if (is.matrix(x)) ncol(x) else 1}, Distances)
InitMaxStableRF(x, y = NULL, z = NULL, grid, model, param, maxstable,
 method = NULL, register = 0, gridtriple = FALSE)
MaxStableRF(x, y = NULL, z = NULL, grid, model, param, maxstable,
 method = NULL, n = 1, register = 0, gridtriple = FALSE, ...)
EmpiricalVariogram(x, y = NULL, z = NULL, T=NULL, data, grid, bin,
 gridtriple = FALSE, phi, theta, deltaT)
Kriging(krige.method, x, y=NULL, z=NULL, T=NULL, grid, gridtriple=FALSE,
 model, param, given, data, trend=NULL,pch=".", return.variance=FALSE,
 allowdistanceZero = FALSE, cholesky=FALSE)
CondSimu(krige.method, x, y=NULL, z=NULL, T=NULL, grid, gridtriple=FALSE,
 model, param, method=NULL, given, data, trend=NULL, n=1, register=0,
 err.model=NULL, err.param=NULL, err.method=NULL, err.register=1,
 tol=1E-5, pch=".", paired=FALSE, na.rm=FALSE)
RFparameters(...)
hurst(x, y = NULL, z = NULL, data, gridtriple = FALSE, sort=TRUE,
 block.sequ = unique(round(exp(seq(log(min(3000, dim[1] / 5)),
 log(dim[1]), len=min(100, dim[1]))))),
```

```
 fft.m = c(1, min(1000, (fft.len - 1) / 10)),
 fft.max.length = Inf,
 method=c("dfa", "fft", "var"), mode=c("plot", "interactive"),
 pch=16, cex=0.2, cex.main=0.85,
 PrintLevel=RFoptions()$general$printlevel,height=3.5, ...)
fractal.dim(x, y = NULL, z = NULL, data, grid=TRUE, gridtriple = FALSE,
 bin, vario.n=5, sort=TRUE, fft.m = c(65, 86), fft.max.length=Inf,
 fft.max.regr=150000, fft.shift = 50, method=c("variogram", "fft"),
 mode=c("plot", "interactive"), pch=16, cex=0.2, cex.main=0.85,
 PrintLevel = RFoptions()$general$printlevel, height=3.5, ...)
fitvario(x, y=NULL, z=NULL, T=NULL, data, model, param, lower=NULL,
 upper=NULL, sill=NA, grid=!missing(gridtriple), gridtriple=FALSE, ...)
```

Arguments

`x, y, model, param, dim, Distances, fctcall, n, register, paired, trend, z, T, grid, method, gridtr` as the functions are obsolete, all these arguments are not documented anymore.

Value

See 'version2' for details on these obsolete commands.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

The functions that should be used instead are, for instance, RFcov, RFcovmatrix, RFvariogram, RFsimulate, RFinterpolate, RFempiricalvariogram, RFfit, RFoptions, RFhurst, RFfractaldim

See 'version2' for details on the obsolete commands.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

```
## no examples given, as command is obsolete
```

Others *Auxiliary and other Models*

Description

Here, auxiliary models are given that are not covariance functions or variograms, but which might be used within the definition of a model.

Implemented models

Distribution families See RR.

Evaluation operators See RF.

Random Fields / Random Processes See RP.

Shape functions

Besides any of the covariance functions the following functions can be used as shape functions.

RMangle	defines an anisotropy matrix by angle and a diagonal matrix.
RMball	Indicator of a ball of radius $1/2$.
RMm2r	spectral function belonging to a tail correlation function of the Gneiting class H_n
RMm3b	spectral function belonging to a tail correlation function of the Gneiting class H_n
RMmppplus	operator to define mixed moving maxima (M3) processes
RMmps	spectral functions belonging to a tail correlation function of the Gneiting class H_n
RMpolygon	Indicator of a typical Poisson polygon.
RMrational	shape function used in the Bernoulli paper given in the references
RMrotat	shape function used in the Bernoulli paper given in the references
RMsign	random sign
RMtruncsupport	truncates the support of a shape in a Poisson based model

Special transformations

RMeaxxa	shape function used in the Bernoulli paper given in the references
RMetaxxa	shape function used in the Bernoulli paper given in the references
RMid	identity
RMrotation	shape function used in the Bernoulli paper given in the references

Other models

RMcoord	passing new coordinates in a mixed model
RMuser	User defined covariance model

Author(s)

Alexander Malinowski, <malinowski@math.uni-mannheim.de>

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RFformula, RMmodels, RMmodelsAdvanced

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
(type <- RC_TYPE[(0:1) -length(RC_TYPE)])
RFgetModelNames(type=type, group.by="type")
```

---

papers *Papers involving* **RandomFields** *and co-authored by M. Schlather*

Description

Here, an overview is given over the papers co-authored by M. Schlather that involve **RandomFields**.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software;

References

- Gneiting, T., Kleiber, W. and Schlather, M. (2010) Matern cross-covariance functions for multivariate random fields *J. Amer. Statist. Assoc.* **105**, 1167-1177.

  See GKS11 for the code.
- Gneiting, T., Sevcikova, H., Percival, D.B., Schlather, M., Jiang, Y. (2006) Fast and Exact Simulation of Large Gaussian Lattice Systems in R2: Exploring the Limits. *J. Comput. Graph. Stat.*, **15**, 483-501.

  See GSPSJ06 for the code.
- Scheuerer, M. and Schlather, M. (2012) Covariance Models for Random Vector Fields. *Stochastic Models*, **82**, 433-451.

  See SS12 for the code.
- Schlather, M. (2002) Models for stationary max-stable random fields. *Extremes* **5**, 33-44.

  See S02 for the code.
- Schlather, M. (2010) On some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

  See S10 for the code.

- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Strokorb, K. (2014) Analysis, simulation and prediction of multivariate Random Fields with package **RandomFields**. *Submitted to JSS*

  See 'multivariate_jss' for the vignette.
- Strokorb, K., Ballani, F. and Schlather, M. (2014) In Preparation.

  See SBS14 for the code.

See Also

weather, GSPSJ06, SS12, S02, S10, jss14

Examples

```
# for examples, see the specific papers.
```

`plot-method` *Methods for function* `plot` *in package* **RandomFields**

Description

Plot methods are implemented for simulated random fields (objects of class `RFsp`), explicit covariance models (objects of class `RMmodel`), empirical variograms (objects of class `RFempVariog`) and fitted models (objects of class `RFfit`).

Usage

```
## S4 method for signature 'RFspatialDataFrame,missing'
plot(x, y,
 MARGIN=c(1,2), MARGIN.slices=NULL,
 n.slices = if (is.null(MARGIN.slices)) 1 else 10,
 nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), select.variables, zlim, legend=TRUE,
 MARGIN.movie = NULL, ...)

## S4 method for signature 'RFspatialDataFrame,RFspatialGridDataFrame'
plot(x, y,
 MARGIN=c(1,2), MARGIN.slices=NULL,
 n.slices = if (is.null(MARGIN.slices)) 1 else 10,
 nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), select.variables, zlim, legend=TRUE,
  MARGIN.movie = NULL,...)

## S4 method for signature 'RFspatialDataFrame,RFspatialPointsDataFrame'
```

```
plot(x, y,
 MARGIN=c(1,2), MARGIN.slices=NULL,
 n.slices = if (is.null(MARGIN.slices)) 1 else 10,
 nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), select.variables, zlim, legend=TRUE,
  MARGIN.movie = NULL,...)

## S4 method for signature 'RFspatialDataFrame,matrix'
plot(x, y,
 MARGIN=c(1,2), MARGIN.slices=NULL,
 n.slices = if (is.null(MARGIN.slices)) 1 else 10,
 nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), select.variables, zlim, legend=TRUE,
  MARGIN.movie = NULL,...)
## S4 method for signature 'RFspatialDataFrame,data.frame'
plot(x, y,
 MARGIN=c(1,2), MARGIN.slices=NULL,
 n.slices = if (is.null(MARGIN.slices)) 1 else 10,
 nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), select.variables, zlim, legend=TRUE,
  MARGIN.movie = NULL,...)


## S4 method for signature 'RFspatialGridDataFrame'
persp(x, y,
 MARGIN=c(1,2), MARGIN.slices=NULL,
 n.slices = if (is.null(MARGIN.slices)) 1 else 10,
 nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), select.variables, zlim, legend=TRUE,
 MARGIN.movie = NULL,...)


## S4 method for signature 'RFdataFrame,missing'
plot(x, y, nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), legend=TRUE, ...)

## S4 method for signature 'RFdataFrame,RFdataFrame'
plot(x, y, nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), legend=TRUE, ...)

## S4 method for signature 'RFdataFrame,matrix'
plot(x, y, nmax=6,
```

```
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), legend=TRUE, ...)

## S4 method for signature 'RFdataFrame,data.frame'
plot(x, y, nmax=6,
 plot.variance = (!is.null(x@.RFparams$has.variance) &&
 x@.RFparams$has.variance), legend=TRUE, ...)

## S4 method for signature 'RFempVariog,missing'
plot(x, model=NULL,

nmax.phi=NA, nmax.theta=NA, nmax.T=NA,
 plot.nbin=TRUE, plot.sd=FALSE, variogram=TRUE, boundaries = TRUE,...)

## S4 method for signature 'RFfit,missing'
plot(x, model=NULL,  fit.method="ml",
 nmax.phi=NA, nmax.theta=NA, nmax.T=NA,
 plot.nbin=TRUE, plot.sd=FALSE, variogram = TRUE, boundaries = TRUE,...)

## S4 method for signature 'RMmodel,missing'
plot(x, y, dim=1, n.points=200,
 fct.type=NULL, MARGIN, fixed.MARGIN, maxchar=15, ...)

## S3 method for class 'RMmodel'
points(x, y, n.points=200, fct.type=NULL, ...)

## S3 method for class 'RMmodel'
lines(x, y, n.points=200, fct.type=NULL, ...)
```

Arguments

x	object of class RFsp or RFempVario or RFfit or RMmodel; in the latter case, x can be any sophisticated model but it must be either stationary or a variogram model
y	ignored in most methods
MARGIN	vector of two; two integer values giving the coordinate dimensions w.r.t. which the field or the covariance model is to be plotted; in all other directions, the first index is taken
MARGIN.slices	integer value; if $[space-time-dimension > 2]$, MARGIN.slices can specify a third dimension w.r.t. which a sequence of slices is plotted. Currently only works for grids.
fixed.MARGIN	only for class(x)=="RMmodel" and if dim > 2; a vector of length dim-2 with distance values for the coordinates that are not displayed
n.slices	integer value. The number of slices to be plotted; if n.slices>1, nmax is set to 1. Or n.slices is a vector of 3 elements: start, end, length. Currently only works for grids.

`nmax` the maximal number of the `x@.RFparams$n` iid copies of the field that are to be plotted

`MARGIN.movie` integer. If given a sequence of figures is shown for this direction. This option is in an experimental stage. It works only for grids.

`...` arguments to be passed to methods; mainly graphical arguments, or further models in case of class `'RMmodel'`, see Details.

`fit.method` character; only for `class(x)=="RFfit"`; a vector of slot names for which the fitted covariance or variogram model is to be plotted; should be a subset of `slotNames(x)` for which the corresponding slots are of class `"RMmodelFit"`; by default, the maximum likelihood fit will be plotted

`nmax.phi` integer; only for `class(x)=="RFempVario"`; the maximal number of bins of angle phi that are to be plotted

`nmax.theta` integer; only for `class(x)=="RFempVario"`; the maximal number of bins of angle theta that are to be plotted

`nmax.T` integer; only for `class(x)=="RFempVario"`; the maximal number of different time bins that are to be plotted

`plot.nbin` logical; only for `class(x)=="RFempVario"`; indicates whether the number of pairs per bin are to be plotted

`plot.sd` logical; only for `class(x)=="RFempVario"`; indicates whether the calculated standard deviation (`x@sd`) is to be plotted (in form of arrows of length +-1*sd)

`variogram` logical; This argument should currently not be set by the user. If `TRUE` then the empirical variogram is plotted, else an estimate for the covariance function

`boundaries` logical; only for `class(x)=="RFempVario"` and the anisotropic case where `model` is given. As the empirical variogram is calculated on a sector of angles, no exact variogram curve corresponds to the mean values in this sector. If `boundaries=TRUE` the values of the variogram on the sector boundaries are plotted. If `FALSE` some kind of mean model values are plotted. Neither the boundaries may contain the values of emprical variogram nor does the mean values need to be close the emprical variogram.

`dim` must equal 1 or 2; only for `class(x)=="RMmodel"`; the covariance function and the variogram are plotted as a function of $\mathsf{R}^{\mathsf{dim}}$.

`n.points` integer; only for `class(x)=="RMmodel"`; the number of points at which the model evaluated (in each dimension); defaults to 200

`fct.type` character; only for `class(x)=="RMmodel"`; must equal `NULL`, `"Cov"` or `"Variogram"`; controls whether the covariance (`fct.type="Cov"`) or the variogram (`fct.type="Variogram"`) is plotted; `NULL` implies automatic choice, where `"Cov"` is chosen whenever the model is stationary

`model` object of class `RMmodel`; only for `class(x)=="RFempVario"` or `class(x)=="RFfit"`; a list of covarianve or variogram models that are to be plotted into the same plot as the empirical variogram (and the fitted models)

`plot.variance` logical, whether variances should be plotted if available

`select.variables`
vector of integers or list of vectors. The argument is only of interest for multivariate models. Here, `length(select.variables)` gives the number of pictures shown (excuding the plot for the variances, if applicable). If `select.variables`

is a vector of integers then exactly these components are shown. If `select.variables` is a list, each element should be a vector up to length $l \leq 3$:

- $l = 1$
  the component is shown in the usual way
- $l = 2$
  the two components are interpreted as vector and arrows are plotted
- $l = 3$
  the first component is shown as single component; the remaining two component are interpreted as a vector and plotted into the picture of the first component

`legend` logical, whether a legend should be plotted

`zlim` vector of length 2 with the usual meaning. In case of multivariate random fields, `zlim` can also be a character wih the value 'joint' indicating that all plotted compoments shall have the same zlim OR a matrix with two rows, where the i-th column gives the zlim of the i-th variable OR a list with entries named `data` and `var` if a separate `zlim` for the Kriging variance is to be used.

`maxchar` maximum number of characters used in the legend (for multiplots only)

Details

Internally, `...` are passed to `image` and `plot.default`, respectively; if, by default, multiple colors, xlabs or ylabs are used, also vectors of suitable length can be passed as `col`, `xlab` and `ylab`, respectively.

One exception is the use of `...` in `plot` for class `'RMmodel'`. Here, further models might be passed. All models must have names starting with `model`. If `'.'` is following in the name, the part of the name after the dot is shown in the legend. Otherwise the name is ignored and a standardized name derived from the model definition is shown in the legend. Note that for the first argument a name cannot be specified.

Methods

`signature(x = "RFspatialGridDataFrame", y = "missing")` Generates nice image plots of simulation results for simulation on a grid and space-time-dimension ≥ 2. If space-time-dimension ≥ 3, plots are on 2-dimensional subspaces. Handles multivariate random fields (`.RFparams$vdim>1`) as well as repeated iid simulations (`.RFparams$vdim>n`).

`signature(x = "RFspatialGridDataFrame", y = "RFspatialPointsDataFrame")` Similar to method for `y="missing"`, but additionally adds the points of `y`. Requires `MARGIN.slices=NULL` and `all.equal(x@.RFparams, y@.RFparams)`.

`signature(x = "RFspatialGridDataFrame", y = "matrix")` Similar to method for `y="missing"`, but additionally adds the points of `y`. Requires `MARGIN.slices=NULL` and `all.equal(x@.RFparams, y@.RFparams)`.

`signature(x = "RFspatialPointsDataFrame", y = "RFspatialGridDataFrame")` Throws an error. Probably `x` and `y` have been interchanged.

`signature(x = "RFspatialPointsDataFrame", y = "missing")` Similar to method for class `RFspatialGridDataFrame`, but for non-gridded simulation results. Instead of a grid, only existing points are plotted with colors indicating the value of the random field at the respective location. Handles multivariate random fields (`.RFparams$vdim>1`) as well as repeated iid simulations (`.RFparams$vdim>n`).

`signature(x = "RFspatialPointsDataFrame", y = "RFspatialPointsDataFrame")` Similar to method for `y="missing"`, but additionally adds the points of `y`. Requires `all.equal(x@.RFparams, y@.RFparams)`.

`signature(x = "RFgridDataFrame", y = "missing")` Generates plots of simulation results for space-time-dimension $= 1$. Handles different values for the number of repetitions as well as multivariate responses.

`signature(x = "RFpointsDataFrame", y = "missing")` Similar to method for class `RFgridDataFrame`, but for non-gridded data.

`signature(x = "RFempVario", y = "missing")` Gives nice plots of the empirical variogram; handles binning in up to three space-dimensions and a time-dimension, where the empirical variogram is plotted along lines which are directed according to the angle-centers given in `x@phi.centers` and `x@theta.centers`; arbitrary theoretical model curves can be added to the plot by using the argument `model`. If no bins are given, i.e. (`x@bin=NULL`), `image`-plots are generated.

`signature(x = "RFfit", y = "missing")` Combines the plot of the empirical variogram with the estimated covariance or variogram model (theoretical) curves; further models can be added via the argument `model`.

`signature(x = "RMmodel", y = "missing")` Generates covariance function or variogram function plots in one or two dimensions.

Author(s)

Alexander Malinowski, Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again



## define the model:
model <- RMtrend(mean=0.5) + # mean
RMstable(alpha=1, var=4, scale=10) +
# see help("RMstable")
# for additional arguments
RMnugget(var=1) # nugget


###############################################################
## Plot of covariance structure

plot(model)
plot(model, xlim=c(0, 30))
plot(model, xlim=c(0, 30), fct.type="Variogram")
plot(model, xlim=c(-10, 20), fct.type="Variogram", dim=2)

###############################################################
## Plot of simulation results
```

```
## define the locations:
from <- 0
step <- .1
len <- if (interactive()) 50 else 5 ## nicer, but also time consuming if len <- 00
x1D <- GridTopology(from, step, len)
x2D <- GridTopology(rep(from, 2), rep(step, 2), rep(len, 2))
x3D <- GridTopology(rep(from, 3), rep(step, 3), rep(len, 3))

## 1-dimensional
sim1D <- RFsimulate(model = model, x=x1D, n=6)
plot(sim1D, nmax=4)

## 2-dimensional
sim2D <- RFsimulate(model = model, x=x2D, n=6)
plot(sim2D, nmax=4)
plot(sim2D, nmax=4, col=terrain.colors(64),
main="My simulation", xlab="my_xlab")

## 3-dimensional
model <- RMmatern(nu=1.5, var=4, scale=2)
sim3D <- RFsimulate(model = model, x=x3D)
plot(sim3D, MARGIN=c(2,3), MARGIN.slices=1, n.slices=4)


############################################################
## empirical variogram plots

x <- seq(0, 10, 0.05)
bin <- seq(from=0, by=.2, to=3)

model <- RMexp()
X <- RFsimulate(x=cbind(x), model=model)
ev1 <- RFempiricalvariogram(data=X, bin=bin)
plot(ev1)

model <- RMexp(Aniso = cbind(c(10,0), c(0,1)))
X <- RFsimulate(x=cbind(x,x), model=model)
ev2 <- RFempiricalvariogram(data=X, bin=bin, phi=3)
plot(ev2, model=list(exp = model))


############################################################
## plot Fitting results
x <- c(0,.1,20)
x <- seq(0, 10, 0.05)

model <- RMexp(Aniso = cbind(c(10,0), c(0,1)))
X <- RFsimulate(x=cbind(x,x), model=model)
fit <- RFfit(~RMexp(Aniso=matrix(nc=2, nr=2, NA)), data=X, fit.nphi = 2)
plot(fit)
```

```
#############################################################
## plot Kriging results
model <- RMwhittle(nu=1.2, scale=2)
x <- runif(200, max=step*len/2)
y <- runif(200, max=step*len/2) # 200 points in 2 dimensional space
sim <- RFsimulate(model = model, x=x, y=y)

interpolate <- RFinterpolate(model=model, x=x2D, data=sim)
plot(interpolate)
plot(interpolate, sim)



#############################################################
## plotting vector-valued results
model <- RMdivfree(RMgauss(), scale=4)
x <- y <- seq(-10,10, if (interactive()) 0.5 else 5)
simulated <- RFsimulate(model = model, x=x, y=y, n=1)
plot(simulated)
plot(simulated, select.variables=list(1, 1:3, 4))



#############################################################
## options for the zlim argument
model <- RMdelay(RMstable(alpha=1.9, scale=2), s=c(0, 4)) +
         RMdelay(RMstable(alpha=1.9, scale=2), s=c(4, 0))
simu <- RFsimulate(model, x, y)

plot(simu, zlim=list(data=cbind(c(-6,2), c(-2,1)), var=c(5,6)))
plot(simu, zlim=cbind(c(-6,2), c(-2,1)))
plot(simu, zlim=c(-6,2))
plot(simu, zlim="joint")
```

`Print` *Nice print function returning also the names automatically*

Description

prints variable names and the values

Usage

```
Print(..., digits = 6, empty.lines = 2)
```

Arguments

...	any object that can be print-ed
digits	see print
empty.lines	number of leading empty lines

Value

prints the names and the values; for vectors cat is used and for lists str

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
  a <- 4
  b <- list(c=5, g=7)
  m <- matrix(1:4, nc=2)
  Print(a, b, m)
```

PrintModelList *Information about the implemented covariance models*

Description

PrintModelList prints the list of currently implemented models including the corresponding simulation methods

Usage

```
PrintModelList(operators=FALSE, internal=FALSE, newstyle=TRUE)
```

Arguments

operators	logical. Flag whether operators should be also considered.
internal	logical. Flag whether internal models should be also considered. In case of PrintModelList and internal=2, variants of internal models are also printed.
newstyle	logical. If FALSE then only the old style model names (Version 2 and earlier) are shown. These names can still be used in the list definition of models, see RMmodelsAdvanced. If TRUE then the standard names will also be shown.

Details

See RMmodel for a description of the models and their use.

Value

`PrintModelList` prints a table of the currently implemented covariance functions and the matching methods. `PrintModelList` returns `NULL`.

Note

From version 3.0 on, the command `PrintModelList()` is replaced by the call `RFgetModelNames(internal=FALSE)`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RFgetModelNames

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

PrintModelList()
```

RFcov *Evaluate Covariance and Variogram Functions*

Description

RFcov returns the values of a covariance function; RFvariogram returns the values of a variogram; RFpseudovariogram returns the values of a pseudovariogram; RFcovmatrix returns the covariance matrix for a set of points RFfctn returns the values of a shape function;

Usage

```
RFcov(model, x, y = NULL, z = NULL, T = NULL, grid, distances, dim,
      fctcall = c("Cov", "CovMatrix", "Variogram", "Pseudovariogram",
                  "Fctn"),
       ...)

RFvariogram(model, x, y = NULL, ...)

RFpseudovariogram(model, x, y = NULL, ...)
```

```
RFcovmatrix(...)

RFfctn(...)
```

Arguments

model	object of class RMmodel; the covariance or variogram model, which is to be evaluated
x	vector or $(n \times$ dim$)$-matrix, where n is the number of points at which the covariance function is to be evaluated; in particular, if the model is isotropic or dim=1 then x is a vector. x
y	second vector or matrix for non-stationary covariance functions
z	z-component of point if xyzT-specification of points is used
T	T-component of point if xyzT-specification of points is used
grid	boolean; whether xyzT specify a grid
distances	vector; only if the function RFcovmatrix is used; the lower triangular part of the distance matrix column-wise; equivalently the upper triangular part of the distance matrix row-wise; either x or distances must be missing
dim	dimension of the coordinate space in which the model is applied
fctcall	internal. This argument should not be considered by the user
...	arguments passed to RFcov (RFcovmatrix) and arguments passed to RFoptions

Details

RFcovmatrix returns a covariance matrix. Here a matrix of of coordinates (x) or a vector or a matrix of distances is expected. RFcovmatrix allows also for variogram models. Then the negative of the variogram matrix is returned.

Value

RFcov returns a vector of values of the covariance function.

RFvariogram returns a vector of values of the variogram model.

RFpseudovariogram returns a vector of values of the variogram model.

RFcovmatrix returns a covariance matrix.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

# locations:
x <- matrix(runif(15), ncol=3)
 # coordinate matrix of 5 arbitrary points
 # p1, p2, p3, p4, p5 in 3-dimensional space
y <- matrix(runif(15),ncol=3)
 # coordinate matrix of 5 arbitrary points
 # q1, q2, q3, q4, q5 in 3-dimensional space


######################################################
# get available models
RFgetModelNames()

######################################################
# Example 1: a stationary covariance model

model <- RMexp()

# covariance only depends on differences
# of locations hi=pi-qi in 3-dimensional space
# therefore, the following 2 commands yield the same

RFcov(model=model, x=x, y=y)
RFcov(model=model, x=x-y)
 # yields 5 values C(hi)=C(pi,qi) for i=1,2,3,4,5



################################################
# Example 2: get covariance matrix C(x_i,x_j)
# at given locations x_i, i=1,...,n
#
# here for an isotropic stationary covariance model
# yields a 4 times 4 covariance matrix of the form
# C(0)   C(5)   C(3)   C(2.5)
# C(5)   C(0)   C(4)   C(2.5)
# C(3)   C(4)   C(0)   C(2.5)
# C(2.5) C(2.5) C(2.5) C(0)

model <- RMexp() # the covariance function C(x,y)=C(r) of this model
#                  depends only on the distance r between x and y
RFcovmatrix(model=model, distances=c(5,3,2.5,4,2.5,2.5), dim=4)



######################################################
# Example 3: distinguish the different uses of x and y
```

```
x <- c(1,2,1)
y <- c(4,5,6)

# coordinate space 1-dimensional, evaluated at 3 points:
RFcov(model=model, x=as.matrix(x), y=as.matrix(y))
# coordinate space is 3-dimensional, evaluated at a pair of points
RFcov(model=model, x=t(x), y=t(y))
```

`RFcrossvalidate` *Fitting model parameters to spatial data (regionalised variables) and to linear (mixed) models*

Description

The function estimates arbitrary parameters of a random field specification with various methods. Currenty, the model to be fitted can be

- Gaussian random fields
- linear models

The fitting of max-stable random fields and others has not been implemented yet.

Usage

```
D
RFcrossvalidate(model, x, y = NULL, z = NULL, T = NULL, grid, data,
 lower = NULL, upper = NULL, bc_lambda, method="ml",
 users.guess = NULL,
 distances = NULL, dim, optim.control = NULL,transform = NULL,
 full = FALSE, ...)
```

Arguments

`model, x, y, z, T, grid, data,lower, upper, bc_lambda, users.guess, distances, dim, optim.control,`	see `RFfit`
`method`	Single method to be used for estimating, either one of the `methods` or one of the `sub.methods` see `RFfit`
`full`	logical. if `TRUE` then crossvalidatiaon is also performed for intermediate models used in `RFfit` (if any).

Value

An object of the class "RFcrossvalidate" which is a list with the following components, cf. xvalid in the package **geoR** :

data	the original data.
predicted	the values predicted by cross-validation.
krige.var	the cross-validation prediction variance.
error	the differences data - predicted value.
std.error	the errors divided by the square root of the prediction variances.
p	In contrast to **geoR** the p-value is returned, i.e. the probability that a difference with absolute value larger than the absolute value of the actual difference is observed. A method for summary returns summary statistics for the errors and standard errors similar to **geoR**. If cross_refit = TRUE and detailed_output = TRUE the returned object also constains a fitted which is a list of fitted models.

Methods

print prints the summary

summary gives a summary

Note

An important option is cross_refit that determines whether the model is refitted for each location left out. Default is FALSE. See also RFoptions.

Note

This function does not depend on the value of RFoptions()$PracticalRange. The function RFcrossvalidate always uses the standard specification of the covariance model as given in RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Ribeiro, P.J., Jr. and Diggle, P.J (2014) R package **geoR**.
- Burnham, K. P. and Anderson, D. R. (2002) *Model selection and Multi-Model Inference: A Practical Information-Theoretic Approach.* 2nd edition. New York: Springer.

See Also

RFratiotest RFfit RMmodel, RandomFields, weather.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again



RFoptions(modus_operandi="sloppy")

n <- if (interactive()) 100 else 5

############################################################
## simulate some data first
points <- if (interactive()) 100 else 40
x <- runif(points, 0, 3)
y <- runif(points, 0, 3) ## random points in square [0, 3]^2
model <- RMgencauchy(alpha=1, beta=2)
d <- RFsimulate(model, x=x, y=y, grid=FALSE, n=n) #1000


############################################################
## estimation; 'NA' means: "to be estimated"
estmodel <- RMgencauchy(var=NA, scale=NA, alpha=NA, beta=2) +
            RMtrend(mean=NA)
RFcrossvalidate(estmodel, data=d)
```

RFdistr *Evaluating distribution families*

Description

Through RRdistr distribution families can the passed to **RandomFields** to create distributions available in the RMmodel defintions

Usage

```
RFddistr(model, x, dim=1, ...)
RFpdistr(model, q, dim=1, ...)
RFqdistr(model, p, dim=1, ...)
RFrdistr(model, n, dim=1, ...)
RFdistr(model, x, q, p, n, dim=1, ...)
```

Arguments

model	an RRmodel
x	the location where the density is evaluated
q	the location there the probability function is evaluated
p	the value where the quantile function is evaluated
n	the number of random values to be drawn
dim	the dimension of the vector to be drawn
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions

Details

RFdistr is the generic function for the 4 functions belonging to a distribution.

Value

as described in the arguments

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RRgauss, RR

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## a very toy example to understand the use
model <- RRdistr(norm())
v <- 0.5
Print(RFdistr(model=model, x=v), dnorm(x=v))
Print(RFdistr(model=model, q=v), pnorm(q=v))
Print(RFdistr(model=model, p=v), qnorm(p=v))

n <- 10
r <- RFdistr(model=model, n=n, seed=0)
set.seed(0); Print(r, rnorm(n=n))


## note that a conditional covariance function given the
## random parameters is given here:
model <- RMgauss(scale=exp())
for (i in 1:3) {
  RFoptions(seed = i + 10)
```

```
  readline(paste("Model no.", i, ": press return", sep=""))
  plot(model)
  readline(paste("Simulation no.", i, ": press return", sep=""))
  plot(RFsimulate(model, x=seq(0,10,0.1)))
}
```

RFempiricalvariogram *Empirical (Cross-)Variogram*

Description

Calculates the empirical (cross-)variogram. The empirical (cross-)variogram of two random fields X and Y is given by

$$\gamma(r) := \frac{1}{2N(r)} \sum_{(t_i,t_j)|t_{i,j}=r} (X(t_i) - X(t_j))(Y(t_i) - Y(t_j))$$

where $t_{i,j} := t_i - t_j$, and where $N(r)$ denotes the number of pairs of data points with distancevector $t_{i,j} = r$.

Usage

```
RFempiricalvariogram(x, y = NULL, z = NULL, T = NULL, data, grid,
 bin, phi, theta, deltaT, distances, vdim, ...)
```

Arguments

x	matrix of coordinates, or vector of x coordinates, or object of class GridTopology or raster. If matrix, ncol(x) is the dimension of the index space. Matrix notation is required in case of more than 3 spatial dimensions; in this case, if grid=FALSE, x_ij is the i-th coordinate in the j-th dimension. Otherwise, if grid=TRUE, the columns of x are interpreted as gridtriples (see grid). if of class GridTopology , x is interpreted as grid definition and grid is automatically set to TRUE. Coordinates are not required if the data is an object of class RFsp, as these objects already contain its coordinates
y	optional vector of y coordinates, ignored if x is a matrix
z	optional vector of z coordinates, ignored if x is a matrix
T	optional vector of time coordinates, T must always be an equidistant vector or given in a gridtriple format (see grid); for each component of T, the random field is simulated at all location points; the argument T is in an experimental stage.
grid	logical; determines whether the vectors x, y, and z or the columns of x should be interpreted as a grid definition (see Details). If grid=TRUE, either x, y, and z must be equidistant vectors in ascending order or the columns of x must be given in the gridtriple format c(from, stepsize, len) (see Details); Not required if data is of class RFsp

data	matrix, data.frame or object of class RFsp;
bin	a vector giving the borders of the bins; If not specified an array describing the empirical (pseudo-)(cross-) variogram in every direction is returned.
phi	an integer defining the number of sectors one half of the X/Y plane shall be devided into. If not specified, either an array is returned (if bin missing) or isotropy is assumed (if bin specified)
theta	an integer defining the number of sectors one half of the X/Z plane shall be devided into. Use only for dimension $d = 3$ if phi is already specified
deltaT	vector of length 2, specifying the temporal bins. The internal bin vector becomes `seq(from=0, to=deltaT[1], by=deltaT[2])`
distances	object of class dist representing the upper trianguar part of the matrix of Euclidean distances between the points at which the field is to be simulated; only applicable for stationary and isotropic models; if not NULL, dim must be given and x, y, z and T must be missing or NULL.
vdim	the number of variables of a multivariate data set. If not given and data is an RFsp object created by **RandomFields**, the information there is taken from there. Otherwise vdim is assumed to be one. NOTE: still the argument vdim is an experimental stage.
...	further options and control arguments for the simulation that are passed to and processed by RFoptions.

Details

RFempiricalvariogram computes the empirical cross-variogram for given (multivariate) spatial data.

The spatial coordinates x, y, z should be vectors. For random fields of spatial dimension $d > 3$ write all vectors as colums of matrix x. In this case do neither use y, nor z and write the colums in gridtriple notation.

If the data is spatially located on a grid a fast algorithm based on the fast Fourier transformed (fft) will be used. As advanced option the calculation method can also be changed for grid data (see RFoptions.)

It is also possible to use RFempiricalvariogram to calulate the pseudovariogram (see RFoptions).

Value

RFempiricalvariogram returns objects of class RFempVariog.

Author(s)

Sebastian Engelke, <sebastian.engelke@unil.ch> https://hec.unil.ch

Johannes Martini, <jmartin2@gwdg.de> http://stochastik.math.uni-goettingen.de

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.

Stein, M. L. (1999) *Interpolation of Spatial Data.* New York: Springer-Verlag

See Also

RMstable, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

n <- 1 ## use n <- 2 for better results

## isotropic model
model <- RMexp()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
z <- RFsimulate(model, x=x, n=n)
emp.vario <- RFempiricalvariogram(data=z)
plot(emp.vario, model=model)


## anisotropic model
model <- RMexp(Aniso=cbind(c(2,1), c(1,1)))
x <- seq(0, 10, if (interactive()) 0.05 else 1)
z <- RFsimulate(model, x=x, y=x, n=n)
emp.vario <- RFempiricalvariogram(data=z, phi=4)
plot(emp.vario, model=model)


## space-time model
model <- RMexp()
x <- seq(0, 10, if (interactive()) 0.05 else 1)
T <- c(0, if (interactive()) 0.1 else 1, if (interactive()) 100 else 10)
z <- RFsimulate(x=x, T=T, model=model, n=n)
emp.vario <- RFempiricalvariogram(data=z, deltaT=c(10, 1))
plot(emp.vario, model=model, nmax.T=3)


## multivariate model
model <- RMbiwm(nudiag=c(1, 2), nured=1, rhored=1, cdiag=c(1, 5),
                s=c(1, 1, 2))
x <- seq(0, 20, if (interactive()) 0.1 else 2)
z <- RFsimulate(model, x=x, y=x, n=n)
emp.vario <- RFempiricalvariogram(data=z)
plot(emp.vario, model=model)


## multivariate and anisotropic model
```

```
model <- RMbiwm(A=matrix(c(1,1,1,2), nc=2),
                nudiag=c(0.5,2), s=c(3, 1, 2), c=c(1, 0, 1))
x <- seq(0, 20, if (interactive()) 0.1 else 2)
data <- RFsimulate(model, x, x, n=n)
ev <- RFempiricalvariogram(data=data, phi=4)
plot(ev, model=model, boundaries=FALSE)
```

`RFempVariog-class` *Class* `RFempVariog`

Description

Class for RandomField's representation of empirical variograms

Arguments

`object, x, ...` see the respective generic function; The argument

Slots

`centers`: the bin centres of the spatial distances

`emp.vario`: value of the empirical variogram

`var`: the empirical (overall) variance in the data

`sd`: standard deviation of the variogram cloud within each bin

`n.bin`: number of bins

`phi.centers`: centres of the bins with respect to the (first) angle (for anisotropic empricial variograms only)

`theta.centers`: centres of the bins with respect to the second angle (for anisotropic empricial variograms in 3D only)

`T`: the bin centres of the time axis

`vdim`: the multivariate dimension

`coord.units`: string giving the units of the coordinates, see also option `coordunits` of `RFoptions`.

`variab.units`: string giving the units of the variables, see also option `varunits` of `RFoptions`.

`call`: language object; the function call by which the object was generated

Methods

plot `signature(x = "RFempVariog")`: gives a plot of the empirical variogram, for more details see `plot-method`.

as `signature(x = "RFempVariog")`: converts into other formats, only implemented for target class `list`.

show `signature(x = "RFfit")`: returns the structure of `x`

persp codesignature(obj = "RFempVario"): generates nice `persp` plots

print `signature(x = "RFfit")`: identical with show-method

summary provides a summary

Details

`print` returns also an invisible list that is convenient to access.

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>

See Also

`RFempiricalvariogram`, `plot-method`

Examples

```
#  see 'RFempiricalvariogram'
```

RFfit — *Fitting model parameters to spatial data (regionalised variables) and to linear (mixed) models*

Description

The function estimates arbitrary parameters of a random field specification with various methods. Currenty, the model to be fitted can be

- Gaussian random fields
- linear models

The fitting of max-stable random fields and others has not been implemented yet.

Usage

```
RFfit(model, x, y = NULL, z = NULL, T = NULL, grid, data,
 lower = NULL, upper = NULL, bc_lambda, methods,
 sub.methods, optim.control = NULL, users.guess = NULL,
 distances = NULL, dim, transform = NULL, ...)
```

Arguments

`model`	covariance model, see `RMmodel` or type `RFgetModelNames()` to get all options. All parameters that are set to `NA` will be estimated; see the examples below.
`x`	vector of x coordinates, or object of class `GridTopology` or `raster`; For more options see RFsimulateAdvanced.
`y`	vector of y coordinates
`z`	vector of z coordinates
`T`	vector of T coordinates; these coordinates are given in triple notation, see `RFsimulate`.
`data`	vector or matrix of values measured at `coord`; If a matrix is given then the columns are interpreted as independent realisations. If also a time component is given, then in the data the indices for the spatial components run the fastest. If an `m`-variate model is used, then each realisation is given as `m` consecutive columns of `data`.
`lower`	list or vector. Lower bounds for the parameters. If `param` is a vector, `lower` has to be a vector as well and its length must equal the number of parameters to be estimated. The order of `param` has to be maintained. A component being `NA` means that no manual lower bound for the corresponding parameter is set. If `param` is a list, `lower` has to be of (exactly) the same structure.
`upper`	list or vector. Upper bounds for the parameters. See also lower.
`grid`	boolean. Whether coordinates give a grid
`bc_lambda`	a vector of at most two numerical components (just one component corresponds to two identical ones) which are the parameters of the box-cox-transformation: $\frac{x_1^{\lambda}-1}{\lambda}_1 + \lambda_2$ If the model is univariate, the first parameter can be estimated by using `NA`.
`methods`	Main methods to be used for estimating. If several methods, estimation will be performed with each method and the results reported.
`sub.methods`	variants of the least squares fit of the variogram. See Details. variants of the maximum likelihood fit of the covariance function. See Details.
`users.guess`	User's guess of the parameters. All the parameters must be given using the same rules as for either `param` (except that no NA's should be contained) or `model`.
`distances,dim`	Instead of `x`-coordinates, distances might be given. The the dimension of the space `dim` must be given explicitely.
`optim.control`	control list for `optim`, which uses 'L-BFGS-B'. However `parscale` may not be given.
`transform`	this is an attempt to allow binding between parameters, e.g. one parameter value is supposed to equal another one, See examples below. `transform=list()` is not valid for estimating, but returns structural information to set up the correct function. See examples below.
`...`	further options and control arguments for the simulation that are passed to and processed by `RFoptions`.

Details

For details on the simulation methods see

- fitgauss for Gaussian random fields
- fitgauss for linear models

If x-coordinates are not given, the function will check `data` for `NA`s and will perform imputing.

The function has many more options to tune the optimizer, see `RFoptions` for details.

If the model defines a Gaussian random field, the options for `methods` and `submethods` are currently `"ml"` and `c("self", "plain", "sqrt.nr", "sd.inv", "internal")`, respectively.

Value

The result depends on the logical value of `spConform`. If `TRUE`, an S4 object is greated. In case the model indicates a Gaussian random field, an RFfit object is created.

If `spConform=FALSE`, a list is returned. In case the model indicates a Gaussian random field, the details are given in fitgauss.

Note

This function does not depend on the value of `RFoptions()$PracticalRange`. The function `RFfit` always uses the standard specification of the covariance model as given in `RMmodel`.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Burnham, K. P. and Anderson, D. R. (2002) *Model selection and Multi-Model Inference: A Practical Information-Theoretic Approach.* 2nd edition. New York: Springer.

See Also

`RFratiotest` `RMmodel`, `RandomFields`, `weather`.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again



RFoptions(modus_operandi="sloppy")


n <- if (interactive()) 100 else 5
```

```
##########################################################
## simulate some data first                             ##
points <- if (interactive()) 100 else 40
x <- runif(points, 0, 3)
y <- runif(points, 0, 3) ## random points in square [0, 3]^2
model <- RMgencauchy(alpha=1, beta=2)
d <- RFsimulate(model, x=x, y=y, grid=FALSE, n=n) #1000


##########################################################
## estimation; 'NA' means: "to be estimated"            ##
estmodel <- RMgencauchy(var=NA, scale=NA, alpha=NA, beta=2) +
            RMtrend(mean=NA)
RFfit(estmodel, data=d)


##########################################################
## Estimation with fixed sill (variance + nugget       ##
## equals a given constant)                             ##
estmodel <- RMgencauchy(var=NA, scale=NA, alpha=NA, beta=NA) +
            RMnugget(var=NA) + RMtrend(mean=NA)
RFfit(estmodel, data=d, fit.sill=1, fit.optim_var_elimination="try")



##########################################################
## estimation in a anisotropic framework                ##
x <- y <- (1:3)/4
model <- RMexp(Aniso=matrix(nc=2, c(4,2,-2,1)), var=1.5)
d <- RFsimulate(model, x=x, y=y, n=n)
estmodel <- RMexp(Aniso=matrix(nc=2, c(NA,NA,-2,1)), var=NA) +
            RMtrend(mean=NA)
RFfit(estmodel, data=d, fit.nphi=20)



##########################################################
## AN EXAMPLE HOW TO USE OF PARAMETER 'transform'       ##
## estimation of coupled parameters (first column of    ##
##           the matrix 'Aniso' has identical entries) ##
# source("RandomFields/tests/source.R")
RFfit(estmodel, data=d, transform=list()) # shows positions of NAs
f <- function(param) param[c(1,2,2)]
RFfit(estmodel, data=d, transform=list(c(TRUE, TRUE, FALSE), f))
```

RFfit-class *Class* RFfit

Description

Class for RandomField's representation of model estimation results

Arguments

object, ...	see the respective generic function; The argument
method	string. selects that set of parameters which has been obtained through a specific fitting method
full	logical. if TRUE submodels are reported as well (if available).

Creating Objects

Objects are created by the function RFfit

Slots

autostart: RMmodelFit; contains the estimation results for the method 'autostart' including a likelihood value, a constant trend and the residuals

boxcox: logical; whether the parameter of a Box Cox tranformation has been estimated

coord.units: string giving the units of the coordinates, see also option coordunits of RFoptions.

deleted: integer vector. positions of the parameters that has been deleted to get the set of variables, used in the optimazation

ev: list; list of objects of class RFempVariog, contains the empirical variogram estimates of the data

fixed: list of two vectors. The fist gives the position where the parameters are set to zero. The second gives the position where the parameters are set to one.

internal1: RMmodelFit; analog to slot 'autostart'

internal2: RMmodelFit; analog to slot 'autostart'

internal3: RMmodelFit; analog to slot 'autostart'

lowerbounds: RMmodel; covariance model in which each parameter value gives the lower bound for the respective parameter

ml: RMmodelFit; analog to slot 'autostart'

modelinfo: Table with information on the parameters: name, boundaries, type of parameter

n.covariates: number of covariates

n.param: number of parameters (given by the user)

n.variab: number of variables (used internally); n.variab is always less than or equal to n.param

number.of.data: the number of data values passed to RFfit that are not NA or NaN

`number.of.parameters`: total number of parameters of the model that had to be estimated including variances, scales, co-variables, etc.

`p.proj`: vector of integers. The original position of those parameters that are used in the submodel

`plain`: RMmodelFit; analog to slot 'autostart'

`report`: if not empty, it indicates that this model should be reported and gives a standard name of the model.

Various function, e.g. `print.RMmodelFit` uses this information if their argument `full` equals `TRUE`.

`self`: RMmodelFit; analog to slot 'autostart'

`sd.inv`: RMmodelFit; analog to slot 'autostart'

`sqrt.nr`: RMmodelFit; analog to slot 'autostart'

`submodels`: list. Sequence (in some cases even nested sequence) of models that is used to determine an initial value in

`table`: matrix; summary of estimation results of different methods

`transform`: function;

`true.tsdim`: time space dimension of the (original!) data, even for submodels that consider parts of separable models.

`true.vdim`: multivariability of the (original!) data, even for submodels that consider independent models for the multivariate components.

`upperbounds`: RMmodel; see slot 'lowerbounds'

`users.guess`: RMmodelFit; analog to slot 'autostart'

`ml`: RMmodelFit; analog to slot 'autostart'; with maximum likelihood method

`v.proj`: vector of integers. The components selected in one of the submodels

`variab.units`: string giving the units of the variables, see also option `varunits` of `RFoptions`.

`x.proj`: logical or integer. If logial, it means that no separable model is considered there. If integer, then it gives the considered directions of a separable model

`Z`: standardized list of information on the data

Methods

plot `signature(x = "RFfit")`: gives a plot of the empirical variogram together with the fitted model, for more details see `plot-method`.

show `signature(x = "RFfit")`: returns the structure of `x`

persp codesignature(obj = "RFfit"): generates `persp` plots

print `signature(x = "RFfit")`: identical with show-method, additional argument is `max.level`

[`signature(x = "RFfit")`: enables accessing the slots via the `"["`-operator, e.g. `x["ml"]`

as `signature(x = "RFfit")`: converts into other formats, only implemented for target class `RFempVariog`

anova performs a likelihood ratio test base on a chisq approximation

summary provides a summary

logLik provides an object of class `"logLik"`

AIC,BIC provides the AIC and BIC information, respectively

Further 'methods'

```
AICc.RFfit(object, ..., method="ml", full=FALSE)]
AICc.RF_fit(object, ..., method="ml", full=TRUE)
```

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>, Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

AICc:

- Hurvich, C.M. and Tsai, C.-L. (1989) Regression and Time Series Model Selection in Small Samples *Biometrika*, **76**, 297-307.

See Also

RFfit, RFempiricalvariogram, RMmodel-class, RMmodelFit-class plot-method

Examples

```
# see RFfit
```

RFformula — *RFformula - syntax to design random field models with trend or linear mixed models*

Description

It is described how to create a formula, which can e.g. be used as an argument of RFsimulate and RFfit to simulate and to fit data accordingly to the model described by the formula.

In general, the created formula serves two purposes:

- to describe models in the "Linear Mixed Models"-framework including fixed and random effects
- to define models for random fields including trend surfaces from a geostatistical point of view.

Thereby, fixed effects and trend surfaces are adressed via the expression RMfixed and the function RMtrend; the covariance structures of the zero-mean multivariate normally distributed random effects and random field components are adressed by objects of class RMmodel, which allow for a very flexible covariance specification.

Details

The formula should be of the type

$$response \tilde{} fixedeffects + randomeffects + errorterm$$

or

$$response \tilde{} trend + zero - meanrandomfield + nuggeteffect,$$

respectively.

Thereby:

- response
  optional; name of response variable
- fixed effects/trend:
  optional, should be a sum (using `+`) of components either of the form `X@RMfixed(beta)` or `RMtrend(...)` with X being a design matrix and β being a vector of coefficients (see `RMfixed` and `RMtrend`).
  Note that a fixed effect of the form X is interpreted as `X@RMfixed(beta=NA)` by default (and β is estimated provided that the formula is used in `RFfit`).
- random effects/zero-mean random field:
  optional, should be a sum (using `+`) of components of the form `Z@model` where Z is a design matrix and `model` is an object of class `RMmodel`.
  `Z@model` describes a vector of random effects which is normally distributed with zero mean and covariance matrix $Z\Sigma Z^T$ where Z^T is the transpose of Z and Σ is the covariance matrix according to `model`.
  Note that a random effect/random fluctuation of the form `model` is viewed as `I@model` where I is the identity matrix of corresponding dimension.
- error term/nugget effect
  optional, should be of the form `RMnugget(...)`. `RMnugget` describes a vector of iid Gaussian random variables.
  Please note that the character "@" in the RFformula-context can only be used to multiply design-matrices with corresponding vectors of fixed or random effects, whereas in the context of S4-classes "@" is used to access slots of corresponding objects.

Note

(additional) argument names should always start with a capital letter. Small initial letters are reserved for `RFoptions`.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and P. Delfiner (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York, Chichester: John Wiley & Sons.
- McCulloch, C. E., Searle, S. R. and Neuhaus, J. M. (2008) *Generalized, linear, and mixed models.* Hoboken, NJ: John Wiley & Sons.

- Ruppert, D. and Wand, M. P. and Carroll, R. J. (2003) *Semiparametric regression.* Cambridge: Cambridge University Press.

See Also

RMmodel, RFsimulate, RFfit, RandomFields.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

RFoptions(modus_operandi="sloppy")

###############################################################
#
# Example  : Simulation and fitting of a two-dimensional
# Gaussian random field with exponential covariance function
#
###############################################################

V <- 10
S <- 0.3
model <- RMexp(var=V, scale=S)
x <- y <- seq(1, 3, by= if (interactive()) 0.1 else 1)

simulated <- RFsimulate(model = model, x=x, y=y)
# add overall mean/trend of 3 to the response/simulated data
simulated@data <- simulated@data + 3
#returns an object of class RFspatialPointsDataFrame including
#coordinates and the simulated response vector
plot(simulated)


# an alternative code to the above code:
simulated2 <- RFsimulate(model = ~ 1@RMfixed(beta=3) +
               RMexp(var=V, scale=S),x=x, y=y, V=V, S=S)
plot(simulated2)

# Estimate parameters of underlying covariance function via
# maximum likelihood
model.na <- ~ 1@RMfixed(beta=NA) + RMexp(var=NA, scale=NA)
fitted <- RFfit(model=model.na, data=simulated)

# compare sample mean of data with ML estimate:
mean(simulated@data[,1])
fitted@ml@trend
```

RFfractaldim *RFfractaldimension*

Description

The function estimates the fractal dimension of a process

Usage

```
RFfractaldim(x, y = NULL, z = NULL, data, grid,
 bin= seq(min(ct$x[2, ]) / 2,
          min(ct$x[2,] * ct$x[3,] / 4, vario.n * min(ct$x[2,]) + 1),
          min(ct$x[2,])),
 vario.n=5,
 sort=TRUE,
 fft.m = c(65, 86), ## in % of range of l.lambda
 fft.max.length=Inf,
 fft.max.regr=150000,
 fft.shift = 50, # in %; 50:WOSA; 100: no overlapping
 method=c("variogram", "fft"),
 mode=c("plot", "interactive"),
 pch=16, cex=0.2, cex.main=0.85,
 printlevel = RFoptions()$general$printlevel,
 height=3.5,
 ...)
```

Arguments

x	matrix of coordinates, or vector of x coordinates; if x is not given a grid with unit grid length is assumed
y	vector of y coordinates
z	vector of z coordinates
data	the values measured.
grid	determines whether the vectors x, y, and z should be interpreted as a grid definition, see Details. grid does not apply for T.

bin	sequence of bin boundaries for the empirical variogram
vario.n	first vario.n value of the empirical variogram are used for the regression fit that are not NA.
sort	If TRUE then the coordinates are permuted such that the largest grid length is in x-direction; this is of interest for algorithms that slice higher dimensional fields into one-dimensional sections.
fft.m	numeric vector of two components; interval of frequencies for which the regression should be calculated; the interval is given in percent of the range of the frequencies in log scale.
fft.max.length	The first dimension of the data is cut into pieces of length fft.max.length. For each piece the FFT is calculated and then the average for all pieces is taken. The pieces may overlap, see the argument fft.shift.
fft.max.regr	If the fft.m is too large, parts of the regression fit will take a very long time. Therefore, the regression fit is calculated only if the number points given by fft.m is less than fft.max.regr.
fft.shift	This argument is given in percent [of fft.max.length] and defines the overlap of the pieces defined by fft.max.length. If fft.shift=50 the WOSA estimator is given; if fft.shift=100 no overlap exist.
method	list of implemented methods to calculate the fractal dimension; see Details
mode	character. A vector with components 'nographics', 'plot', or 'interactive': 'nographics' no graphical output 'plot' the regression line is plotted 'interactive' the regression domain can be chosen interactively Usually only one mode is given. Two modes may make sense in the combination c("plot", "interactive") in which case all the results are plotted first, and then the interactive mode is called. In the interactive mode, the regression domain is chosen by two mouse clicks with the left mouse; a right mouse click leaves the plot.
pch	vector or scalar; sign by which data are plotted.
cex	vector or scalar; size of pch.
cex.main	The size of the title in the regression plots.
printlevel	integer. If printlevel is 0 nothing is printed. If printlevel=1 error messages are printed. If printlevel=2 warnings and the regression results are given. If printlevel>2 tracing information is given.
height	height of the grahics window
...	graphical arguments

Details

The function calculates the fractal dimension by various methods:

- variogram method
- Fourier transform

Value

The function returns a list with elements `vario`, `fft` corresponding to the 2 methods given in the Details.

Each of the elements is itself a list that contains the following elements.

`x`	the x-coordinates used for the regression fit
`y`	the y-coordinates used for the regression fit
`regr`	the return list of the `lm`.
`sm`	smoothed curve through the (x,y) points
`x.u`	`NULL` or the restricted x-coordinates given by the user in the interactive plot
`y.u`	`NULL` or y-coordinates according to `x.u`
`regr.u`	`NULL` or the return list of `lm` for `x.u` and `y.u`
`D`	the fractal dimension
`D.u`	`NULL` or the fractal dimension corresponding to the user's regression line

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

variogram method

- Constantine, A.G. and Hall, P. (1994) Characterizing surface smoothness via estimation of effective fractal dimension. *J. R. Statist. Soc. Ser. B* **56**, 97-113.

fft

- Chan, Hall and Poskitt (1995)

See Also

`RMmodel`, `RFhurst`

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- seq(0, 10, if (interactive()) 0.001 else 1)
z <- RFsimulate(RMexp(), x)
if (interactive()) str(RFfractaldim(data=z))
```

RFfunction *Evaluation operators (RF commands)*

Description

Here, all the RF_name_ commands are listed.

Models that are treated internally as operators on RMmodels

The user's RMmodel is supplemented internally by operators that are tacitely assumed, e.g. RPgauss.

Further completions of the user's model determine what should be done with the model, e.g. calculation of the covariance (RFcov). The following list gives those RFfunctions that have an internal represenation as completion to the user's model.

User's function	Internal task
RFcov	assigns to a covariance model the covariance values at given locations
RFcovmatrix	assigns to a covariance model the matrix of covariance values at given locations
RFfctn	assigns to a model the value of the function at given locations. In case of a covariance model RFfctn i
RFdistr	generic function assigning to a distribution family various values of the distribution random sample)
RFpseudovariogram	assigns to a model the values of the pseudo variogram at given locations
RFsimulate	assigns to a model a realisation of the corresponding random field
RFvariogram	assigns to a model the values of the (cross-)variogram at given locations

Estimation and Inference

User's function	Description
RFcrossvalidate	cross validation for Gaussian fields
RFempiricalvariogram	empirical variogram
RFfit	(maximum likelihood) fitting of the parameters
RFinterpolate	'kriging' and 'imputing'
RFratiotest	likelihood ratio test for Gaussian fields

Graphics for Gaussian fields

RFgui	educational tool for * manual selection of a covariance model * manual fitting to the empirical variogram
RFfractaldim	determination of the fractal dimension
RFhurst	determination of the Hurst effect (long range dependence)

Coordinate transformations

RFearth2cartesian	transformation of earth coordinates to cartesian coordinates
RFearth2dist	transformation of earth coordinates to Euclidean distances

Information from and to RandomFields

RFgetMethodNames	currently implemented list of simulation methods
RFgetModel	returns the model used in a RFfunction, with some more details
RFgetModelInfo	similar to RFgetModel, but with detailed information on the implementation
RFgetModelNames	lists the implemented models
RFoptions	options of package RandomFields

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RC, RM, RP, RR, RMmodelgenerator

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

z <- RFsimulate(model=RMexp(), 1:10)
RFgetModel(RFsimulate, show.call = TRUE)  # user's definition
RFgetModel(RFsimulate, show.call = FALSE) # main internal part
```

RFgetMethodNames *Simulation Techniques*

Description

RFgetMethodNames prints and returns a list of currently implemented methods for simulating Gaussian random fields and max stable random fields

Usage

```
RFgetMethodNames()
```

Details

By default, RFsimulate automatically chooses an appropriate method for simulation. The method can also be set explicitly by the user via RFoptions, in particular by passing gauss.method=_a valid method string_ as an additional argument to RFsimulate or by globally changing the options via RFoptions(gauss.method=_a valid meth
The following methods are available:

- (random spatial) Averages
  – details soon
- Boolean functions.
  See marked point processes.
- circulant embedding.
  Introduced by Dietrich & Newsam (1993) and Wood and Chan (1994).

  Circulant embedding is a fast simulation method based on Fourier transformations. It is garantueed to be an exact method for covariance functions with finite support, e.g. the spherical model.

  See also cutoff embedding and intrinsic embedding for variants of the method.
- cutoff embedding.
  Modified circulant embedding method so that exact simulation is garantueed for further covariance models, e.g. the whittle matern model. In fact, the circulant embedding is called with the cutoff hypermodel, see RMmodel, and $A = B$ there. cutoff embedding halfens the maximum number of elements models used to define the covariance function of interest (from 10 to 5).

  Here multiplicative models are not allowed (yet).
- direct matrix decomposition.
  This method is based on the well-known method for simulating any multivariate Gaussian distribution, using the square root of the covariance matrix. The method is pretty slow and limited to about 8000 points, i.e. a 20x20x20 grid in three dimensions. This implementation can use the Cholesky decomposition and the singular value decomposition. It allows for arbitrary points and arbitrary grids.
- hyperplane method.
  The method is based on a tessellation of the space by hyperplanes. Each cell takes a spatially constant value of an i.i.d. random variables. The superposition of several such random fields yields approximatively a Gaussian random field.
- intrinsic embedding.
  Modified circulant embedding so that exact simulation is garantueed for further *variogram* models, e.g. the fractal brownian one. Note that the simulated random field is always *non-stationary*. In fact, the circulant embedding is called with the Stein hypermodel, see RMmodel, and $A = B$ there.

  Here multiplicative models are not allowed (yet).
- Marked point processes.
  Some methods are based on marked point process $\Pi = \bigcup[x_i, m_i]$ where the marks m_i are deterministic or i.i.d. random functions on R^d.

  - `add.MPP` (Random coins).
    Here the functions are elements of the intersection $L_1 \cap L_2$ of the Hilbert spaces L_1 and L_2. A random field Z is obtained by adding the marks:

$$Z(\cdot) = \sum_{[x_i, m_i] \in \Pi} m_i(\cdot - x_i)$$

    In this package, only stationary Poisson point fields are allowed as underlying unmarked point processes. Thus, if the marks m_i are all indicator functions, we obtain a Poisson random field. If the intensity of the Poisson process is high we obtain an approximative Gaussian random field by the central limit theorem - this is the `add.mpp` method.
  - `max.MPP` (Boolean functions).
    If the random functions are multiplied by suitable, independent random values, and then the maximum is taken, a max-stable random field with unit Frechet margins is obtained - this is the `max.mpp` method.

- `nugget`.
  The method allows for generating a random field of independent Gaussian random variables. In the isotropic case and if the simple notation of a model (with `model` and `param`) is used, this method is called automatically if the nugget effect is positive except the method `"circulant embedding"` or `"direct"` have been explicitely.
  The method has been extended to zonal anisotropies, see also argument `nugget.tol` in `RFoptions`.
- `particular` method
  – details missing –
- Random coins.
  See marked point processes.
- `sequential` This method is programmed for spatio-temporal models where the field is modelled sequentially in the time direction conditioned on the previous k instances. For $k = 5$ the method has its limits for about 1000 spatial points. It is an approximative method. The larger k the better. It also works for certain grids where the last dimension should contain the highest number of grid points.
- `spectral TBM` (Spectral turning bands).
  The principle of `spectral TBM` does not differ from the other turning bands methods. However, line simulations are performed by a spectral technique (Mantoglou and Wilson, 1982).
  The standard method allows for the simulation of 2-dimensional random fields defined on arbitrary points or arbitrary grids. Here realisation is given as the cosine with random amplitude and random phase.
- `TBM2`, `TBM3` (Turning bands methods; turning layers).
  It is generally difficult to use the turning bands method (`TBM2`) directly in the 2-dimensional space. Instead, 2-dimensional random fields are frequently obtained by simulating a 3-dimensional random field (using `TBM3`) and taking a 2-dimensional cross-section. TBM3 allows for multiplicative models; in case of anisotropy the anisotropy matrices must be multiples of the first matrix or the anisotropy matrix consists of a time component only (i.e. all components are zero except the very last one).
  `TBM2` and `TBM3` allow for arbitrary points, and arbitrary grids (arbitrary number of points in each direction, arbitrary grid length for each direction).
  Note: Both the precision and the simulation time depend heavily on `TBM*.linesimustep` and `TBM*.linesimufactor` that can be set by `RFoptions`. For covariance models with larger values of the scale parameter, `TBM*.linesimufactor=2` is too small.

The turning layers are used for the simulations with time component. Here, if the model is a multiplicative covariance function then the product may contain matrices with pure time component. All the other matrices must be equal up to a factor and the temporal part of the anisotropy matrix (right column) may contain only zeros, except the very last entry.

Value

an invisible string vector of the Gaussian methods.

Automatic selection algorithm

— details coming soon —

Note

Most methods possess additional arguments, see `RFoptions()` that control the precision of the result. The default arguments are chosen such that the simulations are fine for many models and their parameters. The example in `RFempiricalvariogram()` shows a way of checking the precision.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Gneiting, T. and Schlather, M. (2004) Statistical modeling with covariance functions. *In preparation.*

Lantuejoul, Ch. (2002) *Geostatistical simulation.* **New York:** Springer.

Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

Original work:

- Circulant embedding:

  Chan, G. and Wood, A.T.A. (1997) An algorithm for simulating stationary Gaussian random fields. *J. R. Stat. Soc., Ser. C* **46**, 171-181.

  Dietrich, C.R. and Newsam, G.N. (1993) A fast and exact method for multidimensional Gaussian stochastic simulations. *Water Resour. Res.* **29**, 2861-2869.

  Dietrich, C.R. and Newsam, G.N. (1996) A fast and exact method for multidimensional Gaussian stochastic simulations: Extensions to realizations conditioned on direct and indirect measurement *Water Resour. Res.* **32**, 1643-1652.

  Wood, A.T.A. and Chan, G. (1994) Simulation of stationary Gaussian processes in $[0,1]^d$ *J. Comput. Graph. Stat.* **3**, 409-432.

  The code used in *RandomFields* is based on Dietrich and Newsam (1996).

- Intrinsic embedding and Cutoff embedding:

  Stein, M.L. (2002) Fast and exact simulation of fractional Brownian surfaces. *J. Comput. Graph. Statist.* **11**, 587–599.

Gneiting, T., Sevcikova, H., Percival, D.B., Schlather, M. and Jiang, Y. (2005) Fast and Exact Simulation of Large Gaussian Lattice Systems in R^2: Exploring the Limits *J. Comput. Graph. Statist.* Submitted.

- Markov Gaussian Random Field:

  Rue, H. (2001) Fast sampling of Gaussian Markov random fields. *J. R. Statist. Soc., Ser. B*, **63** (2), 325-338.

  Rue, H., Held, L. (2005) *Gaussian Markov Random Fields: Theory and Applications.* Monographs on Statistics and Applied Probability, no **104**, Chapman \& Hall.

- Turning bands method (TBM), turning layers:

  Dietrich, C.R. (1995) A simple and efficient space domain implementation of the turning bands method. *Water Resour. Res.* **31**, 147-156.

  Mantoglou, A. and Wilson, J.L. (1982) The turning bands method for simulation of random fields using line generation by a spectral method. *Water. Resour. Res.* **18**, 1379-1394.

  Matheron, G. (1973) The intrinsic random functions and their applications. *Adv. Appl. Probab.* **5**, 439-468.

  Schlather, M. (2004) Turning layers: A space-time extension of turning bands. *Submitted*

- Random coins:

  Matheron, G. (1967) *Elements pour une Theorie des Milieux Poreux*. Paris: Masson.

See Also

RMmodel, RFsimulate, RandomFields.

Examples

```
RFgetMethodNames()
```

`RFgetModelInfo` *Internal information*

Description

The function returns internal information about the simulation of a random field and the calculation of covariance functions

Usage

```
RFgetModel(register, explicite.natscale, show.call=FALSE)

RFgetModelInfo(register, level = 1,
               spConform = RFoptions()$general$spConform,
               which.submodels = c("user", "internal", "both"),
               modelname = NULL)
```

Arguments

register	$0, ..., 21$ or an evaluating function, e.g. RFsimulate. Place where intermediate calculations are stored. See also section Registers in RFoptions.
explicite.natscale	logical. Advanced option. If missing, then the model is returned as stored. If FALSE then any RMnatsc is ignored. If TRUE then any RMnatsc is tried to be combined with leading RMS, or returned as such.
show.call	logical. If FALSE then the model is shown as interpreted. If TRUE then the user's input including the calling function is returned. See example below.
level	integer [0..5]; level of details, i.e. the higher the number the more details are given. If level >= 10 then the leading internal model is also given (which is, in general, not of interest by the user).
spConform	see RFoptions
which.submodels	Internally, the sub-models are represented in two different ways 'internal' and 'user'. The latter is very close to the model defined by the user.
modelname	string. If modelname is given then it returns the first appearance of the covariance model with name modelname. If meth is given then the model within the method is returned.

Details

RFgetModelInfo(register, ignore.active=TRUE) is useful for debugging and specialists' need to control the algorithm, see the examples in RFoptions and RFsimulate.

If RFoptions()$Storage=FALSE then values of the internal registers are not kept if RFsimulate has been called. Hence RFgetModelInfo cannot provide any information.

Value

List of internal information is returned.

Note

Put Storing=TRUE, see RFoptions if you like to have more internal information in case of an expected failure of an initialisation of a random field simulation.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RFsimulate

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMexp(scale=4, var=2) + RMnugget(var=3) + RMtrend(mean=1)
z <- RFsimulate(model, 1:4, storing=TRUE)
RFgetModelInfo(RFsimulate)


RFgetModel(RFsimulate, show.call=FALSE)
RFgetModel(RFsimulate, show.call=TRUE)
```

`RFgetModelNames` *Names of implemented covariance and variogram models*

Description

Displays the names of covariance and variogram models (see `RMmodel`) and returns them as a list. The user may specify and group the models according to the following properties:

- type of function (`"positive definite"`, `"negative definite"`, etc.)
- whether the function depends on two arguments (`"kernel"`) or on one argument only (`"single variable"`)
- types of isotropy
- whether the model is an operator
- whether the model is a normal scale mixture
- whether the model has a finite range covariance
- validity in certain dimensions of the coordinate space
- maximal possible dimension of the coordinate space
- uni- or multivariety

See `Details` for an explanation and `RMmodelgenerator` for possible states (values) of these properties.

Usage

```
RFgetModelNames(type = RC_TYPE, domain = RC_DOMAIN,
                isotropy = RC_ISOTROPY, operator = c(TRUE, FALSE),
                monotone = RC_MONOTONE,
                implied_monotonicities = length(monotone) == 1,
                finiterange = c(TRUE, FALSE),
                valid.in.dim = c(1, Inf),
                vdim = c(1, 5),
                group.by=NULL,
                simpleArguments = FALSE,
                internal, newnames)
```

Arguments

`type, domain, isotropy, operator, monotone, finiterange, vdim`
: see constants for the definition of `RC_TYPE`, `RC_DOMAIN`, etc. See also `RMmodelgenerator`.

`implied_monotonicities`
: logical. If `TRUE` then all the models with a stronger monotonocity than the required one are also shown.

`valid.in.dim`
: an optional integer indicating the dimension of the space where the model is valid

`group.by`
: an optional character string; must be one of `'type'`, `'domain'`, `'isotropy'`, `'operator'`, `'monotone'`, `'finiterange'`,`'maxdim'`,`'vdim'`

`simpleArguments`
: logical. if `TRUE`, only models are considered whose arguments are all integer or real valued.

`internal, newnames`
: both logical; `internal` might be also integer valued. If any of them are given, `RFgetModelNames` behaves very differently. See the Notes below.

Details

The plain call `RFgetModelNames()` simply gives back a vector of the names of all implemented covariance and variogram models and operators, i.e. members of the class `RMmodelgenerator`.

The following arguments can be specified. Then exact matches are returned.

E.g. `type="negative definite"` returns only models that are negative definite, but not positive definite. To get all negative definite models including the positive definite one, use `RFgetModelNames(type="positive` . This structure also applies for type="tail correlation fucntion" and type="negative definite".

`type` specifies the class of functions; for the meaning of the possible values see `RMmodelgenerator`

`stationarity` specifies the type of stationarity; for the meaning of the possible values see `RMmodelgenerator`

`isotropy` specifies the type of isotropy; for the meaning of the possible values see `RMmodelgenerator`

`operator` indicates whether the model is an operator, i.e. it requires at least one submodel, e.g. `RMplus` or `RMdelay` are operators; see `RMmodelgenerator`

`monotone` indicates what kind of monotonicity is known, e.g., whether the model is a normal scale mixture, the latter including `RMexp` or `RMcauchy`; see `RMmodelgenerator`

`finiterange` indicates whether the covariance of the model has finite range, e.g. `RMcircular` or `RMnugget` have covariances with finite range; see `RMmodelgenerator`

`valid.in.dim` If `valid.in.dim=n` is passed, all models which are valid in dimension n are displayed. Otherwise `valid.in.dim` should be bivariate vector giving the range of requested dimensions.

`maxdim` if a positive integer, it specifies the maximal possible dimension of the coordinate space; note that a model which is valid in dimension n is also valid in dimension $n-1$; `maxdim=-1` means that the maximal possible dimension depends on the parameters of the `RMmodel` object; `vdim=-2` means that the maximal possible dimension is adopted from the called submodels; see also `RMmodelgenerator`

vdim if a positive integer, vdim specifies, whether the model is $vdim$-variate; vdim=-1 means that being multivariate in a certain dimension depends on the parameters of the RMmodel object; vdim=-2 means that being multivariate in a certain dimension is adopted from the called submodels; see also RMmodelgenerator

If vdim is bivariate then a range is given.

group.by If group.by="propertyname" is passed, the displayed models are grouped according to propertyname.

All arguments allow also for vectors of values. In case of valid.in.dim the smallest value is taken. The interpretation is canonical.

Note that the arguments stationarity, isotropy, operator, monotone, finiterange, maxdim, vdim are also slots (attributes) of the SP4-class RMmodelgenerator.

Value

Either a vector of model names if the argument group.by is not used; or a list of vectors of model names if the argument group.by is used (with list elements specified by the categories of the grouping argument).

In case internal or newnames is given, RFgetModelNames prints a table of the currently implemented covariance functions and the matching methods. RFgetModelNames returns NULL.

Note

In case internal or newnames is given, only the values of internal, newnames and operator are considered. All the other arguments are ignored and RFgetModelNames prints a table of the currently implemented covariance functions and the matching methods:

- internal:
  if TRUE also RMmodels are listed that are internal, hence invisible to the user. Default: FALSE.
- newnames:
  The model names of version 2 of **RandomFields** and earlier can still be used in the model definitions. Namely when the list notation is chosen; see Advanced RMmodels for the latter. If the internal or newnames is given, then these old names are shown; if newnames=TRUE then also the usual names are shown. Default: FALSE.

  In fact, both internal and public models can have different variants implemented. These variants are also shown if internal has a value greater than or equal to2,
- operator:
  see above.

Here, also an indication is given, which method for simulating Gaussian random fields matches the model.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

constants, RMmodelgenerator, RMmodel, RandomFields, RC_DOMAIN, RC_ISOTROPY

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

# get vector of names of all functions
RFgetModelNames()

# any kind of positive definite functions
RFgetModelNames(type="positive definite", group.by="type")

# get vector of names of all stationary models
RFgetModelNames(type="positive definite", domain="single variable")

# get list of models grouped by the stationarity attribute
RFgetModelNames(group.by=c("type"))
```

RFgridDataFrame-class *Class* RFgridDataFrame

Description

Class for attributes in one-dimensional space.

Creating Objects

Objects can be created by using the functions RFgridDataFrame or conventional2RFspDataFrame or by calls of the form as(x, "RFgridDataFrame"), where x is of class RFgridDataFrame.

Slots

.RFparams: list of up to 5 elements;

- n is the number of repetitions of the random field contained in the data slot
- vdim gives the dimension of the values of the random field, equals 1 in most cases
- has.variance indicates whether information on the variance is available,
- coord.units gives the names of the units for the coordinates
- variab.units gives the names of the units for the variables

`data`: object of class `data.frame`, containing attribute data

`grid`: object of class `GridTopology`.

Methods

plot `signature(obj = "RFgridDataFrame")`: generates nice plots of the random field; if $space-time-dim2$, a two-dimensional subspace can be selected using the argument `MARGIN`; to get different slices in a third direction, the argument `MARGIN.slices` can be used; for more details see `plot-method` or type `method?plot("RFgridDataFrame")`

show `signature(x = "RFgridDataFrame")`: uses the show-method for class `SpatialGridDataFrame`.

print `signature(x = "RFgridDataFrame")`: identical to show-method

RFspDataFrame2conventional `signature(obj = "RFgridDataFrame")`: conversion to a list of non-**sp**-package based objects; the data-slot is converted to an array of dimension $[1 * (vdim > 1) + space - time - dimension + 1 * (n > 1)]$

coordinates `signature(x = "RFgridDataFrame")`: returns the coordinates

[`signature(x = "RFgridDataFrame")`: selects columns of data-slot; returns an object of class `RFgridDataFrame`.

[<- `signature(x = "RFgridDataFrame")`: replaces columns of data-slot; returns an object of class `RFgridDataFrame`.

as `signature(x = "RFgridDataFrame")`: converts into other formats, only implemented for target class `RFpointsDataFrame`

cbind `signature(...)`: if arguments have identical topology, combine their attribute values

as.matrix `signature(x = "RFgridDataFrame")`: converts data-slot to matrix

as.array `signature(x = "RFgridDataFrame")`: converts data-slot to array

as.vector `signature(x = "RFgridDataFrame")`: converts data-slot to vector

Details

Methods `summary` and `dimensions` are defined for the "parent"-class `RFsp`.

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>

See Also

`RFspatialGridDataFrame`, which is for point locations in higher dimensional spaces, `RFsp`

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

x <- seq(0,10,length=100)
f <- RFsimulate(model=RMgauss(), x=x, n=3)
```

```
str(f)
str(RFspDataFrame2conventional(f))
coordinates(f)[1:25,]
str(f[2]) ## selects second column of data-slot
all.equal(f, cbind(f,f)[1:3]) ## TRUE

plot(f, nmax=2)
```

RFgui *Graphical User Interface For Fitting Covariance Models And Variograms*

Description

This is a nice instructive graphical tool useful in particular for teaching classes

Usage

```
RFgui(data, x, y, same.algorithm = TRUE, ev, xcov, ycov,
      sim_only1dim=FALSE, wait = 0, ...)
```

Arguments

data	see RFempiricalvariogram. If data is given, the empirical variogram is shown
x	a sequence of the locations of the simulated process; if not given, x is determined by data and if data is not given by default values
y	a sequence of numbers if a simulation on R^d is performed. Default is y = x; see x for details.
same.algorithm	Force the picture being simulated with the same algorithm so that the pictures are always directly comparable. The disadvantage is that some models are simulated only (very) approximatively.
ev	instead of the data, the emprirical variogram itself might be passed
xcov	sequence of the locations where the covariance function is plotted
ycov	Only for anisotropic models. sequence of the locations where the covariance function is also plotted
sim_only1dim	Logical. The argument determines whether a process should be simulated on the line or on the plane
wait	integer. See details.
...	further options and control arguments for the simulation that are passed to and processed by RFoptions.

Details

If wait is negative the xterm does not wait for the tkltk-window to be finished. Further the variable RFgui.model is created in the environment .GlobalEnv and contains the currently chosen variable in the gui. RFgui always return NULL.

If wait is non-negative the xterm waits for the tkltk-window to be finished. RFgui returns invisibly the last chosen model (or NULL if no model has been chosen). RFgui idles a lot when wait=0. It idles less for higher values by sleeping about wait microseconds. Of course the handling in the tkltk window get slower also. Reasonable values for wait are within [0,1000].

same.alg = TRUE is equivalent to setting circulant.trials=1, circulant.simu_method = "RPcirculant", circulant.force=TRUE, circulant.mmin=-2.

Value

If wait < 0 the the function returns NULL else it returns the last chosen RMmodel.

If wait < 0, a side effect effect of RFgui is the creation of the variable RFgui.model on .GlobalEnv.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

Author(s) of the code: Daphne Boecker <d.boecker@gmx.de>

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

soil for a further example

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
RFgui()
```

RFhurst *Hurst coefficient*

Description

The function estimates the Hurst coefficient of a process

Usage

```
RFhurst(x, y = NULL, z = NULL, data, sort = TRUE,
 block.sequ = unique(round(exp(seq(log(min(3000, dimen[1]/5)),
 log(dimen[1]),
 len = min(100, dimen[1])))))),
 fft.m = c(1, min(1000, (fft.len - 1)/10)),
 fft.max.length = Inf, method = c("dfa", "fft", "var"),
 mode = c("plot", "interactive"),
 pch = 16, cex = 0.2, cex.main = 0.85,
 printlevel = RFoptions()$general$printlevel, height = 3.5,
 ...)
```

Arguments

x	matrix of coordinates, or vector of x coordinates
y	vector of y coordinates
z	vector of z coordinates
data	the data
sort	logical. If TRUE then the coordinates are permuted such that the largest grid length is in x-direction; this is of interest for algorithms that slice higher dimensional fields into one-dimensional sections.
block.sequ	ascending sequences of block lengths for which the detrended fluctuation analysis and the variance method is performed.
fft.m	vector of 2 integers; lower and upper endpoint of indices for the frequency which are used in the calculation of the regression line for the periodogram near the origin.
fft.max.length	if the number of points in x-direction is larger than fft.max.length then the segments of length fft.max.length are considered, shifted by fft.max.length/2 (WOSA-estimator).
method	list of implemented methods to calculate the Hurst parameter; see Details
mode	character. A vector with components 'nographics', 'plot', or 'interactive': 'nographics' no graphical output 'plot' the regression line is plotted 'interactive' the regression domain can be chosen interactively Usually only one mode is given. Two modes may make sense in the combination c("plot", "interactive") in which case all the results are plotted first, and then the interactive mode is called. In the interactive mode, the regression domain is chosen by two mouse clicks with the left mouse; a right mouse click leaves the plot.
pch	vector or scalar; sign by which data are plotted.
cex	vector or scalar; size of pch.
cex.main	font size for title in regression plot; only used if mode includes 'plot' or 'interactive'

printlevel	integer. If printlevel is 0 or 1 nothing is printed. If printlevel=2 warnings and the regression results are given. If printlevel>2 tracing information is given.
height	height of the graphics window
...	graphical arguments

Details

The function is still in development. Several functionalities do not exist - see the code itself for the current stage.

The function calculates the Hurst coefficient by various methods:

- detrended fluctuation analysis (dfa)
- aggregated variation (var)
- periodogram or WOSA estimator (fft)

Value

The function returns a list with elements dfa, varmeth, fft corresponding to the three methods given in the Details.

Each of the elements is itself a list that contains the following elements.

x	the x-coordinates used for the regression fit
y	the y-coordinates used for the regression fit
regr	the coefficients of the lm.
sm	smoothed curve through the (x,y) points
x.u	NULL or the restricted x-coordinates given by the user in the interactive plot
y.u	NULL or y-coordinates according to x.u
regr.u	NULL or the coefficients of lm for x.u and y.u
H	the Hurst coefficient
H.u	NULL or the Hurst coefficient corresponding to the user's regression line

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

detrended fluctuation analysis

- Peng, C.K., Buldyrev, S.V., Havlin, S., Simons, M., Stanley, H.E. and Goldberger, A.L. (1994) Mosaic organization of DNA nucleotides *Phys. Rev. E* **49**, 1685-1689

aggregated variation

- Taqqu, M.S. and Teverovsky, V. (1998) On estimating the intensity of long range dependence in finite and infinite variance time series. In: Adler, R.J., Feldman, R.E., and Taqqu, M.S. *A Practical Guide to Heavy Tails, Statistical Techniques an Applications.* Boston: Birkhaeuser
- Taqqu, M.S. and Teverovsky, V. and Willinger, W. (1995) Estimators for long-range dependence: an empirical study. *Fractals* **3**, 785-798

periodogram

- Percival, D.B. and Walden, A.T. (1993) *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques*, Cambridge: Cambridge University Press.
- Welch, P.D. (1967) The use of Fast Fourier Transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms *IEEE Trans. Audio Electroacoustics* **15**, 70-73.

See Also

RMmodel, RFfractaldim

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

x <- runif(1000)
if (interactive()) {
 h <- RFhurst(1:length(x), data=x)
} else {
 h <- RFhurst(1:length(x), data=x, mode = "nographics")
}
```

RFinterpolate *Interpolation methods*

Description

The function allows for different methods of interpolation. Currently only various kinds of kriging are installed.

Usage

```
RFinterpolate(model, x, y = NULL, z = NULL, T = NULL, grid, data,
              distances, dim, err.model, method = "ml", ...)
```

Arguments

model	string; covariance model, see RMmodel, or type RFgetModelNames() to get all options.
x	$(n \times d)$ matrix or vector of x coordinates, or object of class GridTopology or raster; coordinates of n points to be kriged. For more options see RFsimulateAdvanced.
y	optional vector of y coordinates
z	optional vector of z coordinates
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of T=seq(from=From, by=By, len=Len) one may also write T=c(From, By, Len).
grid	logical; determines whether the vectors x, y, and z should be interpreted as a grid definition; RandomFields can find itself the correct value in nearly all cases. See also RFsimulateAdvanced.
data	Matrix, data.frame or object of class RFsp; coordinates and response values of measurements; If a matrix or a data.frame, the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field which are kriged *separately*; if the argument x is missing, data may contain NAs, which are then replaced by the kriged values (imputing); for details on matching of variable names see RFsimulateAdvanced; if of class RFsp
distances	another alternative to pass the (relative) coordinates, see RFsimulateAdvanced.
dim	Only used if distances are given.
err.model	For conditional simulation and random imputing only. Usually err.model=RMnugget(var=var), or not given at all (error-free measurements).
method	character. A single method out of methods or sub.methods, see RFfit.
...	Two intrinsic arguements can be given: **given** matrix. If data is a matrix, then the coordinates can be given separately, namely by given where, in each row, a single location is given **MARGIN** character or integer. This argument is only given when the mean is kriged, i.e. method='M' (see RFoptions). Then it has a similar meaning as in apply: e.g. if MARGIN="T" in a space-time model, then spatial means are calculated for each temporal instance. For further optional arguments, see RFoptions.

Details

In case of intrinsic cokriging (intrinsic kriging for a multivariate random fields) the pseudo-cross-variogram is used (cf. Ver Hoef and Cressie, 1991).

Value

The value depends on the additional argument variance.return, see RFoptions.

If variance.return=FALSE (default), Kriging returns a vector or matrix of kriged values corresponding to the specification of x, y, z, and grid, and data.

data: a vector or matrix with *one* column
* grid=FALSE. A vector of simulated values is returned (independent of the dimension of the random field)
* grid=TRUE. An array of the dimension of the random field is returned (according to the specification of x, y, and z).

data: a matrix with *at least two* columns
* grid=FALSE. A matrix with the ncol(data) columns is returned.
* grid=TRUE. An array of dimension $d+1$, where d is the dimension of the random field, is returned (according to the specification of x, y, and z). The last dimension contains the realisations.

If variance.return=TRUE, a list of two elements, estim and var, i.e. the kriged field and the kriging variances, is returned. The format of estim is the same as described above. The format of var is accordingly.

Note

Important options are

- method (overwriting the automatically detected variant of kriging)
- return_variance (returning also the kriging variance)
- locmaxm (maximum number of conditional values before neighbourhood kriging is performed)
- fillall imputing estimates location by default
- varnames and coordnames in case data.frames are used to tell which column contains the data and the coordinates, respectively.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

Marco Oesting, <oesting@math.uni-mannheim.de>

Author(s) of the code:: Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

Alexander Malinowski, <malinowski@math.uni-mannheim.de>

Marco Oesting, <oesting@math.uni-mannheim.de>

References

Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.

Cressie, N.A.C. (1993) *Statistics for Spatial Data.* New York: Wiley.

Goovaerts, P. (1997) *Geostatistics for Natural Resources Evaluation.* New York: Oxford University Press.

Ver Hoef, J.M. and Cressie, N.A.C. (1993) Multivariate Spatial Prediction. *Mathematical Geology* **25**(2), 219-240.

Wackernagel, H. (1998) *Multivariate Geostatistics.* Berlin: Springer, 2nd edition.

See Also

RMmodel, RFempiricalvariogram, RandomFields,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## Preparation of graphics
if (interactive()) dev.new(height=7, width=16)
RFoptions(always_close_screen=FALSE)

## creating random variables first
## here, a grid is chosen, but does not matter
p <- 3:8
points <- as.matrix(expand.grid(p,p))
model <- RMexp() + RMtrend(mean=1)
data <- RFsimulate(model, x=points)
plot(data)
x <- seq(0, 9, 0.25)

## Simple kriging with the exponential covariance model
model <- RMexp()
z <- RFinterpolate(model, x=x, y=x, data=data)
plot(z, data)

## Simple kriging with mean=4 and scaled covariance
model <- RMexp(scale=2) + RMtrend(mean=4)
z <- RFinterpolate(model, x=x, y=x, data=data)
plot(z, data)


## Ordinary kriging
model <- RMexp() + RMtrend(mean=NA)
z <- RFinterpolate(model, x=x, y=x, data=data)
plot(z, data)




close.screen(all = TRUE)
```

RFoldstyle *RFoldstyle*

Description

This functions is written only for package writers who have based their code on RandomFields version 2.

It avoids warnings if the old style is used, and sets `spConform = FALSE`.

Usage

```
RFoldstyle(old=TRUE)
```

Arguments

`old`	logical

Value

`NULL`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

See 'version2' for details on the commands of version 2.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

GaussRF(x=1:10, model="exp", param=c(0,1,0,1), grid=TRUE)

RFoldstyle()
GaussRF(x=1:10, model="exp", param=c(0,1,0,1), grid=TRUE)
```

RFoptions *Setting control arguments*

Description

RFoptions sets and returns control arguments for the analysis and the simulation of random fields

Usage

```
RFoptions(..., no.readonly = TRUE)
```

Arguments

...	arguments in tag = value form, or a list of tagged values.
no.readonly	If RFoptions is called without argument then all arguments are returned in a list. If no.readonly=TRUE then only rewritable arguments are returned.

Details

The subsections below comment on
1. general**: General options**
2. br**: Options for Brown-Resnick Fields**
3. circulant**: Options for circulant embedding methods** RPcirculant
4. coords**: Options for coordinates and units**
5. direct**: Options for simulating by simple matrix decomposition**
6. distr**: Options for distributions, in particular** RRrectangular
7. empvario**: Options for calculating the empirical variogram**
8. fit**: Options for** RFfit**,** RFratiotest**, and** RFcrossvalidate
9. gauss**: Options for simulating Gaussian random fields**
10. graphics**: Options for graphical output**
11. gui**: Options for** RFgui
12. hyper**: Options for simulating hyperplane tessellations**
13. krige**: Options for Kriging**
14. maxstable**: Options for simulating max-stable random fields**
15. mpp**: Options for the random coins (shot noise) methods**
16. nugget**: Options for the nugget effect**
17. registers**: Register numbers**
18. sequ**: Options for the sequential method**
19. special**: Options for some special methods**
20. spectral**: Options for the spectral (turning bands) method**
21. tbm**: Options for the turning bands method**
22. internal**: Internal**
General comments

1. General options

`allowdistanceZero` boolean. Only used in `RFinterpolate` and in `RFfit`. If true, then multiple observations or identical locations are allowed within a single data set. In this case, the coordinates are slightly scattered, so that the points have some tiny distances.

Default: `FALSE`.

`cPrintlevel` `cPrintlevel` is automatically set to `printlevel` when `printlevel` is changed.

0 : no message
1 : important (error) messages and warnings
2 : less important messages
3 : recursive call tracing
4 : details of recursive call tracing
5 : function flow information of large functions
6 : errors that are internally treated
7 : details on building up the covariance structure
8 : details on taking the square root of the covariance matrix
9 : details on intermediate calculations
10 : further details on intermediate calculations

Note that `printlevel` works on the R level whereas `cPrintlevel` works on the C level.

Default: 1

`detailed_output` logical. if `TRUE` some function, e.g. `RFcrossvalidate` will return additional information.

`every` integer. if greater than zero, then every `every`th iteration is printed if simulated by TBM or random coin method. The value zero means that nothing is printed.

Default: `0`

`exactness` logical or NA.

- `TRUE`: add.MPP, hyperplanes and all turning bands methods are excluded. If the circulant embedding method is considered as badly behaved, then the matrix decomposition methods are preferred.
- `FALSE`: if the circulant embedding method is considered as badly behaved or the number of points to be simulated is large, the turning bands methods are rather preferred.
- `NA`: approximative non-exact methods are excluded, i.e. TBM2 if the Abel transform of the covariance function cannot be given explicitely.

Default: `NA` .

`expected_number_simu` positive integer which is usally set internally as the value of the argument `n` in `RFsimulate`. The argument `expected_number_simu` should be set only by an advanced users and only if `RFsimulate` will be called with argument `n` alone.

`gridtolerance` used in `RFsimulate` to see if the coordinates build a grid for x, y, z, T-values. This argument is also used in case of conditional simulation where the data locations might ly on a grid.

Default: `1e-6`

asList logical. Lists of arguments are treated slightly different from non-lists. If `asList=FALSE` they are treated the same way as non-lists. This options being set to `FALSE` after calling `RFoptions` it should be set as first element of a list.

Default: `TRUE`

`matrix_inversion` vector of integers gives the methods (and their sequence) to inverse a matrix:

0 : Choleskey decomposition
1 : QR decomposition
2 : SVD

This option is currently used only in `RFinterpolate`.

Default: `c(0, 2)`.

`matrix_tolerance` In case of SVD, it is checked whether the eigenvalue is less than `matrix_tolerance`. Then the eigenvalue is not inverted, but the inverse is set to zero.

Default: 1e-6.

`modus_operandi` character. One of the values `"careless"`, `"sloppy"`, `"easygoing"`, `"normal"`, `"precise"`, `"pedantic"`, `"neurotic"` . **This argument is in an experimental stage and its definition and effects will change very likely in near future.** This argument sets a lot of argument at once related to estimation and simulation. `"careless"` prefers rather fast algorithms, but the results might be very rough approximations. By way of contrast, `"neurotic"` will try very hard to return exact result at the cost of hugh computing times.

Default: `"normal"`

`na_rm_lines` logical. If `TRUE` then a line of the data that contains a `NA` value is deleted. Otherwise it is tried to deal with the `NA` value at higher costs of computing time.

Default: `FALSE`.

`pch` character. `RFfit`: shown before evaluating any method; if `pch!=""` then one or two additional steps in the MLE methods are marked by "+" and "#".

Simulation:

The character is printed after each performed simulation if more than one simulation is performed at once. If `pch='!'` then an absolute counter is shown instead of the character. If `pch='%'` then a counter of percentages is shown instead of the character. Note that also '^H's are printed in the last two cases, which may have undesirable interactions with some few other R functions, e.g. `Sweave`.

Default: `'*'`.

`practicalrange` logical or integer. If not `FALSE` the range of primitive covariance functions is adjusted so that cov(1) is zero for models with finite range. (Operators are too complex to be adjusted; for anisotropic covariance the practical range is not well defined.)

The value of cov(1) is about 0.05 (for `scale=1`) for models without range. See `RMmodel` or type
`RFgetModelNames(type="positive definite",domain="single variable", isotropy="isotropic", operator=FALSE, vdim=1)`
for the list of primitive models.

- `FALSE` : the practical range ajustment is not used.
- `TRUE` : `practicalrange` is applicable only if the value is known exactly, or, at least, can be approximated by a closed formula.
- `2` : if the practical range is not known exactly it is approximated numerically.

Default: `FALSE` .

`printlevel` If `printlevel`≤ 0 there is not any output on the screen. The higher the number the more tracing information is given.

0 : no message
1 : important (error) messages and warnings
2 : less important messages
3 : recursive call tracing
4 : details of recursive call tracing
5 : function flow information of large functions
6 : errors that are internally treated
7 : details on intermediate calculations
8 : further details on intermediate calculations

Default: 1

seed integer. If NULL or NA set.seed is **not** called. Otherwise, set.seed(seed) is set before simulations are performed, e.g. by RFsimulate or RFdistr.

If the argument is set locally, i.e., within a function, it has the usual local effect. If it is set globally, i.e. by RFoptions the seed is fixed for **all subsequent** calls.

If the number of simulations n is greater than one and if RFoptions(seed=seed) is set, the ith simulation is started with the seed 'seed$+i-1$'.

Note also that RFratiotest has its own argument seed with a slightly different meaning.

spConform logical. spConform=TRUE might be used by a standard user as this allows the comfortable use of plot, for instance, while spConform=FALSE is **much** faster and and consumes **much less memory**, hence might be used by programmers or advanced users.

Details: if spConform=TRUE then RFsimulate and many other functions return an sp-object (which is an S4 object). Otherwise, matrices or lists are returned as defined in RandomFields 2.0, see the manuals for the specific functions. Frequently, the latter have now a class attribute to make the output nicer.

Note: for large data sets (to be generated), spConform=TRUE should **not** be used.

Default: TRUE

skipchecks logical. If TRUE, several checks whether the given parameter values and the dimension are within the allowed range is skipped. Do not change the value of this variable except you really know what you do.

Default: FALSE $

storing Logical. If FALSE then the intermediate results are destroyed after the simulation of the random field(s) or if an error had occured. If storing=TRUE, then additional simulations can be performed by calling RFsimulate with at most the argument n. This call can then be much faster, but the a rather large amount of memory could be kept.

When storing turned from TRUE to FALSE by global call then all registers are deleted. Advanced: With RFoptions(storing=list(FALSE, register,model_register)) single registers can be deleted.

Default: FALSE

Ttriple Logical or NA. If TRUE, then triple for the time argument T is expected, containing start, step (by), length. If FALSE a sequence on a grid is expected. If NA then the decision is automatic, but will lead to an error if ambiguous.

vdim_close_together logical. Used especially in functions that create covariance matrices. If the model is multivariate, then two ways of ordering the matrix exist. To consider first

all variables at a certain location (`vdim_close_together=TRUE`) or to consider first all locations keeping the variable fixed (`vdim_close_together=FALSE`). Note that several simulation methods rely on the value `FALSE`, so that these methods will not work anymore if `vdim_close_together=TRUE`.

Default: `FALSE`.

2. Options for Brown-Resnick Fields

`corr_factorr` to do

`deltaAM` to do

`maxtrendmem` integer; the maximal number of trends for shifted locations that may be stored at the same time when simulating BR processes via `RPbrshifted`; if `maxtrendmem` is large, multiple trend evaluation may be avoided.

Default: `1e8` .

`meshsize` positive; width of the grid on which the shape functions in the M3 representation of BR processes are simulated; only used for simulation of BR processes via `RPbrmixed`.

Default: `0.1` .

`optim_mixed` `0, 1, 2`; only used for simulation of BR processes via `RPbrmixed`.

If `optim_mixed=0`, the arguments `lambda` and `areamat` of `RPbrshifted` are used for the simulation.

If `optim_mixed=1`, `lambda` is estimated for `areamat=1`.

If `optim_mixed=2`, `areamat` is optimized and `lambda` is estimated.

Default: `1` .

`optim_mixed_maxpoints` positive integer; only used for simulation of BR processes via `RPbrmixed` with `optim_mixed>0`. Maximal number of Poisson points used for the optimization of `areamat` and the estimation of `lambda`.

Default: `10000` .

`optim_mixed_tol` value in $[0, 1]$; only used for simulation of BR processes via `RPbrmixed` with `optim_mixed=2`. In this case, `areamat` is optimized under the constraint that the probability of drawing the shape function incorrectly is bounded by `optim_mixed_tol` (cf. Oesting et al., 2012).

Default: `0.01` .

`variobound` positive; the shape functions in the mixed moving maxima representation are cut off where the variogram belonging to `phi` exceeds `variobound`.

Default: `8.0` .

`vertnumber` positive integer; for an efficient simulation of the shape functions in the M3 representation of BR processes, the component E from of the domain $[x_0, \infty] \times E$ of the underlying Poisson point process is sub-dividedinto cubes (cf. Oesting et al., 2012); `vertical` is the number of vertical breaks of E; only used for simulation of BR processes via `RPbrmixed` with `optim_mixed=2`.

Default: `7` .

3. `circulant`**: Options for circulant embedding methods, cf.** `RPcirculant`

These options influence the standard circulant embedding method, cutoff circulant embedding intrinsic circulant embedding. It can also influence `RPtbm` if the line is simulated with any circulant embedding method.

`approx_maxgrid` See RPcirculant

`approx_step` See RPcirculant

`dependent` See RPcirculant

`force` See RPcirculant

`maxGB` See RPcirculant

`maxmem` See RPcirculant

`mmin` See RPcirculant

`strategy` See RPcirculant

`tolIm` See RPcirculant

`tolRe` See RPcirculant

`trials` See RPcirculant

`useprimes` See RPcirculant

4. `coords`**: Options for coordinates and units**

`coordinate_system` One of the values `"auto"`, `"cartesian"`, `"earth"`

Default: `"auto"`

`coordnames` integer vector of length 2 or an increasing sequence of integers or character. This parameter gives the coordinate columns in a data frame, either by starting column and ending column or the sequence or by names. In the first case, single codeNAs might be included, meaning 'from the beginning' or 'until the end'. If both values are `NA`, then, depending on the context, either an error message is returned or it is assumed that the first columns give the coordinates.

`coordunits` any string. If `coordinate_system = "earth"` and longitude and latitude are transformed to 3d cartesian coordinates, `coordunits` determines whether the radius is given in kilometers (`"km"`) or miles (`"miles"`). If empty, then `"km"` is chosen.

Default: `""`

`varnames` integer vector of length 2 or an increasing sequence of integers or character. This parameter gives the data columns in a data frame, either by starting column and ending column or the sequence or by names. In the first case, single codeNAs might be included, meaning 'from the beginning' or 'until the end'. If both values are `NA`, then for keywords 'data', 'value' and 'variable' will be searched for. If none of them are found, depending on the context, either an error message is returned or it is assumed that the last columns give the data.

`new_coordunits` internal and should not be set by the user.

Default: `""`

`varunits` vector of characters. The default units of the variables.

Default: `""`

`xyz_notation` logical or `NA`. Used by RMuser only.

`NA` : automatic choice (if possible)

`false` : notation (x, y) should not be understood as as kernel definition, not as xyz notation

`true`: xyz notation used

5. `direct`**: Options for simulating by simple matrix decomposition**

`max_variab` See `RPdirect`

`root_method` See `RPdirect`

`svdtolerance` See `RPdirect`

6. `distr`**: Options for distributions, in particular** `RRrectangular`

`innermin` Default value to simulate from the `RRrectangular` distribution. The minimal length of the interval where the Taylor expansion shall be valid.

Default: `1e-20` .

`maxit` Default value to simulate from the `RRrectangular` distribution.

The number of iterative steps where the the constant of the Taylor development is increased, to find an upper bound for the given function.

Default: `20` .

`maxsteps` Default value to simulate from the `RRrectangular` distribution.

`maxsteps` is usually the number of steps in the middle part of the approximation. From this value and the length between the determined endpoints for the approximation at the origin and in the tail, the step length is calculated. If the step length is less than `minsteplen` the number of steps is reduced.

Default: `1000` .

`mcmc_n` In case of the use of MCMC it leaves out $n-1$ member of the Markov chain bevor the n member is returned. See also maxsteps.

Default: `15` .

`minsteplen` Default value to simulate from the `RRrectangular` distribution. The minimal step length for the middle part of approximation, which is a step function,

Default: `0` (i.e. not used as a criterion.)

`outermax` Default value to simulate from the `RRrectangular` distribution. The largest possible endpoint for the middle part that approximates the function by a step function. See also `innermax`.

Default: 20.

`parts` Default value to simulate from the `RRrectangular` distribution.

`parts` determines the number of tests that are performed to check whether a proposed power function is an upper bound for the given function, at the origin and the tail.

Default: `8` .

`repetitions` Minimal number of realisations to determine a quantity of the distribution by MCMC. E.g. to determine the integral value c in the paper of Oesting, Schlather, Zhou.

Default: 1000.

`safety` Default value to simulate from the `RRrectangular` distribution.

First, at the origin, the first power function of the Taylor expansion is taken as potential upper function. The constant of the power function are increased by factor $1+$`safety` and the exponent of the function similarly decreased. A number of test evaluations is performed to check whether this modified function is indeed a upper bound. If not, the considered interval at the origin is reduced iteratively, the constants of the power function further increased and the exponent decreased. If `maxit` iteration have been performed without success, the search for an upper bound fails. The search at the origin also fails if the interval around the origin has become less than `innermin`.

Similar procedure is performed for the tail.

Default: 0.08 .

7. empvario**: Options for calculating the empirical variogram**

fft Logical. Determines whether FFT should be used for data on a grid Default: TRUE.

phi0 numeric. In case of anisotropic fields directional cones are considered. The argument phi0 determines one of the boundaries, hence all boundaries for a given fixed number of cones.

Default: 0.

pseudovariogram logical. Only in the multivariate case. Whether the pseudovariogram or the crossvariogram should be calculated.

Default: FALSE.

theta0 numeric. In case of anisotropic fields directional cones are considered. The argument theta0 determines one of the boundaries, hence all boundaries for a given fixed number of cones. The argument theta0 refers to the second angle in a polar coordinate representation in 3 dimensions.

Default: 0.

tol0 numeric. Estimated values of the empirical variogram below tol0 times the grid step in the third dimension are considered to be zero. Hence the respective values are set to zero.

Default: 1e-13.

8. fit**: Options for** RFfit**,** RFratiotest**, and** RFcrossvalidate

algorithm Default: -1.

approximate_functioncalls In case the parameter vector is too close to the given bounds, the ML target function is evaluated on a grid to get a new initial value for the ML estimation. The number of points of the grid is approximately approximate_functioncalls.

Default: 50

bc_lambda_lb lower bound for the Box-Cox transformation

Default: -10.

bc_lambda_ub upper bound for the Box-Cox transformation

Default: 10.

bin_dist_factor numeric. The empirical variogram is calculated up the distance bin_dist_factor times (maximum distance among any pair of locations)

Default: 0.5.

bins vector of explicit boundaries for the bins or the number of bins for the empirical variogram (used in the LSQ target function, which is described at the beginning of the Details). Note that for anisotropic models, the value of bins might be enlarged.

Default: 20.

critical logical or signed integer.

If critical=FALSE and if the result of any maximum likelihood method is on a borderline, then the optimisation is redone in a modified way (which takes about double extra time)

If critical=TRUE and if the result of any maximum likelihood method is on a borderline, then a kind of profile likelihood optimization is done (which takes about 10 times extra time)

If critical>=2 then a kind of profile likelihood optimization is always done (which takes about n_crit times extra time) for an automatically chosen selection of the model parameters.

If critical>=3 then a kind of profile likelihood optimization is always done (which takes about n_crit times extra time) for all the parameters.

If critical<0 then none of the refined methods are performed.

Default: TRUE.

cross_refit logical. For each of the subset of the cross-validation method the parameters have to be fitted to the given model. If cross_refit is TRUE, this is done, but takes a huge amount of time. If FALSE, the model is fitted only once to the data and the value at each point is predicted with the same model given the values of the other points.

Default: FALSE.

likelihood character. types of likelihood are "auto", "full", "composite", "selection";

Default: "auto"

lowerbound_scale_factor The lower bound for the scale is determined as

(minimum distance between different pairs of points) / lowerbound_scale_factor.

Default: 3.

lowerbound_scale_ls_factor For the LSQ target function a different lower bound for the scale is used. It is determined as

(minimum distance between different pairs of points) / lowerbound_scale_ls_factor.

Default: 5.

lowerbound_sill absolute lower bound for variance and nugget. See lowerbound_var_factor.

Default: 1E-10.

lowerbound_var_factor The lower bound for the nugget and the variance is determined as var(data) / lowerbound_var_factor. If a standard model definition is given and either the nugget or the variance is fixed, the parameter to be estimated must also be greater than lowerbound_sill.

Default: 10000.

maxmixedvar upper bound for variance in a mixed model; so, the covariance model for mixed model part might be calibrated appropriately

max_neighbours integer. Maximum number of locations (with depending values) that are allowed.

Default: 5000.

minbounddistance If any value of the parameter vector returned from the ML estimation is closer than minbounddistance to any of the bounds or if any value has a relative distance smaller than minboundreldist, then it is assumed that the MLE algorithm has dropped into a local minimum, and it will be continued with evaluating the ML target function on a grid, cf. the beginning paragraphs of the Details.

Default: 0.001.

minboundreldist relative distance to the bounds below which a part of the algorithm is considered as having failed. See minbounddistance.

Default: 0.02.

min_diag Minimal value of any estimated diagonal matrix element.

Default: 1e-7.

`n_crit` integer. The approximate profiles that are considered.

Default: `10`.

`nphi` scalar or vector of 2 components. If it is a vector then the first component gives the first angle of the xy plane and the second one gives the number of directions on the half circle. If scalar then the first angle is assumed to be zero. Note that a good estimation of the variogramm by LSQ with a anisotropic model a large value for `ntheta` might be needed (about 20).

Default: `1`.

`ntheta` scalar or vector of 2 components. If it is a vector then the first component gives the first angle in the third direction and the second one gives the number of directions on the half circle. If scalar then the first angle is assumed to be zero.

Note that a good estimation of the variogramm by LSQ with a anisotropic model a large value for `ntheta` might be needed (about 20).

Default: `1`.

`ntime` scalar or vector of 2 components. if `ntimes` is a vector, then the first component are the maximum time distance (in units of the grid length `T[3]`) and the second component gives the step size (in units of the grid length `T[3]`). If scalar then the step size is assumed to 1 (in units of the grid length `T[3]`).

Default: `20`.

`only_users` boolean. If true then only `users_guess` is used as a starting point for the fitting algorithms

Default: `FALSE`.

`optimiser` Optimiser used in `RFfit` and takes one of the values `"optim"`, `"optimx"`, `"soma"`, `"nloptr"`, `"GenSA"`, `"minqa"`, `"pso"` or `"DEoptim"`.

Default: `"optim"`.

`optim_var_elimination` This argument takes the values `'never'`, `'respect bound'`, `'try'`, `'yes'`, and should only be set by the advanced user. Background of this option is that a global variance can optimized analytically.

The meaning of the values is as follows.

- `'never'` A global variance is never tried to be eliminated
- `'respect bound'` A global variance is eliminated if such a variance is detected and the user did not indicate bounds for the parameters.
- `'try'` A global variance is eliminated if such a variance is detected.
- `'yes'` A global variance is tried to be eliminated altough the algorithm did not find an indication. Here, the full responsibility is left to the user. (This option might make sense if `transform` is given.) This option is only overwritten when it does not make sense, e.g. no variance is estimated.

Default: `'respect bound'`.

`refine_onborder` logical. If `TRUE` and an estimated parameter of the model is close to the boundary, a second search for the optimum is started.

Default: `TRUE`

`minmixedvar` lower bound for variance in a mixed model; so, the covariance model for mixed model part might be calibrated appropriately

Default: 1/1000

`solvesigma` Logical. – experimental stage! If a mixed effect part is present where the variance has to be estimated, then this variance parameter is solved iteratively within the profile likelihood function, if `solvesigma=TRUE`.This makes sense if the number of independent variables is very small. If `solvesigma=FALSE` then the variance parameter is treated as any other parameter to be estimated.

Default: `FALSE`.

`ratiotest_approx` logical. if `TRUE` the approximative formula that twice the difference of the likelihoods follow about a χ^2 distribution is used. The parameter of freedom equals the number of parameters to be estimated for the covariance function, including those for the covariates.

Default: `TRUE`

`reoptimise` logical. If `TRUE && !only_users` then at a very last step, the optimisation is redone with currently best parameters and likelihood as scale parameter for `optim`.

Default: `TRUE`.

`scale_max_relative_factor` If the initial scale value for the ML estimation obtained by the LSQ target function is less than $(minimumdistancebetweendifferentpairsofpoints)/$ `scale_max_relative_factor`

a warning is given that probably a nugget effect is present. Note: if `scale_max_relative_factor` is greater than `lowerbound_scale_ls_factor` then no warning is given as the scale has the lower bound $(minimumdistancebetweendifferentpairsofpoints)/$ `lowerbound_scale_ls_factor`.

Default: `1000`

`scale_ratio` `RFfit` uses `parscale` and `fnscale` in the calls of `optim`. As these arguments should have the magnitude of the estimated values, `RFfit` checks this by calculating the absolute log ratios. If they are larger than `scale_ratio`, `parscale` and `fnscale` are reset and the optimisation is redone.

Default: `0.1`.

`shortnamelength` The names of the variables in the returned table are abbreviated by taking the first `shortnamelength` letters.

Default: `4`.

`sill` Additionally to estimating `nugget` and `variance` separately, they may also be estimated together under the condition that `nugget + variance = sill`. For the latter a finite value for `sill` has to be supplied, and `nugget` and `variance` are set to `NA`.

`sill` is only used for the standard model.

Default: `NA`.

`smalldataset` If the number of locations is considered as small, then some more data are kept in the storage to accelerate the estimation algorithm.

Default: `2000`.

`split` logical. If `TRUE` then `RFfit` checks whether a space-time covariance model or a multivariate covariance model can be split into components, so that certain parameters can be estimated separately.

Default: `TRUE`.

`splitn_neighbours` integer. In case the maximum number of locations `maxn` is exceeded, then `RFfit` tries to split the data set into parts of size `split` or less, but never more than `maxn`.

Default: `c(3000, 200, 1000)`.

splitfactor_neighbours The total number of neighbouring boxes in each direction $1+2$splitfactor, including the current box itself.

Default: 2.

split_refined logical. If TRUE then also submodels are fitted if splitted. This takes more time, but anova and RFratiotest, for instance, will give additional information.

Default: TRUE.

upperbound_scale_factor The upper bound for the scale is determined as

upperbound_scale_factor * (maximum distance between all pairs of points).

Default: 3.

upperbound_var_factor The upper bound for the variance and the nugget is determined as upperbound_var_factor * var(data)

Default: 10.

use_naturalscaling logical. Only used if model is given in standard (simple) way. If TRUE then *internally*, rescaled covariance functions will be used for which cov(1)≈0.05. use_naturalscaling has the advantage that scale and the form parameters of the model get 'orthogonal', but use_naturalscaling does not work for all models.

Note that this argument does not influence the output of RFfit: the parameter vector returned by RFfit refers *always* to the standard covariance model as given in RMmodel. (In contrast to practicalrange in RFoptions.)
Advantages if use_naturalscaling=TRUE:

- scale and the shape parameter of a parameterised covariance model can be estimated better if they are estimated simultaneously.
- The estimated bounds calculated by means of upperbound_scale_factor and lowerbound_scale_factor, etc. might be more realistic.
- in case of anisotropic models, the inverse of the elements of the anisotropy matrix should be in the above bounds.

Disadvantages if use_naturalscaling=TRUE:

- For some covariance models with additional parameters, the rescaling factor has to be determined numerically. Then, more time is needed to perform RFfit.

Default: TRUE.

use_spam Should the package spam (sparse matrices) be used for matrix calculations? If TRUE **spam** is always used. If FALSE, it is never used. If NA its use is determined by the size and the sparsity of the matrix.

Default: NA.

9. gauss: Options for simulating Gaussian random fields

approx_zero Value below which a correlation is considered to be essentially zero. This argument is used to determine the practical range of covariance function with non-compact support.

Default: 0.05

direct_bestvar integer. When searching for an appropriate simuation method the matrix decomposition method (method="direct") is preferred if the number of variables is less than or equal to direct_bestvariables.

Default is 800.

loggauss See RPgauss

paired ("Antithetic pairs".) Logical. If TRUE then the second half of the simulations is logical. If TRUE then the second half of the simulations is obtained by only changing the signs of all the standard Gaussian random variables, on which the first half of the simulations is based. Default is FALSE.

stationary_only See RPgauss

10. graphics**: Options for graphical output**

always_close_screen logical. If FALSE the current device is kept as it is. Otherwise the action depends on the value of height: if height is not positive then close.screen is performed on the current device. Else the current device is closed.

Default: FALSE.

always_open_screen logical. if TRUE a new graphical window is opened for every plot if a standard graphical output is used. If NA then the value is set to interactive()

Default: TRUE.

grPrintlevel integer values 0, 1, 2. The higher the more text is shown in the plot.

Default: 1.

height real number. if height is greater than zero then it gives the height of a single figure in a plot create by **RandomFields**; always a new window is opened. See also always_close_screen.

If plots with multiple figures are shown, the height and width of the plot will be increased by a factor up the ones given by increase_upto.

Default: 6.

pdffile argument file in pdf If "" then no internal naming is performed.

Default: "".

jpegfile argument file in jpeg If "" then no internal naming is performed.

Default: "".

filenumber integer. Starting number of the file if onefile=FALSE. It is set to 0 whenever pdffile is changed and onefile=FALSE.

Default 0.

onefile about the behaviour of argument onefile in pdf

Default: FALSE.

increase_upto See height.

Default: c(3,4).

11. gui**: Options for c**RFgui

alwaysSimulate logical. If TRUE then a new random field is simulated whenever a parameter is changed. Otherwise only the covariance function or the variogram is re-plotted; simulations are performed only when the correponding button is pressed.

Default: TRUE.

simu_method "RPcirculant", "RPcutoff", "RPintrinsic", "RPtbm", "RPspectral", "RPdirect", "RPsequential", "RPaverage", "RPnugget", "RPcoins", "RPhyperplane", "RPspecific", "any method".

Default: "RPcirculant".

`size` vector of 2 components. Grid size of the simulated stochastic processes. The two components of the vector correspond to one-dimensional and two-dimensional processes, respectively.

Default: `c(1024, 64)`.

12. `hyper`: Options for simulating hyperplane tessellations

`mar_distr` integer. This argument should not be changed yet.

It codes the marginal distribution used in the simulation:

0 : uniform distribution
1 : Frechet distribution with form argument `mar.param`
2 : Bernoulli distribution (Binomial with $n = 1$) with argument `mar.param`

Default: `0` .

`mar_param` Argument used for the marginal distribution. The argument should not be changed yet.

Default: `NA` .

`maxlines` integer. Maximum number of allowed lines.

Default: `1000` .

`superpos` integer. number of superposed hyperplane tessellations.

Default: `300` .

13. `krige`: Options for Kriging

`cholesky_R` if `TRUE` cholesky decomposition is used instead of LU.

Default: `FALSE`.

`fillall` logical value for imputing. If true all the components are estimated whether they are `NA` or not.

Default: `TRUE`.

`locmaxn` Kriging is conditions on maximal `locmaxn` points. If the data contain more points, neighbourhood kriging is performed.

Default: `8000`.

`locsplitfactor` In case of neighbourhood kriging, the area is split into small boxes. The complete neighbourhood contains (2 * `locsplitfactor` +1) boxes in each direction.

Default: `2`.

`locsplitn` vector of 3 components. A box should contain no more than `locsplitn[1]` points, but never less than `locsplitn[2]`. If a box had originally less than `locsplitn[2]` points, then the box is increased until at least `locsplitn[3]` points are in the box.

Default: `c(5000, 200, 1000)`.

`method` `'S'` : simple kriging
`'O'` : ordinary kriging
`'M'` : kriging the mean
`'U'` : universal kriging
`'I'` : intrinsic kriging
`'A'` : automatic choice determined on the given model. In some situation an automatic choice is not possible and the `method` must be given explicitely.

Default: `'A'`.

`return.variance` logical. If `FALSE` the kriged field is returned. If `TRUE` a list of two elements, `estim` and `var`, i.e. the kriged field and the kriging variances, is returned.

Default: `FALSE`.

14. `maxstable`: Options for simulating max-stable random fields

`check_every` integer. In order to get a precise simulation result, by definition, the maximum must be taken, for each shape function, over alle locations of interest. Clearly, small values will not play a role. To this end, the global minimum has to be determined. The calculation of the global minimum is expensive and therefor should not be done too frequently. On the other hand, rare updates increases the computing times for taking the maximum over a single shape functions. Here, after every `check_every` considered shape function, the global minimum is calculated. It is expected that a good choice for `check_every` is in in the interval $[10, 100]$.

(For ease and for concerns of efficiency, the more adequate, local minimum is not considered.)

Default: 30 .

`density_ratio` value in $[0, 1]$. This argument is considered only if `flat=-1` and the simulation is performed on a grid. Then, the ratio between the highest and the lowest value is calculated within the convex hull of the grid. If the value is less than `density_ratio` then the grid points are considered separately. Else the density is considered to be constant in the convex hull of the grid.

Default: 0.0.

`eps_zhou` positive real number, which gives the aimed relative precision. E.g. if `eps_zhou=0.01` then the first 2 digits should be correct.

Default: 0.01

`flat` `-1, FALSE, TRUE`. The argument is considered only if the simulation is performed on a grid. If `flat` is logical, then the density is considered to `flat` in the convex hull of the grid. If `flat=-1` the choice is done automatically.

Default: -1 .

`max_gauss` The simulation of the max-stable process based on random fields uses a stopping rule that necessarily needs a finite upper endpoint of the marginal distribution of the random field. In the case of `Brown-Resnick processes`, `extremal Gaussian fields`, and `extremal t fields`, the upper endpoint is approximated by `standardmax`.

Default: `3.0` .

`max_n_zhou` positive integer. The overall constant c in the paper of Oesting, Schlather, Zhou (2014) has to be determined by MCMC, if the shape functions are random.

The two arguments, `min_n_zhou` and `max_n_zhou`, give the minimal and the maximal number of simulations that are performed. To economize computer time the values of c is partially estimated when the shape functions are simulated. If the number of shape functions is larger than the number of simulations given by `eps_zhou` then no further simulation is performed to determine c.

Default: 1000 and 10000000, respectively.

`maxpoints` positive integer; the maximal number of Poisson points to be simulated for one realization of the max-stable random field. This option will not be considered for most of the users.

Default: `2e9` .

mcmc_zhou positive integer. In case of random shape functions, an MCMC step is required. mcmc_zhou-1 equals the number of members of the MCMC chain that are left out before the next value of the chain is returned.
Default: 20

min_n_zhou see max_n_zhou

xi Extreme value index.
Default: 1.0 .

15. mpp: Options for the random coins (shot noise) methods

n_estim_E integer. Number of draws from the distribution of the scale to estimate the mean of the distribution. This is used only if the mean of the scale distribution is not explicitely given. Default: 50000 .

intensity real. Number of superposed realisations (to approximate the normal distribution; total number for all (additive) components with same anisotropy); if intensity<=0.0 then only a single value is simulated (for checking).
Default: 100 .

about_zero In certain cases (Coins,RMtruncsupport), functions are assumed to zero if the value is less than about_zero.
Default: 0.001 .

16. nugget: Options for the nugget effect
Simulating a nugget effect is per se trivial. However, it gets complicated and best methods (including direct and circulant embedding!) fail if zonal anisotropies are considered, where sets of points have to be identified that belong to the same subspace of eigenvalue 0 of the anisotropy matrix.

tol see RPnugget

17. registers: Register numbers
Model for different purposes are or can be stored at different places. They are called registers and have non-negative numbers up to 21 (currently). Some of these predefined registers are given here. Except for register it is hard to see why users should change the values.

condregister Register number for conditional simulation
Default: 20

errregister Additional register for conditional simulation if error term is envolved.
Default: 21

guiregister Register number for RFgui.
Default: 14

interpolregister Register number for kriging
Default: 19

register number in 0:9; place where intermediate calculation for random field simulation are stored; the number refers to 10 internal registers 0..9.
Changing the register number only makes sense, when two different random fields, say, are to be simulated alternatingly, several times in a row. Then the simlulation speed can be increased if several registers are used, storing=TRUE and RFsimulate is used with the only argument n.
Default: 0

18. `sequ`: Options for the sequential method

`back_steps` See `RPsequential`

`initial` See `RPsequential`

`max_variables` See `RPsequential`

19. `special`: Options for specific methods

`multicopies` Only used by `RMmult`. The covariance functions are multiplied if the corresponding independent random fields are multiplied. To get an approximative Gaussian random fields with a multiplicative covariance functions the average over `multicopies` products of random fields is calculated.

20. `spectral`: Options for the spectral (turning bands) method

`ergodic` In case of an additive model and `ergodic=FALSE`, the additive component are chosen proportional to their variance. In total `lines` are simulated. If `ergodic=TRUE`, the components are simulated separately and then added.

Default: `FALSE` .

`prop_factor` see `RPspectral`

`sigma` see `RPspectral`

`sp_grid` see `RPspectral`

`sp_lines` see `RPspectral`

21. `tbm`: Options for the turning bands method

`center` Scalar or vector. If not `NA`, the `center` is used as the center of the turning bands for `TBM2` and `TBM3`. Otherwise the center is determined automatically such that the line length is minimal. See also `points` and the examples below.

Default: `NA` .

`fulldim` positiv integer. The dimension of the space into which the simulated field is embedded. So, the value `fulldim` must be at least the dimension of the field.

Default: 3.

`grid` Logical. The angle of the lines is random if `grid=FALSE`, and $k\pi/$`lines` for k in `1:lines`, otherwise. This argument is only considered ift he spectral measure, not the density is used.

Default: `TRUE` .

`layers` Logical or integer. If `TRUE` then the turning layers are used whenever a time component is given. If `NA` the turning layers are used only when the traditional TBM is not applicable. If `FALSE` then turning layers may never be used.

Default: `TRUE` .

`lines` Number of lines used.

Default: `60` .

`linesimustep` If `linesimustep` is positive the grid on the line has lag `linesimustep`. See also `linesimufactor`.

Default: `0.0` .

`linesimufactor` `linesimufactor` or `linesimustep` must be non-negative; if `linesimustep` is positive then `linesimufactor` is ignored. If both arguments are naught then `points` is used (and must be positive). The grid on the line is `linesimufactor`-times finer than the smallest distance. See also `linesimustep`.

Default: `2.0` .

`points` integer. If greater than 0, `points` gives the number of points simulated on the TBM line, hence must be greater than the minimal number of points given by the size of the simulated field and the two paramters `TBMx.linesimufactor` and `TBMx.linesimustep`. If `points` is not positive the number of points is determined automatically. The use of `center` and `points` is highlighted in an example below.

Default: `0`.

`reduceddim` if positiv integer, then the value itself. If negativ, then the value is substracted from fulldim.

Default: `-2`.

22. `internal`: Internal options mostly for warnings and messages

All these options should not be changed by the user unless he/she really known what he/she is doing.

Most of the options below change their value in a session without the user's notice.

`stored.init` internally used logical argument. This option is closely related to `storing` which controls whether intermediate calculations should be stored to have faster repeated simulations.

This user option is internally overwritten if the user calls several simulations at once. This current value is stored in `stored.init`.

Default: `FALSE`.

`warn_ambiguous` internally used logical argument. Usually, the argument `grid` in `RFsimulate`, for instance, can or should be given. If not given, the system takes a default definition. Additionally a message is displayed in this case if `ambiguous=TRUE`.

Default: `FALSE`.

`warn_aspect_ratio` internally used logical argument. if `TRUE` then a warning is given not a standard graphical device is used and the package plots try to keep a certain aspect ratio.

Default: `TRUE`

`warn_colour_palette` internally used logical argument. If none of the packages **RColorBrewer** and **colorspace** are available and graphics are displayed, a message is displayed.

Default: `TRUE`.

`warn_coordinates` internally used logical argument. If `TRUE` then a transformation from earth coordinates to cartesian coordinates is reported.

Default: `TRUE`.

`warn_newAniso` internally used logical argument. If `newAniso=TRUE` and the argument `Aniso` is used in the model definition, then a message is displayed that the matrix `Aniso` is multiplied from the right by x, where up to Version 2.0 the argument `aniso` was available which was multiplied from the left by x.

Default: `TRUE`.

warn_new_definitions internally used logical argument. If warn_new_defintions=TRUE then a warning is returned when models are used whose definition has changed recently.

Default: TRUE.

warn_newstyle internally used logical argument. If TRUE a message is displayed the by the argument spConform=FALSE oldstyle return values are obtained instead of S4 objects.

Default: TRUE.

warn_normal_mode internally used logical argument. if TRUE then the function RFfit displays the message that other values for the option modus_operandi are available.

Default: TRUE.

warn_scale internally used logical argument. If warn_scale=TRUE then a scale less than 10 [km] is reported if earth coordinates are transformed to cartesian coordinates.

Default: TRUE.

warn_oldstyle internally used logical argument. If TRUE a warning is given if an obsolete function from Version 2 is used.

Default: TRUE.

warn_on_grid internally used logical argument. If a (one-dimensional) grid is given, but the argument grid=FALSE, e.g. in RFsimulate, this contraction is reported if warn_on_grid=TRUE

Default: TRUE.

Value

NULL if any argument is given, and the full list of arguments, otherwise.

if no.readonly=FALSE then additionally, a list called readonly is included containing
* covmaxchar: the maximum length of a model name * covnr: number of currently implemented variogram/covariance models (-1 means that none of the functions like RFsimulate, RFfit , etc., have been called yet.)
* distrmaxchar: max. name length for a distribution
* distrnr: number of currently implemented distributions
* maxdim: maximum number of dimensions for a random field
* maxmodels: maximum number of elementary models in definition of a complex covariance model
* methodmaxchar: max. name length for methods
* methodnr: number of currently implemented simulation methods

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

Oesting, M., Schlather, M. and Zhou, C. (2013) On the Normalized Spectral Representation of Max-Stable Processes on a compact set. *arXiv*, **1310.1813**

See Also

RFsimulate, RFoptionsAdvanced, RandomFields, and RFgetMethodNames.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
RFoptions()


############################################################
##                                                        ##
## use of exactness                                       ##
##                                                        ##
############################################################
x <- seq(0, 1, if (interactive()) 1/30 else 0.5)
model <- RMgauss()

for (exactness in c(NA, FALSE, TRUE)) {
  readline(paste("\n\nexactness: `", exactness, "'; press return"))
  z <- RFsimulate(model, x, x, exactness=exactness,
                  stationary_only=NA, storing=TRUE)
  print(RFgetModelInfo(which="internal")$internal$name)
}
```

RFoptionsAdvanced *Setting control arguments of* **RandomFields** – *advanced examples*

Description

Some more complex examples for the use of RFoptions are given.

Examples

```
############################################################
##                      EXAMPLE 1                         ##
## The following gives an example on the advantage of     ##
## dependent=TRUE for simulating with RPcirculant if, in a ##
## study, most of the time is spent with simulating the   ##
## Gaussian random fields. Here, the covariance at a pair  ##
## of points is estimated for n independentent repetitions ##
## and 2*n locally dependent dependent repetitions .      ##
## To get the precision, the procedure is repeated m times.##
############################################################
```

```
# In the example below, local.dependent speeds up the simulation
# by about factor 16 at the price of an increased variance of
# factor 1.5

x <- seq(0, 1, len=10)
y <- seq(0, 1, len=10)
grid.size <- c(length(x), length(y))
meth <- RPcirculant
model <- RMexp(var=1.1, Aniso=matrix(nc=2, c(2,0.1,1.5,1)))

m <- if (interactive()) 5 else 2
n <- if (interactive()) 100 else 2

# using local.dependent=FALSE (which is the default)
c1 <- numeric(m)
time <- unix.time(
  for (i in 1:m) {
    cat("", i, "out of", m, "\n")
    z <- RFsimulate(meth(model), x, y, n=n, pch="",
                    dependent=FALSE, spConform=FALSE, trials=5)
    c1[i] <- cov(z[1,length(y), ], z[length(x), 1, ])
}) # many times slower than with local.dependent=TRUE below

true.cov <- RFcov(model, t(y[c(1, length(y))]), t(x[c(length(x), 1)]))
print(time)
Print(true.cov, mean(c1), sd(c1), empty.lines=1)## true mean is zero

# using local.dependent=TRUE ...
c2 <- numeric(m)
time <- unix.time(
  for (i in 1:m) {
    cat("", i)
    z <- RFsimulate(meth(model), x, y, n=2 * n, pch="",
                    dependent=TRUE, spConform=FALSE, trials=5)
    c2[i] <- cov(z[1,length(y),], z[length(x), 1 , ])
})

print(time)                                        ## 20 times faster
Print(true.cov, mean(c2), sd(c2), empty.lines=1) ## much better results

## the sd is samller (using more locally dependent realisations)
## but it is (much) faster! Note that for n=n2 instead of n=2 * n,
## the value of sd(c2) would be larger due to the local dependencies
## in the realisations.



 ##############################################################
 ##                      EXAMPLE 2                           ##
 ## This example shows that the same realisation can be      ##
 ## obtained on different grid geometries (or point          ##
```

```
 ## configurations, i.e. grid, non-grid) using TBM          ##
 ##############################################################

x1 <- seq(-150,150,1)
y1 <- seq(-15, 15, 1)
x2 <- seq(-50, 50, 1)
model <- RPtbm(RMexp(scale=10))

RFoptions(storing=TRUE)
mar <- c(2.2, 2.2, 0.1, 0.1)
points <- 700

###### simulation of a random field on long thin stripe
z1 <- RFsimulate(model, x1, y1, center=0, seed=0,
                 points=points, storing=TRUE, spConform=FALSE)
ScreenDevice(height=1.55, width=12)
par(mar=mar)
image(x1, y1, z1, col=rainbow(100))
polygon(range(x2)[c(1,2,2,1)], range(y1)[c(1,1,2,2)],
        border="red", lwd=3)


###### definition of a random field on a square of shorter diagonal
z2 <- RFsimulate(model, x2, x2, register=1, seed=0,
                 center=0, points=points, spConform=FALSE)
ScreenDevice(height=4.3, width=4.3)
par(mar=mar)
image(x2, x2, z2, zlim=range(z1), col=rainbow(100))
polygon(range(x2)[c(1,2,2,1)], range(y1)[c(1,1,2,2)],
        border="red", lwd=3)
tbm.points <- RFgetModelInfo(level=3)$loc$totpts
Print(tbm.points, empty.lines=0) # number of points on the line
```

RFpointsDataFrame-class

Class RFpointsDataFrame

Description

Class for attributes in one-dimensional space that are not on a grid.

Creating Objects

Objects can be created by using the functions RFpointsDataFrame or conventional2RFspDataFrame or by calls of the form as(x, "RFpointsDataFrame"), where x is of class RFpointsDataFrame.

Slots

data: object of class data.frame, containing attribute data

coords: n-times-1 matrix of coordinates (each row is a point)

.RFparams: list of 2; .RFparams$n is the number of repetitions of the random field contained in the data slot, .RFparams$vdim gives the dimension of the values of the random field, equals 1 in most cases

Methods

plot signature(obj = "RFpointsDataFrame"): generates nice plots of the random field; if $space-time-dim2$, a two-dimensional subspace can be selected using the argument MARGIN; to get different slices in a third direction, the argument MARGIN.slices can be used; for more details see plot-method or type method?plot("RFpointsDataFrame")

show signature(x = "RFpointsDataFrame"): uses the show-method for class SpatialPointsDataFrame.

print signature(x = "RFpointsDataFrame"): identical to show-method

RFspDataFrame2conventional signature(obj = "RFpointsDataFrame"): conversion to a list of non-**sp**-package based objects; the data-slot is converted to an array of dimension $[1 * (vdim > 1) + space - time - dimension + 1 * (n > 1)]$

coordinates signature(x = "RFpointsDataFrame"): returns the coordinates

[signature(x = "RFpointsDataFrame"): selects columns of data-slot; returns an object of class RFpointsDataFrame.

[<- signature(x = "RFpointsDataFrame"): replaces columns of data-slot; returns an object of class RFpointsDataFrame.

as signature(x = "RFpointsDataFrame"): converts into other formats, only implemented for target class RFgridDataFrame

cbind signature(...): if arguments have identical topology, combine their attribute values

Details

Methods summary and dimensions are defined for the "parent"-class RFsp.

Author(s)

Alexander Malinowski <malinows@math.uni-goettingen.de>

See Also

RFspatialPointsDataFrame, which is for point locations in higher dimensional spaces, RFsp

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

x <- runif(100)
f <- RFsimulate(model=RMexp(), x=x, n=3)

str(f)
str(RFspDataFrame2conventional(f))
coordinates(f)[1:25,]
str(f[2]) ## selects second column of data-slot
all.equal(f, cbind(f,f)[1:3]) ## TRUE

plot(f, nmax=2)
```

RFratiotest *Likelihood ratio test*

Description

The function performs an approximate chi^2 test or a Monte Carlo likelihood ratio test based on fitgauss. Currently it only works for Gaussian random fields.

Usage

```
RFratiotest(nullmodel, alternative, x, y = NULL, z = NULL, T = NULL, grid, data,
alpha, n = 5 / alpha, seed = 0,
lower = NULL, upper = NULL, bc_lambda, methods,
sub.methods, optim.control = NULL, users.guess = NULL,
distances = NULL, dim, transform = NULL, ...)
```

Arguments

nullmodel, alternative	See Details. The set of parameters to be estimated for nullmodel should be a subset of the parameters to be estimated for alternative if alternative is given.
alpha	value in [0,1] or missing. Significance level.
n	integer. The test is based on n-1 simulations.
seed	integer. If not NULL and not NA, the .Random.seed is set to seed. Otherwise, set.seed is set to the value of RFoptions{}$general$seed if the latter is not NA.
x, y, z, T, grid, data, lower, upper, bc_lambda, methods, sub.methods, optim.control, users.guess,	see RFfit

Details

`nullmodel` (and the `alternative`) can be

- a covariance model, see `RMmodel` or type `RFgetModelNames()` to get all options.
  Depending weather the `RFoptions` `ratiotest_approx` is TRUE the the chisq approximation is performed. Otherwise a Monte Carlo ratio test is performed.
- `RFfit` or `RMmodelFit`
  Here, a chisq approximative test is always performed on the already fitted models.

`RFratiotest` tries to detect whether `nullmodel` is a submodel of `alternative`. If it fails,

- a message is printed that says that an *automatic* detection has not been possible;
- it is not guaranteed anymore that the `alternative` model returns a (log) likelihood that is at least as large as that of the `nullmodel`, even if `nullmodel` is a submodel of `alternative`. This is due to numerical optimisation which is never perfect.

Otherwise it is guaranteed that the `alternative` model has a (log) likelihood that is at least as large as that of the `nullmodel`.

Value

The test returns a message whether the null hypothesis, i.e. the smaller model is accepted. Invisibly, a list that also contains

- p, the p-value
- `n`
- `data.ratio` the log ratio for the data
- `simu.ratio` the log ratio for the simulations
- `data.fit` the models fitted to the data
- `msg` the message that is also directly returned

It has S3 class `"RFratiotest"`.

Methods

print prints the summary

summary gives a summary

Note

An important `RFoptions` is `ratiotest_approx`.

Note

Note that the likelihood ratio test may take a huge amount of time.

Note

This function does not depend on the value of RFoptions()$PracticalRange. The function RFratiotest always uses the standard specification of the covariance model as given in RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RFfit RMmodel, RandomFields, weather.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

data(soil)  ## see also ?soil
soil <- RFspatialPointsDataFrame(
 coords = soil[, c("x.coord", "y.coord")],
 data = soil[, c("moisture", "NO3.N", "Total.N",
 "NH4.N", "DOC", "N20N")],
 RFparams=list(vdim=6, n=1)
 )

model <- ~1 + RMplus(RMwhittle(scale=NA, var=NA, nu=NA), RMnugget(var=NA))
submodel <- ~1 + RMplus(RMwhittle(scale=NA, var=NA, nu=NA), RMnugget(var=0))

RFratiotest(submodel, model, data=soil["moisture"],
            modus_operandi="sloppy")
```

RFsimulate *Simulation of Random Fields*

Description

This function simulates **unconditional** random fields:

- univariate and multivariat, spatial and spatio-temporal Gaussian random fields
- fields based on Gaussian fields such as Chi2 fields or Binary fields, see RP.
- stationary Poisson fields

- stationary max-stable random fields.

It also simulates **conditional** random fields for

- univariate and multivariat, spatial and spatio-temporal Gaussian random fields

Here, only the simulation of Gaussian random fields is described. For other kind of random fields (binary, max-stable, etc.) or more sophisticated approches see RFsimulateAdvanced.

Usage

```
RFsimulate(model, x, y=NULL, z=NULL, T=NULL, grid,
 data, distances, dim, err.model, n=1, ...)
```

Arguments

model	object of class RMmodel, RFformula or formula; specifies the model to be simulated; the best is to consider the examples below, first. • if of class RMmodel, model specifies a covariance or variogram model of a Gaussian random field; type RFgetModelNames(type="CovType") for a list of available models; see also RMmodel • if of class RFformula or formula , submodel specifies a linear mixed model where random effects can be modelled by Gaussian random fields; see RFformula for details on model specification. • for (many) more options see RFsimulateAdvanced.
x	vector of x coordinates, or object of class GridTopology or raster; For more options see RFsimulateAdvanced.
y	optional vector of y coordinates
z	optional vector of z coordinates
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of T=seq(from=From, by=By, len=Len) one may also write T=c(From, By, Len).
grid	logical; RandomFields can find itself the correct value in nearly all cases. See also RFsimulateAdvanced.
data	For conditional simulation and random imputing only. If data is missing, unconditional simulation is performed. Matrix, data.frame or object of class RFsp; coordinates and response values of measurements in case that conditional simulation is to be performed; If a matrix or a data.frame, the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field; if the argument x is missing, data may contain NAs, which are then replaced by conditionally simulated values (random imputing); for details on matching of variable names see Details; if of class RFsp
distances	another alternative to pass the (relative) coordinates, see RFsimulateAdvanced.
dim	Only used if distances are given.

err.model	For conditional simulation and random imputing only. Usually err.model=RMnugget(var=var), or not given at all (error-free measurements).
n	number of realizations to generate. For a very advanced feature, see the notes in RFsimulateAdvanced.
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions

Details

By default, all Gaussian random fields have zero mean. Simulating with trend can be done by including RMtrend in the model, see the examples below.

If data is passed, conditional simulation based on simple kriging is performed:

- if of class RFsp, ncol(data@coords) must equal the dimension of the index space. If data@data contains only a single variable, variable names are optional. If data@data contains more than one variable, variables must be named and model must be given in the tilde notation resp ~ ... (see RFformula) and "resp" must be contained in names(data@data).
- If data is a matrix or a data.frame, either ncol(data) equals $(dimension of index space + 1)$ and the order of the columns is (x, y, z, T, response) or, if data contains more than one response variable (i.e. ncol(data) > (dimension of index space + 1)), colnames(data) must contain colnames(x) or those of "x", "y", "z", "T" that are not missing. The response variable name is matched with model, which must be given in the tilde notation. If "x", "y", "z", "T" are missing and data contains NAs, colnames(data) must contain an element which starts with 'data'; the corresponding column and those behind it are interpreted as the given data and those before the corresponding column are interpreted as the coordinates.
- if x is missing, RFsimulate searches for NAs in the data and performs a conditional simulation for them.

Specification of err.model: In geostatistics we have two different interpretations of a nugget effect: small scale variability and measurement error. The result of conditional simulation usually does not include the measurement error. Hence the measurement error err.model must be given separately. For sake of generality, any model (and not only the nugget effect) is allowed. Consequently, err.model is ignored when unconditional simulation is performed.

Value

By default, an object of the virtual class RFsp; result is of class RMmodel.

- RFspatialGridDataFrame if the space-time dimension is greater than 1 and the coordinates are on a grid,
- RFgridDataFrame if the space-time dimension equals 1 and the coordinates are on a grid,
- RFspatialPointsDataFrame if the space-time dimension is greater than 1 and the coordinates are not on a grid,
- RFpointsDataFrame if the space-time dimension equals 1 and the coordinates are not on a grid.

In case of a multivariate

If `n > 1` the repetitions make the last dimension.

See RFsimulateAdvanced for additional options.

Note

Several advanced options can be found in sections 'General options' and 'coords' of RFoptions. In particular, option `spConform=FALSE` leads to a simpler (and faster!) output, see RFoptions for details.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Gneiting, T. and Schlather, M. (2013) Statistical modeling with covariance functions. *In preparation.*

Lantuejoul, Ch. (2002) *Geostatistical simulation.* **New York:** Springer.

Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

See RFsimulateAdvanced for more specific literature.

See Also

RFempiricalvariogram, RFfit, RFgetModelInfo, RFgui, RMmodel, RFoptions, RFsimulateAdvanced, RFsimulate.more.examples

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again


##############################################################
## ##
## ONLY TWO VERY BASIC EXAMPLES ARE GIVEN HERE ##
## see ##
## ?RMsimulate.more.examples ##
## and ##
## ?RFsimulateAdvanced ##
## for more examples ##
## ##
##############################################################

##############################################################
## ##
## Unconditional simulation ##
## ##
```

```
############################################################

## first let us look at the list of implemented models
RFgetModelNames(type="positive definite", domain="single variable",
                iso="isotropic")

## our choice is the exponential model;
## the model includes nugget effect and the mean:
model <- RMexp(var=5, scale=10) + # with variance 4 and scale 10
 RMnugget(var=1) + # nugget
 RMtrend(mean=0.5) # and mean

## define the locations:
from <- 0
to <- 20
len <- if (interactive()) 200 else 2
x.seq <- seq(from, to, length=len)
y.seq <- seq(from, to, length=len)

simu <- RFsimulate(model, x=x.seq, y=y.seq)
plot(simu)



############################################################
## ##
## Conditional simulation ##
## ##
############################################################

# first we simulate some random values at a
# 100 random locations:
n <- if (interactive()) 100 else 2
x <- runif(n=n, min=-1, max=1)
y <- runif(n=n, min=-1, max=1)
data <- RFsimulate(model = RMexp(), x=x, y=y, grid=FALSE)
plot(data)

# let simulate a field conditional on the above data
x.seq.cond <- y.seq.cond <- seq(-1.5, 1.5, length=n)
model <- RMexp()
cond <- RFsimulate(model, x=x.seq.cond, y=y.seq.cond, data=data)
plot(cond, data)
```

RFsimulate.more.examples

Further Examples for the Simulation of Random Fields

Description

This man page will give a collection of basic examples for the use of RFsimulate.

For other kind of random fields (binary, max-stable, etc.) or more sophisticated approches see RFsimulateAdvanced.

See RFsimulate.sophisticated.examples for further examples.

See Also

RFsimulate, RFsimulateAdvanced RFsimulate.sophisticated.examples

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

`RFsimulate.sophisticated.examples`
Sophisticated Examples for the Simulation of Random Fields

Description

This man page will give a collection of basic examples for the use of RFsimulate.

For other kind of random fields (binary, max-stable, etc.) or more sophisticated approches see RFsimulateAdvanced.

See Also

RFsimulate, RFsimulateAdvanced

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RFsimulateAdvanced *Simulation of Random Fields – Advanced*

Description

This function simulates **unconditional** random fields:

- univariate and multivariat, spatial and spatio-temporal Gaussian random fields
- stationary Poisson fields
- Chi2 fields
- t fields
- Binary fields
- stationary max-stable random fields.

It also simulates **conditional** random fields for

- univariate and multivariat, spatial and spatio-temporal Gaussian random fields

For basic simulation of Gaussian random fields, see RFsimulate. See RFsimulate.more.examples and RFsimulate.sophisticated.examples for further examples.

Arguments

`model` object of class `RMmodel`, `RFformula` or `formula`; specifies the model to be simulated

- if of class `RMmodel`, `model` specifies
  - the type of random field by using `RP`functions, e.g.,
    - `RPgauss`: Gaussian random field (default if none of the function in the list are given)
    - `RPsmith`: Smith model
      See `RP` for an overview.
  - the covariance or variogram model inm case of a Gaussian random field (`RPgauss`) and for fields based on Gaussian fields (e.g. `RPbernoulli`); type `RFgetModelNames(type="negative definite", group.by="type")` for a list of available models; see also `RMmodel`
  - the shape function in case of a shot noise process; type `RFgetModelNames(type='shape')` for a list of available models
- if of class `RFformula` or `formula`, `submodel` specifies a linear mixed model where random effects can be modelled by Gaussian random fields; see `RFformula` for details on model specification.

`x` matrix of coordinates, or vector of x coordinates, or object of class `GridTopology` or `raster`; if matrix, `ncol(x)` is the dimension of the index space; matrix notation is required in case of more than 3 space dimensions; in this case, if `grid=FALSE`, `x_ij` is the i-th coordinate in the j-th dimension; otherwise, if `grid=TRUE`, the columns of `x` are interpreted as gridtriples (see `grid`); if of class `GridTopology`, `x` is interpreted as grid definition and `grid` is automatically set to `TRUE`

y	optional vector of y coordinates, ignored if x is a matrix
z	optional vector of z coordinates, ignored if x is a matrix
T	optional vector of time coordinates, T must always be an equidistant vector or given in a gridtriple format (see argument grid); for each component of T, the random field is simulated at all location points
grid	logical; determines whether the vectors x, y, and z or the columns of x should be interpreted as a grid definition (see Details). If grid=TRUE, either x, y, and z must be equidistant vectors in ascending order or the columns of x must be given in the gridtriple format: c(from, stepsize, len). Note: if grid is not given, RFsimulate tries to guess what is meant. c(from, stepsize, len) (see Details)
data	matrix, data.frame or object of class RFsp; coordinates and response values of measurements in case that conditional simulation is to be performed; if a matrix or a data.frame, the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field; if x is missing, data may contain NAs, which are then replaced by conditionally simulated values; if data is missing, unconditional simulation is performed; for details on matching of variable names see Details; if of class RFsp
err.model	same as model; gives the model of the measurement errors for the measured data (which must be given in this case!), see Details, err.model=NULL (default) corresponds to error-free measurements, the most common alternative is err.model=RMnugget(); ignored if data is missing
distances	object of class dist representing the upper trianguar part of the matrix of Euclidean distances between the points at which the field is to be simulated; only applicable for stationary and isotropic models; if not NULL, dim must be given and x, y, z and T must be missing or NULL. If distances are given, the current value of spConform, see RFoptions, is ignored and instead spConform=FALSE is used. (This fact may change in future.)
dim	integer; space or space-time dimension of the field
n	number of realizations to generate
...	further options and control arguments for the simulation that are passed to and processed by RFoptions

Details

RFsimulate simulates different classes of random fields, controlled by the wrapping model.

If the wrapping function of the model argument is a covariance or variogram model (i.e., one of list obtained by RFgetModelNames(type="negative definite", group.by="type"), by default, a Gaussian field with the corresponding covariance structure is simulated. By default, the simulation method is chosen automatically through internal algorithms. The simulation method can be set explicitly by enclosing the covariance function with a method specification.

If other than Gaussian fields are to be simulated, the model argument must be enclosed by a function specifying the type of the random field.

There are different possibilities of passing the locations at which the field is to be simulated. If grid=FALSE, all coordinate vectors (except for the time component T) must have the same length

and the field is only simulated at the locations given by the rows of x or of `cbind(x, y, z)`. If T is not missing, the field is simulated for all combinations $(x[i,], T[k])$ or $(x[i], y[i], z[i], T[k])$, $i = 1, ...,$`nrow(x)`, $k = 1, ...,$`length(T)`, even if `model` is not explicitly a space-time model.
If `grid=TRUE`, the vectors `x`, `y`, `z` and `T` or the columns of `x` and `T` are interpreted as a grid definition, i.e. the field is simulated at all locations (x_i, y_j, z_k, T_l), as given by `expand.grid(x, y, z, T)`. Here, "grid" means "equidistant in each direction", i.e. all vectors must be equidistant and in ascending order. In case of more than 3 space dimensions, the coordinates must be given in matrix notations. To enable different grid lengths for each direction in combination with the matrix notation, the "gridtriple" notation `c(from, stepsize, len)` is used: If `x`, `y`, `z`, `T` or the columns of `x` are of length 3, they are internally replaced by `seq(from=from, to=from+(len-1)*stepsize, by=stepsize)` , i.e. the field is simulated at all locations
`expand.grid(seq(x$from, length.out=x$len, by=x$stepsize), seq(y$from, length.out=y$len, by=y$stepsiz`

If `data` is passed, conditional simulation is performed.

- if of class RFsp, `ncol(data@coords)` must equal the dimension of the index space. If `data@data` contains only a single variable, variable names are optional. If `data@data` contains more than one variable, variables must be named and `model` must be given in the tilde notation `resp ~ ...` (see RFformula) and `"resp"` must be contained in `names(data@data)`.
- If `data` is a matrix or a data.frame, either `ncol(data)` equals $(dimension of index space + 1)$ and the order of the columns is (x, y, z, T, response) or, if `data` contains more than one response variable (i.e. `ncol(data) > (dimension of index space + 1)`), `colnames(data)` must contain `colnames(x)` or those of `"x", "y", "z", "T"` that are not missing. The response variable name is matched with `model`, which must be given in the tilde notation. If `"x", "y", "z", "T"` are missing and `data` contains NAs, `colnames(data)` must contain an element which starts with 'data'; the corresponding column and those behind it are interpreted as the given data and those before the corresponding column are interpreted as the coordinates.
- if `x` is missing, RFsimulate searches for NAs in the data and performs a conditional simulation for them.

Specification of `err.model`: In geostatistics we have two different interpretations of a nugget effect: small scale variability and measurement error. The result of conditional simulation usually does not include the measurement error. Hence the measurement error `err.model` must be given separately. For sake of generality, any model (and not only the nugget effect) is allowed. Consequently, `err.model` is ignored when unconditional simulation is performed.

Value

By default, an object of the virtual class RFsp; result is of class RFspatialGridDataFrame if $[space-time-dimension > 1]$ and the coordinates are on a grid, result is of class RFgridDataFrame if $[space-time-dimension = 1]$ and the coordinates are on a grid, result is of class RFspatialPointsDataFrame if $[space - time - dimension > 1]$ and the coordinates are not on a grid, result is of class RFpointsDataFrame if $[space - time - dimension = 1]$ and the coordinates are not on a grid.

The output format can be switched to the "old" array format using RFoptions, either by globally setting `RFoptions(spConform=FALSE)` or by passing `spConform=FALSE` in the call of RFsimulate. Then the object returned by RFsimulate depends on the arguments `n` and `grid` in the following way:

if `vdim > 1` the `vdim`-variate vector makes the first dimension

if `grid=TRUE` an array of the dimension of the random field makes the next dimensions. Here, the dimensions are ordered in the sequence `x`, `y`, `z`, `T` (if given).

Else if no time component is given, then the values are passed as a single vector. Else if the time component is given the next 2 dimensions give the space and the time, respectively.

if `n > 1` the repetitions make the last dimension

Note: Conversion between the **sp** format and the conventional format can be done using the method `RFspDataFrame2conventional` and the function `conventional2RFspDataFrame`.

`InitRFsimulate` returns 0 if no error has occurred and a positive value if failed.

Note

Advanced options are

- `spConform` (suppressed return of S4 objects)
- `practicalrange` (forces range of covariances to be one)
- `exactness` (chooses the simulation method by precision)
- `seed` (sets `.Random.seed` locally or globally)

See `RFoptions` for further options.

Author(s)

Martin Schlather, `<schlather@math.uni-mannheim.de>` `http://ms.math.uni-mannheim.de/de/publications/software`

References

General

- Lantuejoul, Ch. (2002) *Geostatistical simulation.* **New York:** Springer.
- Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

Original work:

- Circulant embedding:
  Chan, G. and Wood, A.T.A. (1997) An algorithm for simulating stationary Gaussian random fields. *J. R. Stat. Soc., Ser. C* **46**, 171-181.
  Dietrich, C.R. and Newsam, G.N. (1993) A fast and exact method for multidimensional Gaussian stochastic simulations. *Water Resour. Res.* **29**, 2861-2869.
  Dietrich, C.R. and Newsam, G.N. (1996) A fast and exact method for multidimensional Gaussian stochastic simulations: Extensions to realizations conditioned on direct and indirect measurement *Water Resour. Res.* **32**, 1643-1652.
  Wood, A.T.A. and Chan, G. (1994) Simulation of stationary Gaussian processes in $[0,1]^d$ *J. Comput. Graph. Stat.* **3**, 409-432.
  The code used in *RandomFields* is based on Dietrich and Newsam (1996).

- Intrinsic embedding and Cutoff embedding:

  Stein, M.L. (2002) Fast and exact simulation of fractional Brownian surfaces. *J. Comput. Graph. Statist.* **11**, 587–599.

  Gneiting, T., Sevcikova, H., Percival, D.B., Schlather, M. and Jiang, Y. (2005) Fast and Exact Simulation of Large Gaussian Lattice Systems in R^2: Exploring the Limits *J. Comput. Graph. Statist.* Submitted.

- Markov Gaussian Random Field:

  Rue, H. (2001) Fast sampling of Gaussian Markov random fields. *J. R. Statist. Soc., Ser. B*, **63** (2), 325-338.

  Rue, H., Held, L. (2005) *Gaussian Markov Random Fields: Theory and Applications.* Monographs on Statistics and Applied Probability, no **104**, Chapman \& Hall.

- Turning bands method (TBM), turning layers:

  Dietrich, C.R. (1995) A simple and efficient space domain implementation of the turning bands method. *Water Resour. Res.* **31**, 147-156.

  Mantoglou, A. and Wilson, J.L. (1982) The turning bands method for simulation of random fields using line generation by a spectral method. *Water. Resour. Res.* **18**, 1379-1394.

  Matheron, G. (1973) The intrinsic random functions and their applications. *Adv. Appl. Probab.* **5**, 439-468.

  Schlather, M. (2004) Turning layers: A space-time extension of turning bands. *Submitted*

- Random coins:

  Matheron, G. (1967) *Elements pour une Theorie des Milieux Poreux*. Paris: Masson.

See Also

RFoptions, RMmodel, RFgui, methods for simulating Gaussian random fields, RFfit, RFempiricalvariogram, RFsimulate.more.examples, RFsimulate.sophisticated.examples, RPgauss,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RFsp-class *Class* RFsp

Description

"RFsp" is a virtual class which contains the four classes RFspatialGridDataFrame (data on a full grid and $space - time - dimension \geq 2$), RFspatialPointsDataFrame (data not on a grid and $space - time - dimension \geq 2$), RFgridDataFrame (data on a full grid and $space - time - dimension = 1$), RFpointsDataFrame (data not on a grid spaced and $space - time - dimension = 1$)

The first two class subclasses are summarized in "RFspatialDataFrame" whilst the latter two are summarized in "RFdataFrame".

Objects from the Class

are never to be generated; only derived classes can be meaningful

Methods

summary signature(obj = "RFsp"): returns a summary of the object; uses or imitates summary method of class Spatial from the **sp**-package

dimensions signature(obj = "RFsp"): retrieves the number of spatial or spatio-temporal dimensions spanned

isGridded signature(obj = "RFsp"): logical, tells whether the data is on a regular spatial grid

[signature(obj = "RFsp"): selects columns of the data-slot, while all other slots are kept unmodified

[<- signature(obj = "RFsp"): replaces columns of the data-slot, while all other slots are kept unmodified

variance signature(object = "RFsp"): returns the kriging variance if available

Warning

this class is not useful in itself, but the above mentioned classes in this package derive from it

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>, Martin Schlather

See Also

RFspatialGridDataFrame, RFspatialPointsDataFrame, RFgridDataFrame, RFpointsDataFrame, sp2RF

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
## to do
```

RFsp2conventional *coerce RFsp-class to conventional format*

Description

coerce RFsp-class to conventional format

Usage

```
RFspDataFrame2conventional(obj)

RFspDataFrame2dataArray(obj)
```

Arguments

obj	object of class RFsp.

Value

RFspDataFrame2conventional returns a list; RFspDataFrame2dataArray returns an array containing the data-slot of obj;

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>

See Also

RFspatialGridDataFrame, RFspatialPointsDataFrame

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

n <- 3
x <- GridTopology(cellcentre.offset=c(0, 0),
 cellsize=c(1, 0.2),
 cells.dim=c(10, 30))
f <- RFsimulate(model=RMexp(), x=x, n=n)
str(f)
str(RFspDataFrame2conventional(f))
```

RFspatialGridDataFrame-class
Class "RFspatialGridDataFrame"

Description

Class for spatial attributes that have spatial or spatio-temporal locations (at least of dimension 2) on a (full) regular grid. Direct extension of class SpatialGridDataFrame from the **sp**-package. See sp2RF for an explic transformation.

Creating Objects

Objects can be created by using the functions RFspatialGridDataFrame or conventional2RFspDataFrame or by calls of the form as(x, "RFspatialGridDataFrame"), where x is of class RFspatialPointsDataFrame.

Slots

.RFparams: list of 2; .RFparams$n is the number of repetitions of the random field contained in the data slot, .RFparams$vdim gives the dimension of the values of the random field, equals 1 in most cases

data: object of class data.frame, containing attribute data

grid: object of class GridTopology; grid parameters

bbox: matrix specifying the bounding box

proj4string: object of class CRS; projection

Extends

Class "SpatialGridDataFrame", directly. Class "SpatialGrid", by class "SpatialGridDataFrame". Class "Spatial", by class "SpatialGrid".

Methods

contour codesignature(obj = "RFspatialGridDataFrame"): generates contour plots

plot signature(obj = "RFspatialGridDataFrame"): generates nice image plots of the random field; if $space-time-dim2$, a two-dimensional subspace can be selected using the argument MARGIN; to get different slices in a third direction, the argument MARGIN.slices can be used; for more details see plot-method or type method?plot("RFspatialGridDataFrame")

persp codesignature(obj = "RFspatialGridDataFrame"): generates persp plots

show signature(x = "RFspatialGridDataFrame"): uses the show-method for class SpatialGridDataFrame.

print signature(x = "RFspatialGridDataFrame"): identical to show-method

RFspDataFrame2conventional signature(obj = "RFspatialGridDataFrame"): conversion to a list of non-**sp**-package based objects; the data-slot is converted to an array of dimension $[1*(vdim>1)+space-time-dimension+1*(n>1)]$; the grid-slot is converted to a 3-row matrix; the grid definition of a possible time-dimension becomes a seperate list element

RFspDataFrame2dataArray `signature(obj = "RFspatialGridDataFrame")`: conversion of the data-slot to an array of dimension $[space - time - dimension + 2]$, where the space-time-dimensions run fastest, and $vdim$ and n are the last two dimensions

coordinates `signature(x = "RFspatialGridDataFrame")`: calculates the coordinates from grid definition

[`signature(x = "RFspatialGridDataFrame")`: selects columns of data-slot; returns an object of class RFspatialGridDataFrame.

[<- `signature(x = "RFspatialGridDataFrame")`: replaces columns of data-slot; returns an object of class RFspatialGridDataFrame.

as `signature(x = "RFspatialGridDataFrame")`: converts into other formats, only implemented for target class RFspatialPointsDataFrame

cbind `signature(...)`: if arguments have identical topology, combine their attribute values

as.matrix `signature(x = "RFspatialGridDataFrame")`: converts data-slot to matrix

as.array `signature(x = "RFspatialGridDataFrame")`: converts data-slot to array

as.vector `signature(x = "RFspatialGridDataFrame")`: converts data-slot to vector

as.data.frame `signature(x = "RFspatialGridDataFrame")`: converts data-slot and coordinates to a data.frame

Details

Note that in the data-slot, each colums is ordered according to the ordering of `coordinates(grid)`, the first dimension runs fastest and for all BUT the second dimension, coordinate values are in ascending order. In the second dimension, coordinate values run from high to low. Hence, when converting to conventional formats using RFspDataFrame2conventional or RFspDataFrame2dataArray, the data array is re-ordered such that all dimensions are in ascending order. `as.matrix` does not perform re-ordering.

Methods `summary`, `dimensions` and `isGridded` are defined for the "parent"-class RFsp.

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>, Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RFspatialPointsDataFrame-class, which is for point locations that are not on a grid, RFgridDataFrame-class which is for one-dimensional locations, RFsp, sp2RF

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again


n <- 3
```

```
x <- GridTopology(cellcentre.offset=c(0, 0),
 cellsize=c(1, 0.2),
 cells.dim=c(10, 30))
f <- RFsimulate(model=RMexp(), x=x, n=n)

str(f)
str(RFspDataFrame2conventional(f))
str(RFspDataFrame2dataArray(f))
coordinates(f)[1:25,]
str(f[2]) ## selects second column of data-slot
all.equal(f, cbind(f,f)[1:3]) ## TRUE
str(as(f, "RFspatialPointsDataFrame"))

plot(f, nmax=2)

steps <- if (interactive()) c(10, 1, 10, 10) else c(2, 1, 2, 2)
x2 <- rbind(c(0, 0, 0, 0),
 c(1, 0.2, 2, 5),
 steps)
scale <- if (interactive()) 10 else 1
f2 <- RFsimulate(model=RMwhittle(nu=1.2, scale=scale), x=x2, n=n,
                 grid = TRUE)
plot(f2, MARGIN=c(3,4), MARGIN.slices=1, n.slices=6, nmax=2)

f.sp <- RFsimulate(model=RMexp(), x=x, n=n, seed=0)
f.old <- RFsimulate(model=RMexp(), x=x, n=n, spConform=FALSE, seed=0)
all.equal(RFspDataFrame2conventional(f.sp)$data, f.old) ## TRUE
```

`RFspatialPointsDataFrame-class`
Class "RFspatialPointsDataFrame"

Description

Class for spatial attributes that have spatial or spatio-temporal locations (at least of dimension 2) that are not on a grid. Direct extension of class `SpatialPointsDataFrame` from the **sp**-package. See `sp2RF` for an explicite transformation.

Creating Objects

Objects can be created by using the functions `RFspatialPointsDataFrame` or `conventional2RFspDataFrame` or by calls of the form `as(x, "RFspatialPointsDataFrame")`, where `x` is of class `RFspatialPointsDataFrame`.

Slots

`.RFparams`**:** list of 2; `.RFparams$n` is the number of repetitions of the random field contained in the `data` slot, `.RFparams$vdim` gives the dimension of the values of the random field, equals 1 in most cases

`data`: object of class `data.frame`, containing attribute data

`coords.nrs`: See `SpatialPointsDataFrame`

`coords`: matrix of coordinates (each row is a point); in case of SpatialPointsDataFrame an object of class `SpatialPoints` is also allowed see SpatialPoints

`bbox`: matrix specifying the bounding box

`proj4string`: object of class `CRS`; `projection`

Extends

Class `SpatialPointsDataFrame`, directly. Class `SpatialPoints`, by class `SpatialPointsDataFrame`. Class `Spatial`, by class `SpatialPoints`.

Methods

plot `signature(obj = "RFspatialPointsDataFrame")`: generates nice plots of the random field; if $space-time-dim2$, a two-dimensional subspace can be selected using the argument `MARGIN`; to get different slices in a third direction, the argument `MARGIN.slices` can be used; for more details see `plot-method` or type `method?plot("RFspatialPointsDataFrame")`

show `signature(x = "RFspatialPointsDataFrame")`: uses the show-method for class `SpatialPointsDataFrame`.

print `signature(x = "RFspatialPointsDataFrame")`: identical to show-method

RFspDataFrame2conventional `signature(obj = "RFspatialPointsDataFrame")`: conversion to a list of non-**sp**-package based objects; the data-slot is converted to an array of dimension $[1 * (vdim > 1) + space - time - dimension + 1 * (n > 1)]$

coordinates `signature(x = "RFspatialPointsDataFrame")`: returns the coordinates

[`signature(x = "RFspatialPointsDataFrame")`: selects columns of data-slot; returns an object of class `RFspatialPointsDataFrame`.

[<- `signature(x = "RFspatialPointsDataFrame")`: replaces columns of data-slot; returns an object of class `RFspatialPointsDataFrame`.

as `signature(x = "RFspatialPointsDataFrame")`: converts into other formats, only implemented for target class `RFspatialGridDataFrame`

cbind `signature(...)`: if arguments have identical topology, combine their attribute values

as.data.frame `signature(x = "RFspatialGridDataFrame")`: converts data-slot and coordinates to a data.frame

Details

Note that in the data-slot, each colums is ordered according to the ordering of `coordinates(grid)`, the first dimension runs fastest and for all BUT the second dimension, coordinate values are in ascending order. In the second dimension, coordinate values run from high to low. Hence, when converting to conventional formats using `RFspDataFrame2conventional` or `RFspDataFrame2dataArray`, the data array is re-ordered such that all dimensions are in ascending order. `as.matrix` does not perform re-ordering.

Methods `summary` and `dimensions` are defined for the "parent"-class `RFsp`.

Author(s)

Alexander Malinowski <alexander.malinowski@math.uni-goettingen.de>

See Also

RFspatialGridDataFrame-class, which is for point locations that are on a grid, RFpointsDataFrame-class which is for one-dimensional locations, RFsp, sp2RF

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

x <- cbind(runif(50), runif(50))
f <- RFsimulate(model=RMexp(), x=x, n=3)

str(f)
str(RFspDataFrame2conventional(f))
coordinates(f)[1:25,]
str(f[2]) ## selects second column of data-slot
all.equal(f, cbind(f,f)[1:3]) ## TRUE
try(as(f, "RFspatialGridDataFrame")) # yields error

plot(f, nmax=2)

f2 <- RFsimulate(model=RMwhittle(nu=1.2, scale=10), x=cbind(x,x), n=4)
plot(f2, MARGIN=c(3,4), nmax=2)

f.sp <- RFsimulate(model=RMexp(), x=x, n=3, seed=0)
f.old <- RFsimulate(model=RMexp(), x=x, n=3, spConform=FALSE, seed=0)
all.equal(RFspDataFrame2conventional(f.sp)$data, f.old) ## TRUE
```

RMangle *Anisotropy matrix given by angle*

Description

RMangle delivers an anisotropy matrix for the argument Aniso in RMmodel in two dimensions. RMangle requires one two strechting values, passed by ratio or diag, and an angle.

In two dimensions and with angle equal to a and diag equal to $(d1, d2)$ the anisotropy matrix A is

A = diag(d1, d2) %*% matrix(ncol=2, c(cos(a), sin(a), -sin(a), cos(a)))

In three dimensions and with angle equal to a, second angle L and diag equal to $(d1, d2, d3)$ the anisotropy matrix A is

A = diag(d1, d2, d3) %*% matrix(ncol=3, c(cos(a) * cos(L), sin(a) * cos(L), sin(L), -sin(a), cos
)) i.e. Ax turns a vector x first in $x - z$ plane, then in the $x - y$ plane.

Usage

```
RMangle(angle, lat.angle, ratio, diag)
```

Arguments

angle	anglea
lat.angle	second angle; in 3 dimensions only
ratio	equivalent to diag=c(1, 1/ratio); in 2 dimensions only
diag	the diagonal components of the matrix

Value

RMangle returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMtrafo, RMmodel

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMexp(Aniso=RMangle(angle=pi/4, ratio=3))
plot(model, dim=2)

x <- seq(0, 2, if (interactive()) 0.05 else 1)
z <- RFsimulate(x, x, model=model)
plot(z)


model <- RMexp(Aniso=RMangle(angle=pi/4, lat.angle=pi/8, diag=c(1,2,3)))
x <- seq(0, 2, if (interactive()) 0.2 else 1)
z <- RFsimulate(x, x, x, model=model)
plot(z, MARGIN.slices=3)
```

RMaskey *Askey model*

Description

Askey's model

$$C(x) = (1 - x)^{\alpha} 1_{[0,1]}(x)$$

Usage

```
RMaskey(alpha, var, scale, Aniso, proj)
RMtent(var, scale, Aniso, proj)
```

Arguments

`alpha`	a numerical value in the interval [0,1]
`var,scale,Aniso,proj`	optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

Details

This covariance function is valid for dimension d if $\alpha \geq (d + 1)/2$. For $\alpha = 1$ we get the well-known triangle (or tent) model, which is valid on the real line, only.

Value

`RMaskey` returns an object of class `RMmodel`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Covariance function

- Askey, R. (1973) *Radial characteristic functions.* Technical report, Research Center, University of Wisconsin-Madison.
- Golubov, B. I. (1981) On Abel-Poisson type and Riesz means, *Anal. Math.* 7, 161-184.

Applications as covariance function

- Gneiting, T. (1999) Correlation functions for atmospheric data analysis. *Quart. J. Roy. Meteor. Soc.*, 125:2449-2464.
- Gneiting, T. (2002) Compactly supported correlation functions. *J. Multivar. Anal.*, 83:493-508.

- Wendland, H. (1994) *Ein Beitrag zur Interpolation mit radialen Basisfunktionen.* Diplomarbeit, Goettingen.
- Wendland, H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv. Comput. Math., 4:389-396, 1995.

Tail correlation function (for $\alpha \geq [d/2] + 1$)

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

RMmodel, RMbigneiting, RMgengneiting, RMgneiting, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMtent()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMave *Space-time moving average model*

Description

RMave is a univariate stationary covariance model which depends on a normal scale mixture covariance model phi.

The corresponding covariance function only depends on the difference $(h, u) \in \mathbf{R}^d$ between two points in the d-dimensional space and is given by

$$C(h,u) = |E + 2Ahh^tA|^{-1/2}\phi(\sqrt{(}\|h\|^2/2 + (z^th + u)^2(1 - 2h^tA(E + 2Ahh^tA)^{-1}Ah)))$$

where E is the identity matrix. The spatial dimension is $d - 1$ and h is real-valued.

Usage

```
RMave(phi, A, z, spacetime, var, scale, Aniso, proj)
```

Arguments

phi	a covariance model which is a normal mixture, that means an RMmodel whose monotone property equals 'normal mixture', see RFgetModelNames(monotone="normal mixture")
A	a symmetric $d-1 \times d-1$-matrix if the corresponding random field is in the d-dimensional space
z	a $d-1$ dimensional vector if the corresponding random field is on d-dimensional space
spacetime	logical. If FALSE then the model is interpreted as if $h = 0$, i.e. the spatial dimension is d. Default is TRUE
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

See Schlather, M. (2010), Example 13 with l=1)

Value

RMave returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M. (2010) Some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

See Also

RFfit. RFsimulate, RMmodel, RMstp

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## Example of an evaluation of the ave2-covariance function
## in three different ways
## ------------------------------------------------------------
## some parameters A and z
A <- matrix(c(2,1,1,2),ncol=2)
z <- c(1,2)
## h for evalutation
h <- c(1,2)
## some abbreviations
E <- matrix(c(1,0,0,1),ncol=2)
```

```
B <- A %*% h %*% t(h) %*% A
phi <- function(t){return(RFcov(RMwhittle(1), t))}
## ----------------------------------------------------------
## the following should yield the same value 3 times
## (also for other choices of A,z and h)
z1 <- RFcov( model=RMave(RMwhittle(1),A=A,z=z) , x=t(c(h,0)) )
z2 <- RFcov( model=RMave(RMwhittle(1),A=A,z=z,spacetime=FALSE) , x=t(h) )
z3 <- ( (det(E+2*B))^(-1/2) ) *
 phi( sqrt( sum(h*h)/2 + (t(z) %*% h)^2 *
 ( 1-2*t(h) %*% A %*% solve(E+2*B) %*% A %*% h) ) )
##
stopifnot(abs(z1-z2)<1e-12, abs(z2-z3)<1e-12)
```

RMball *RMball*

Description

RMball referes to the indicator function of a ball with radius 1.

Usage

```
RMball(var, scale, Aniso, proj)
```

Arguments

var, scale, Aniso, proj
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RMpolygon RMspheric, RFsimulate, RMmodel.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RMbcw *Model bridging stationary and intrinsically stationary processes*

Description

RMbcw is a variogram model that bridges between some intrinsically stationary isotropic processes and some stationary ones. It reunifies the RMgenfbm, RMdewijsian and RMgencauchy.

The corresponding centered semi-variogram only depends on the distance $r \geq 0$ between two points and is given by

$$\gamma(r) = \frac{(r^{\alpha}+1)^{\beta/alpha}-1}{2^{\beta/alpha}-1}$$

where $\alpha \in (0,2]$ and $beta \leq 2$.

Usage

```
RMbcw(alpha, beta, var, scale, Aniso, proj)
```

Arguments

alpha	a numerical value; should be in the interval (0,2].
beta	a numerical value; should be in the interval (-infty,2].
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above variogram remains unmodified.

Details

For $betaa > 0$, $beta < 0$, $beta = 0$ we have the generalised fractal Brownian motion RMgenfbm, the generalised Cauchy model RMgencauchy, and the de Wisjian model RMdewijsian, respectively.

Hence its two arguments alpha and beta allow for modelling the smoothness and a wide range of tail behaviour, respectively.

Value

RMbcw returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schilling, R.L., Song, R., Vondracek, Z. (2010) Berstein functions. Theory and Applications. De Gruyter
- Schlather, M (2014) A parameteric variogram model bridging between stationary and intrinsically stationary processes. *Submitted* , –

See Also

RMgenfbm, RMgencauchy, RMdewijsian, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMbcw(alpha=1, beta=0.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMbernoulli *Covariance Model for binary field based on a Gaussian field*

Description

RMbernoulli gives the centered **correlation** function of a binary field, obtained by thresholding a Gaussian field.

Usage

```
RMbernoulli(phi, threshold, correlation, centred, var, scale, Aniso, proj)
```

Arguments

phi	covariance function of class RMmodel.
threshold	real valued threshold, see RPbernoulli. Currently only threshold=0.0 is possible. Default: 0.
correlation	logical. If FALSE the corresponding covariance function is returned Default: TRUE. .
centred	logical. If FALSE the uncentred covariance is returned. Default: TRUE.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model yields the covariance function of the field that is returned by RPbernoulli

Value

RMbernoulli returns an object of class RMmodel.

Note

Previous to version 3.0.33 the covariance function was returned, not the correlation function

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Ballani, Schlather

See Also

RPbernoulli RMmodel, RFsimulate,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

threshold <- 0
x <- seq(0, 5, if (interactive()) 0.02 else 1)
GaussModel <- RMgneiting()

n <- if (interactive()) 1000 else 2
z <- RFsimulate(RPbernoulli(GaussModel, threshold=threshold), x=x, n=n)
plot(z)

model <- RMbernoulli(RMgauss(), threshold=threshold, correlation=FALSE)
plot(model, xlim=c(0,5))
z <- as.matrix(z)
estim.cov <- apply(z, 1, function(x) cov(x, z[1,]))
points(x, estim.cov, col="red")
```

RMbessel *Bessel Family Covariance Model*

Description

RMbessel is a stationary isotropic covariance model belonging to the Bessel family. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = 2^{\nu}\Gamma(\nu+1)r^{-\nu}J_{\nu}(r)$$

where $\nu \geq \frac{d-2}{2}$, Γ denotes the gamma function and J_{ν} is a Bessel function of first kind.

Usage

```
RMbessel(nu, var, scale, Aniso, proj)
```

Arguments

nu	a numerical value; should be equal to or greater than $\frac{d-2}{2}$ to provide a valid covariance function for a random field of dimension d.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This covariance models a hole effect (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 92, cf. Gelfand et al. (2010), p. 26).

An important case is $\nu = -0.5$ which gives the covariance function

$$C(r) = \cos(r)$$

and which is only valid for $d = 1$. This equals RMdampedcos for $\lambda = 0$, there.

A second important case is $\nu = 0.5$ with covariance function

$$C(r) = \sin(r)/r$$

and which is valid for $d \leq 3$. This coincides with RMwave.

Note that all valid continuous stationary isotropic covariance functions for d-dimensional random fields can be written as scale mixtures of a Bessel type covariance function with $\nu = \frac{d-2}{2}$ (cf. Gelfand et al., 2010, pp. 21–22).

Value

RMbessel returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.
- Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.
- http://homepage.tudelft.nl/11r49/documents/wi4006/bessel.pdf

See Also

RMdampedcos, RMwave, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMbessel(nu=1, scale=0.1)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMbigneiting *Gneiting-Wendland Covariance Models*

Description

RMbigneiting is a bivariate stationary isotropic covariance model family whose elements are specified by seven parameters.

Let

$$\delta_{ij} = \mu + \gamma_{ij} + 1.$$

Then,

$$C_n(h) = c_{ij}(C_{n,\delta}(h/s_{ij}))_{i,j=1,2}$$

and $C_{n,\delta}$ is the generalised Gneiting model with parameters n and δ, see RMgengneiting, i.e.,

$$C_{\kappa=0,\delta}(r) = (1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \delta + 2\kappa + 1/2;$$

$$C_{\kappa=1,\delta}(r) = (1+\beta r)\,(1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \delta + 2\kappa + 1/2;$$

$$C_{\kappa=2,\delta}(r) = \left(1 + \beta r + \frac{\beta^2 - 1}{3} r^2\right)(1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \delta + 2\kappa + 1/2;$$

$$C_{\kappa=3,\delta}(r) = \left(1 + \beta r + \frac{(2\beta^2 - 3)}{5} r^2 + \frac{(\beta^2 - 4)\beta}{15} r^3\right)(1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \delta + 2\kappa + 1/2.$$

Usage

```
RMbigneiting(kappa, mu, s, sred12, gamma, cdiag, rhored, c, var, scale, Aniso, proj)
```

Arguments

kappa	argument that chooses between the four different covariance models and may take values $0, \ldots, 3$. The model is k times differentiable.
mu	mu has to be greater than or equal to $\frac{d}{2}$ where d is the (arbitrary) dimension of the randomfield.
s	vector of two elements giving the scale of the models on the diagonal, i.e., the vector (s_{11}, s_{22}).
sred12	value in $[-1, 1]$. The scale on the offdiagonals is given by $s_{12} = s_{21} =$ sred12 * $\min\{s_{11}, s_{22}\}$.

gamma	a vector of length 3 of numerical values; each entry is positive. The vector gamma equals $(\gamma_{11}, \gamma_{21}, \gamma_{22})$. Note that $\gamma_{12} = \gamma_{21}$.
cdiag	a vector of length 2 of numerical values; each entry positive; the vector (c_{11}, c_{22})
c	a vector of length 3 of numerical values; the vector (c_{11}, c_{21}, c_{22}). Note that $c_{12} = c_{21}$. Either rhored and cdiag or c must be given.
rhored	value in $[-1, 1]$. See also the Details for the corresponding value of $c_{12} = c_{21}$.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

A sufficient condition for the constant c_{ij} is

$$c_{12} = \rho_{\mathrm{red}} \cdot m \cdot \left(c_{11} c_{22} \prod_{i,j=1,2} \left(\frac{\Gamma(\gamma_{ij} + \mu + 2\kappa + 5/2)}{b_{ij}^{\nu_{ij}+2\kappa+1} \Gamma(1+\gamma_{ij}) \Gamma(\mu + 2\kappa + 3/2)} \right)^{(-1)^{i+j}} \right)^{1/2}$$

where $\rho_{\mathrm{red}} \in [-1, 1]$.

The constant m in the formula above is obtained as follows:

$$m = \min\{1, m_{-1}, m_{+1}\}$$

Let

$$a = 2\gamma_{12} - \gamma_{11} - \gamma_{22}$$
$$b = -2\gamma_{12}(s_{11} + s_{22}) + \gamma_{11}(s_{12} + s_{22}) + \gamma_{22}(s_{12} + s_{11})$$
$$e = 2\gamma_{12} s_{11} s_{22} - \gamma_{11} s_{12} s_{22} - \gamma_{22} s_{12} s_{11}$$
$$d = b^2 - 4ae$$
$$t_j = \frac{-b + j\sqrt{d}}{2a}$$

If $d \geq 0$ and $t_j \notin (0, s_{12})$ then $m_j = \infty$ else

$$m_j = \frac{(1 - t_j/s_{11})^{\gamma_{11}} (1 - t_j/s_{22})^{\gamma_{22}}}{(1 - t_j/s_{12})^{2\gamma_{11}}} m_j = (1 - t_j/s_{11})^{\gamma_{11}} (1 - t_j/s_{22})^{\gamma_{22}} / (1 - t_j/s_{12})^{2\gamma_{11}}$$

In the function RMbigneiting, either c is passed, then the above condition is checked, or rhored is passed then c_{12} is calculated by the above formula.

Value

RMgengneiting returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Bevilacqua, M., Daley, D.J., Porcu, E., Schlather, M. (2012) Classes of compactly supported correlation functions for multivariate random fields. Arxiv.
- Gneiting, T. (1999) Correlation functions for atmospherical data analysis. *Q. J. Roy. Meteor. Soc* Part A **125**, 2449-2464.
- Wendland, H. (2005) *Scattered Data Approximation.* Cambridge Monogr. Appl. Comput. Math.

See Also

RMaskey, RMbiwm, RMgengneiting, RMgneiting, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMbigneiting(kappa=2, mu=0.5, gamma=c(0, 3, 6), rhored=1)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMbiwm *Full Bivariate Whittle Matern Model*

Description

RMbiwm is a bivariate stationary isotropic covariance model whose corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given for $i, j \in \{1, 2\}$ by

$$C_{ij}(r) = c_{ij} W_{\nu_{ij}}(r/s_{ij}).$$

Here W_ν is the covariance of the RMwhittle model. For constraints on the constants see details.

Usage

```
RMbiwm(nudiag, nured12, nu, s, cdiag, rhored, c, notinvnu, var,
 scale, Aniso, proj)
```

Arguments

nudiag	a vector of length 2 of numerical values; each entry positive; the vector (ν_{11}, ν_{22})
nured12	a numerical value in the interval $[1, \infty)$; ν_{21} is calculated as $0.5(\nu_{11}+\nu_{22})*\nu_{red}$.
nu	alternative to nudiag and nured12: a vector of length 3 of numerical values; each entry positive; the vector $(\nu_{11}, \nu_{21}, \nu_{22})$. Either nured and nudiag, or nu must be given.

s	a vector of length 3 of numerical values; each entry positive; the vector (s_{11}, s_{21}, s_{22})
cdiag	a vector of length 2 of numerical values; each entry positive; the vector (c_{11}, c_{22})
rhored	a numerical value; in the interval $[-1, 1]$. See also the Details for the corresponding value of $c_{12} = c_{21}$.
c	a vector of length 3 of numerical values; the vector (c_{11}, c_{21}, c_{22}). Either rhored and cdiag or c must be given.
notinvnu	logical or NULL. If not given (default) then the formula of the (RMwhittle) model applies. If logical then the formula for the RMmatern model applies. See there for details.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Constraints on the constants: For the diagonal elements we have

$$\nu_{ii}, s_{ii}, c_{ii} > 0.$$

For the offdiagonal elements we have

$$s_{12} = s_{21} > 0,$$

$$\nu_{12} = \nu_{21} = 0.5(\nu_{11} + \nu 22) * \nu_{red}$$

for some constant $\nu_{red} \in [1, \infty)$ and

$$c_{12} = c_{21} = \rho_{red}\sqrt{fmc_{11}c_{22}}$$

for some constant ρ_{red} in $[-1, 1]$.

The constants f and m in the last equation are given as follows:

$$f = (\Gamma(\nu_{11}+d/2)\Gamma(\nu_{22}+d/2))/(\Gamma(\nu_{11})\Gamma(\nu_{22})) * (\Gamma(\nu_{12})/\Gamma(\nu_{12}+d/2))^2 * (s_{12}^{2*\nu_{12}}/(s_{11}^{\nu_{11}}s_{22}^{\nu_{22}}))^2$$

where Γ is the Gamma function and d is the dimension of the space. The constant m is the infimum of the function g on $[0, \infty)$ where

$$g(t) = (1/s_{12}^2 + t^2)^{2\nu_{12}+d}(1/s_{11}^2 + t^2)^{-\nu_{11}-d/2}(1/s_{22}^2 + t^2)^{-\nu_{22}-d/2}$$

(cf. Gneiting, T., Kleiber, W., Schlather, M. (2010), Full Bivariate Matern Model (Section 2.2))

Value

RMbiwm returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T., Kleiber, W., Schlather, M. (2010) Matern covariance functions for multivariate random fields *JASA*

See Also

RMparswm, RMwhittle, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- y <- seq(-10, 10, if (interactive()) 0.2 else 5)
model <- RMbiwm(nudiag=c(0.3, 2), nured=1, rhored=1, cdiag=c(1, 1.5),
                s=c(1, 1, 2))
plot(model)
plot(RFsimulate(model, x, y))
```

RMbr2bg *Transformation from Brown-Resnick to Bernoulli*

Description

This function can be used to model a max-stable process based on the a binary field, with the same extremal correlation function as a Brown-Resnick process

$$C_{bg}(h) = \cos(\pi(2\Phi(\sqrt{\gamma(h)/2}) - 1))$$

Here, Φ is the standard normal distribution function, and γ is a **semi-**variogram with sill

$$4(erf^{-1}(1/2))^2 = 2 * \Phi^{-1}(3/4)^2 = 1.819746/2 = 0.9098728$$

Usage

```
RMbr2bg(phi, var, scale, Aniso, proj)
```

Arguments

phi	covariance function of class RMmodel.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

RMbr2bg
binary random field RPbernoulli simulated with RMbr2bg(RMmodel()) has a uncentered covariance function that equals

1. the tail correlation function of the max-stable process constructed with this binary random field
2. the tail correlation function of Brown-Resnick process with variogram RMmodel.

Note that the reference paper is based on the notion of the (genuine) variogram, whereas the package **RandomFields** is based on the notion of semi-variogram. So formulae differ by factor 2.

Value

object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

maxstableAdvanced, RMbr2eg, RMmodel, RMm2r, RPbernoulli, RPbrownresnick, RPschlather,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMexp(var=1.62 / 2)
step <- if (interactive()) 0.05 else 2
y <- seq(0, 10, step)
z <- RFsimulate(RPschlather(RMbr2eg(model)), y, y)
plot(z)
```

RMbr2eg *Transformation from Brown-Resnick to Gauss*

Description

This function can be used to model a max-stable process based on the a binary field, with the same extremal correlation function as a Brown-Resnick process

$$C_{eg}(h) = 1 - 2(1 - 2\Phi(\sqrt{\gamma(h)/2}))^2$$

Here, Φ is the standard normal distribution function, and γ is a **semi-**variogram with sill

$$4(erf^{-1}(1/\sqrt{2}))^2 = 2 * [\Phi^{-1}([1 + 1/\sqrt{2}]/2)]^2 = 4.425098/2 = 2.212549$$

Usage

```
RMbr2eg(phi, var, scale, Aniso, proj)
```

Arguments

`phi`	covariance function of class `RMmodel`.
`var,scale,Aniso,proj`	optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

Details

`RMbr2eg`
The extremal Gaussian model `RPschlather` simulated with `RMbr2eg(RMmodel())` has tail correlation function that equals the tail correlation function of Brown-Resnick process with variogram `RMmodel`.

Note that the reference paper is based on the notion of the (genuine) variogram, whereas the package **RandomFields** is based on the notion of semi-variogram. So formulae differ by factor 2.

Value

object of class `RMmodel`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

maxstableAdvanced, RMbr2bg, RMmodel, RMm2r, RPbernoulli, RPbrownresnick, RPschlather,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMexp(var=1.62 / 2)
binary.model <- RPbernoulli(RMbr2bg(model))
step <- if (interactive()) 0.05 else 2
y <- seq(0, 10, step)
z <- RFsimulate(RPschlather(binary.model), y, y)
plot(z)
```

RMbrownresnick *Tail correlation function of the Brown-Resnick process*

Description

RMbrownresnick defines the tail correlation function of the Brown-Resnick process.

$$C(h) = 2 - 2\Phi(\sqrt{\gamma(h)}/2)$$

where ϕ is the standard normal distribution function and γ is the **semi-**variogram.

Usage

```
RMbrownresnick(phi, var, scale, Aniso, proj);
```

Arguments

phi	variogram of class RMmodel.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

For a given RMmodel the function RMbrownresnick(RMmodel()) 'returns' the tail correlation function of a Brown-Resnick process with variogram RMmodel.

Value

object of class RMmodel

Note

In the paper Kabluchko et al (2009) the variogram instead of the semi-variogram is considered, so the formulae differ slightly.

In Version 3.0.33 a typo has been corrected.

Here, a definition is used that is consistend with the rest of the package.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Kabluchko, Z., Schlather, M. & de Haan, L (2009) Stationary max-stable random fields associated to negative definite functions *Ann. Probab.* **37**, 2042-2065.
- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

RFsimulate, RMm2r, RMm3b, RMmps, RMmodel.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

#plot covariance model of type RMbrownresnick
RMmodel <- RMfbm(alpha=1.5, scale=0.2)
plot(RMbrownresnick(RMmodel))

#simulate and plot corresponding Gaussian random field
x <- if (interactive()) seq(-5, 5, 0.05) else seq(-5, 5, 5)
z <- RFsimulate(RMbrownresnick(RMmodel), x=x, y=x)
plot(z)
```

RMcauchy *Cauchy Family Covariance Model*

Description

RMcauchy is a stationary isotropic covariance model belonging to the Cauchy family. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (1 + r^2)^(- \gamma)$$

where $\gamma > 0$. See also RMgencauchy.

Usage

```
RMcauchy(gamma, var, scale, Aniso, proj)
```

Arguments

gamma	a numerical value; should be positive to provide a valid covariance function for a random field of any dimension.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The paramater γ determines the asymptotic power law. The smaller γ, the longer the long-range dependence. The covariance function is very regular near the origin, because its Taylor expansion only contains even terms and reaches its sill slowly.

Each covariance function of the Cauchy family is a normal scale mixture.

The generalized Cauchy Family (see RMgencauchy) includes this family for the choice $\alpha = 2$ and $\beta = 2\gamma$. The generalized Hyperbolic Family (see RMhyperbolic) includes this family for the choice $\xi = 0$ and $\gamma = -\nu/2$; in this case scale=δ.

Value

RMcauchy returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. and Schlather, M. (2004) Stochastic models which separate fractal dimension and Hurst effect. *SIAM review* **46**, 269–282.
- Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.

See Also

RMcauchytbm, RMgencauchy, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMcauchy(gamma=1)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model, xlim=c(-3, 3))
plot(RFsimulate(model, x=x, n=4))
```

RMcauchytbm *Modifications of the Cauchy Family Covariance Model*

Description

RMcauchytbm() is a shortcut of RMtbm(RMgencauchy()) and is given here for downwards compatibility

Usage

```
RMcauchytbm(alpha, beta, gamma, var, scale, Aniso, proj)
```

Arguments

alpha,beta	see RMgencauchy.
gamma	is the same as fulldim in RMtbm.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMcauchytbm returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. and Schlather, M. (2004) Stochastic models which separate fractal dimension and Hurst effect. *SIAM review* **46**, 269–282.

See Also

RMcauchy, RMgencauchy, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMcauchytbm(alpha=1, beta=1, gamma=3)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMcircular *Circular Covariance Model*

Description

RMcircular is a stationary isotropic covariance model which is only valid for dimensions $d \leq 2$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = 1 - 2/\pi(r\sqrt{(1-r^2)} + arcsin(r))1_{[0,1]}.$$

Usage

```
RMcircular(var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is only valid for dimensions $d \leq 2$. It is a covariance function with compact support (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 82).

Value

RMcircular returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMcircular()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMconstant *Constant Covariance Matrix*

Description

RMconstant is a user-defined covariance according to the given covariance matrix.

Usage

```
RMconstant(M, vdim, element, var, scale, Aniso, proj)
```

Arguments

M	a numerical matrix defining the user-defined covariance for a random field; The matrix should be positive definite, symmetric and its dimension should be equal to the length of observation or simulation vector.
vdim	an integer value; defining the response dimension (for details see RFgetModelNames).
element	integer. If a list of several matrices is given then element chooses the one that should currently be used. Default: 0
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Note, this is not a covariance model for arbitrary locations, the covariance is fixed and user-defined by a given matrix corresponding to the locations. In particular, it is used in RFfit to define neighbour or network structure in the data.

Value

RMconstant returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Ober, U., Ayroles, J.F., Stone, E.A., Richards, S., Zhu, D., Gibbs, R.A., Stricker, C., Gianola, D., Schlather, M., Mackay, T.F.C., Simianer, H. (2012): *Using Whole Genome Sequence Data to Predict Quantitative Trait Phenotypes in Drosophila melanogaster*. PLoS Genet 8(5): e1002685.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
## to do
```

RMcoord *Coordinates in the mixed model representation*

Description

This function is used to define, in the (rare) case, coordinates that differ from the original coordinates to define a covariance matrix for a random effect model

```
~ RMmodel() + Z @ RMcoord(coord=X, RMmodel2())
```

Usage

```
RMcoord(C0, coord, dist)
```

Arguments

C0	covariance function of class RMmodel.
coord, dist	either coordinates or a the lower matrix of a distance matrix can be passed

Value

RMcoord returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RFfit.

Examples

```
 ## For examples see the help page of 'RFformula' ##

## todo
```

RMcoxisham *Cox Isham Covariance Model*

Description

RMcoxisham is a stationary covariance model which depends on a univariate stationary isotropic covariance model C_0, which is a normal scale mixture.

The corresponding covariance function only depends on the difference $(h,t) \in \mathbf{R}^{d+1} = \mathbf{R}^d \times \mathbf{R}$ between two points in $d+1$-dimensional space and is given by

$$C(h,t) = |E + t^\beta D|^{-1/2} C_0([(h - t\mu)^T (E + t^\beta D)^{-1} (h - t\mu)]^{1/2})$$

Here $\mu \in \mathbf{R}^d$ is a vector in d-dimensional space; E is the $d \times d$-identity matrix and D is a $d \times d$-correlation matrix with $|D| > 0$. The parameter β is in $(0, 2]$. Currently, the implementation is done only for $d = 2$.

Usage

```
RMcoxisham(phi,mu,D,beta,var, scale, Aniso, proj)
```

Arguments

phi	a univariate stationary isotropic covariance model for random fields on d-dimensional space, which is moreover a normal scale mixture, that means an RMmodel whose monotone property equals 'normal mixture', see RFgetModelNames(monotone="normal mixture") and whose maxdim is at least 2.
mu	a vector in d-dimensional space
D	a $d \times d$-correlation matrix with $\|D\| > 0$
beta	numeric in the interval $(0, 2]$; default value is 2
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model stems from a rainfall model (cf. Cox, D.R., Isham, V.S. (1988)) and equals the following expectation

$$C(h,t) = \boldsymbol{E}_V C_0(h - Vt)$$

where the random wind speed vector V follows a d-variate normal distribution with expectation mu and covariance matrix $D/2$. (cf. See Schlather, M. (2010), Example 9).

Value

RMcoxisham returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Cox, D.R., Isham, V.S. (1988) A simple spatial-temporal model of rainfall. *Proc. R. Soc. Lond. A*, **415**, 317-328.
- Schlather, M. (2010) On some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMcoxisham(RMgauss(), mu=1, D=1)
x <- seq(0, 10, if (interactive()) 0.3 else 1)
plot(model, dim=2)
plot(RFsimulate(model, x=x, y=x))
```

RMcubic *Cubic Covariance Model*

Description

RMcubic is a stationary isotropic covariance model which is only valid for dimensions $d \leq 3$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (1 - 7r^(2) + 8.75r^3 - 3.5r^5 + 0.75r^7)1_{[0,1]}(r).$$

Usage

```
RMcubic(var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is only valid for dimensions $d \leq 3$. It is a 2 times differentiable covariance function with compact support (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 84).

Value

RMcubic returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMcubic()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMcurlfree *Curlfree Covariance Model*

Description

RMcurlfree is a multivariate covariance model which depends on a univariate stationary covariance model where the covariance function phi(h) is twice differentiable.

The corresponding matrix-valued covariance function C of the model only depends on the difference h between two points and it is given by

$$C(h) = (-\nabla_h (\nabla_h)^T) C_0(h)$$

Usage

```
RMcurlfree(phi, var, scale, Aniso, proj)
```

Arguments

phi	a univariate stationary covariance model (2 or 3 dimensional).
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model returns the potential field in the first component, the corresponding curlfree field and field of sources and sinks in the last component.

See also the models RMdivfree and RMvector.

Value

RMcurlfree returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Scheuerer, M. and Schlather, M. (2012) Covariance Models for Divergence-Free and Curl-Free Random Vector Fields. *Stochastic Models* **28:3**.

See Also

RMdivfree, RMvector, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMcurlfree(RMgauss(), scale=4)
plot(model, dim=2)

x.seq <- y.seq <- seq(-10, 10, if (interactive()) 0.2 else 5)
simulated <- RFsimulate(model=model, x=x.seq, y=y.seq)
plot(simulated, select.variables=list(1, c(1, 2:3), 4))
```

RMcutoff *Gneiting's modification towards finite range*

Description

RMcutoff is a functional on univariate stationary isotropic covariance functions ϕ.

The corresponding function C (which is not necessarily a covariance function, see details) only depends on the distance r between two points in d-dimensional space and is given by

$$C(r) = \phi(r), 0 \leq r \leq d$$

$$C(r) = b_0((dR)^a - r^a)^{2a}, d \leq r \leq dR$$

$$C(r) = 0, dR \leq r$$

The parameters R and b_0 are chosen internally such that C is a smooth function.

Usage

```
RMcutoff(phi, diameter, a, var, scale, Aniso, proj)
```

Arguments

phi	a univariate stationary isotropic covariance model. See, for instance, `RFgetModelNames(type="positive definite", domain="single variable", isotropy="iso`
diameter	a numerical value; should be greater than 0; the diameter of the domain on which the simulation is done
a	a numerical value; should be greater than 0; has been shown to be optimal for $a = 1/2$ or $a = 1$.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The algorithm that checks the given parameters knows only about some few necessary conditions. Hence it is not ensured that the cutoff-model is a valid covariance function for any choice of ϕ and the parameters.

For certain models ϕ, e.g. RMstable, RMwhittle and RMgencauchy, some sufficient conditions are known (cf. Gneiting et al. (2006)).

Value

RMcutoff returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T., Sevecikova, H, Percival, D.B., Schlather M., Jiang Y. (2006) Fast and Exact Simulation of Large Gaussian Lattice Systems in R^2: Exploring the Limits. *J. Comput. Graph. Stat.* **15**, 483–501.
- Stein, M.L. (2002) Fast and exact simulation of fractional Brownian surfaces. *J. Comput. Graph. Statist.* **11**, 587–599

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
 ## For examples see the help page of 'RFsimulateAdvanced' ##
model <- RMexp()
plot(model, model.cutoff=RMcutoff(model, diameter=1), xlim=c(0, 4))
```

RMdagum *Dagum Covariance Model Family*

Description

RMdagum is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = 1 - (1 + r^{-\beta})^{\frac{-\gamma}{\beta}}.$$

The parameters β and γ can be varied in the intervals $(0, 1]$ and $(0, 1)$, respectively.

Usage

```
RMdagum(beta, gamma, var, scale, Aniso, proj)
```

Arguments

beta	numeric in $(0, 1]$
gamma	numeric in $(0, 1)$
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Like the generalized Cauchy model the Dagum family can be used to model fractal dimension and Hurst effect. For a comparison of these see Berg, C. and Mateau, J. and Porcu, E. (2008). This paper also establishes valid parameter choices for the Dagum family, but be careful because therein the model is parameterized differently.

Value

RMdagum returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Berg, C. and Mateau, J. and Porcu, E. (2008) The dagum family of isotropic correlation functions. *Bernoulli* **14**(4), 1134–1149.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMdagum(beta=0.5, gamma=0.5, scale=0.2)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMdampedcos *Exponentially Damped Cosine*

Description

RMdampedcos is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = exp(-\lambda r)\cos(r).$$

Usage

```
RMdampedcos(lambda, var, scale, Aniso, proj)
```

Arguments

`lambda`	numeric. The range depends on the dimension of the random field (see details)
`var, scale, Aniso, proj`	optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

Details

The model is valid for any dimension d. However, depending on the dimension of the random field the following bound for the arguments λ has to be respected:

$$\lambda \geq 1/\tan(\pi/(2d)).$$

This covariance models a hole effect (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 92).

For $\lambda = 0$ we obtain the covariance function

$$C(r) = \cos(r)$$

which is only valid for $d = 1$ and corresponds to `RMbessel` for $\nu = -0.5$, there.

Value

`RMdampedcos` returns an object of class `RMmodel`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.
- Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.

See Also

`RMbessel`, `RMmodel`, `RFsimulate`, `RFfit`.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMdampedcos(lambda=0.3, scale=0.1)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMdelay *Bivariate Delay Effect*

Description

RMdelay is a $(m+1)$-variate stationary covariance model. which depends on a univariate stationary covariance model C_0.

The corresponding covariance function only depends on the difference $h \in \mathbf{R}^d$ between two points in d-dimensional space and is given by

$$C(h) = (C_0(h - s_i + s_j))_{i,j=0,\ldots,m}$$

where $s \in \mathbf{R}^{d \times m}$ and $s_0 = 0$

Usage

```
RMdelay(phi,s,var, scale, Aniso, proj)
```

Arguments

phi	a univariate stationary covariance model, that means an RMmodel whose vdim equals 1.
s	a $d \times m$-dimensional shift matrix, where d is the dimension of the space, giving the components $s = (s_1, \ldots, s_m)$ where the s_i are vectors.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Here, a multivariate random field is obtained from single univariate random field, by shifting it by fixed value.

Value

RMdelay returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M., Strokorb, K. (2013). *Submitted to J. Statist. Software*
- Wackernagel, H. (2003) *Multivariate Geostatistics.* Berlin: Springer, 3nd edition.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

x <- y <- seq(-10,10,0.2)
model <- RMdelay(RMstable(alpha=1.9, scale=2), s=c(4,4))
plot(model, dim=2, xlim=c(-6, 6), ylim=c(-6,6))

simu <- RFsimulate(model, x, y)
plot(simu, zlim="joint")
```

RMdewijsian *Modified DeWijsian Variogram Model*

Description

The modified RMdewijsian model is an intrinsically stationary isotropic variogram model. The corresponding centered semi-variogram only depends on the distance $r \geq 0$ between two points and is given by

$$\gamma(r) = \log(r^\alpha + 1)$$

where $\alpha \in (0, 2]$.

Usage

```
RMdewijsian(alpha, var, scale, Aniso, proj)
```

Arguments

alpha	a numerical value; in the interval (0,2].
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above variogram remains unmodified.

Details

Originally, the logarithmic model $\gamma(r) = \log(r)$ was named after de Wijs and reflects a principle of similarity (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 90). But note that $\gamma(r) = \log(r)$ is not a valid variogram ($\gamma(0)$ does not vanish) and can only be understood as a characteristic of a generalised random field.

The modified RMdewijsian model $\gamma(r) = \log(r^\alpha + 1)$ is a valid variogram model (cf. Wackernagel, H. (2003), p. 336).

Value

RMdewijsian returns an object of class RMmodel

Note

Note that the (non-modified) de Wijsian model equals $\gamma(r) = \log(r)$.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Wackernagel, H. (2003) *Multivariate Geostatistics.* Berlin: Springer, 3nd edition.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMdewijsian(alpha=1)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMdivfree *Divfree Covariance Model*

Description

RMdivfree is a multivariate covariance model which depends on a univariate stationary covariance model where the covariance function phi(h) is twice differentiable.

The corresponding matrix-valued covariance function C of the model only depends on the difference h between two points and it is given by

$$C(h) = (-\Delta E + \nabla\nabla^T)C_0(h)$$

Usage

```
RMdivfree(phi, var, scale, Aniso, proj)
```

Arguments

phi	a univariate stationary covariance model (in 2 or 3 dimensions).
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model returns the potential field in the first component, the corresponding divfree field and the field of curl strength in the last component.

See also the models RMcurlfree and RMvector.

Value

RMdivfree returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Scheuerer, M. and Schlather, M. (2012) Covariance Models for Divergence-Free and Curl-Free Random Vector Fields. *Stochastic Models* **28:3**.

See Also

RMcurlfree, RMvector, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMdivfree(RMgauss(), scale=4)
plot(model, dim=2)

x.seq <- y.seq <- seq(-10, 10, if (interactive()) 0.2 else 5)
simulated <- RFsimulate(model=model, x=x.seq, y=y.seq)

plot(simulated)
plot(simulated, select.variables=1)
plot(simulated, select.variables=2:3)
plot(simulated, select.variables=list(2:3))
plot(simulated, select.variables=list(1, 2:3, 4))
plot(simulated, select.variables=list(1, c(1, 2:3), 4))
```

RMeaxxa *Special models for rotation like fields*

Description

RMeaxxa and RMetaxxa define the auxiliary functions

$$f(h) = h^\top AA^\top h + diag(E)$$

and

$$f(h) = h^\top ARRA^\top h + diag(E)$$

, respectively.

Usage

```
RMeaxxa(E, A)
RMetaxxa(E, A, alpha)
```

Arguments

E	m-variate vector of positive values
A	$m \times k$ matrix
alpha	angle for the rotation matrix R

Details

RMeaxxa is defined in space and returns an m-variate model.

RMetaxxa is a space-time model with two spatial dimensions. The matrix R is a rotation matrix with angle βt where t is the time component.

Value

RMeaxxa and RMetaxxa return an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M. (2010) On some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

See Also

RMmodel, S10

Examples

```
# see S10
```

RMepscauchy *Generalized Cauchy Family Covariance Model*

Description

RMepscauchy is a stationary isotropic covariance model belonging to the generalized Cauchy family. **In contrast to most other models it is not a correlation function.** The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (\varepsilon + r^{\alpha})^{(} - \beta/\alpha)$$

where $\epsilon > 0$, $\alpha \in (0, 2]$ and $\beta > 0$. See also RMcauchy.

Usage

```
RMepscauchy(alpha, beta, eps, var, scale, Aniso, proj)
```

Arguments

alpha	a numerical value; should be in the interval (0,2] to provide a valid covariance function for a random field of any dimension.
beta	a numerical value; should be positive to provide a valid covariance function for a random field of any dimension.
eps	a positive value
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model has a smoothness parameter α and a paramater β which determines the asymptotic power law. More precisely, this model admits simulating random fields where fractal dimension *D* of the Gaussian sample and Hurst coefficient *H* can be chosen independently (compare also RMlgd.): Here, we have

$$D = d + 1 - \alpha/2, \alpha \in (0, 2]$$

and

$$H = 1 - \beta/2, \beta > 0.$$

I. e. the smaller β, the longer the long-range dependence.

The covariance function is very regular near the origin, because its Taylor expansion only contains even terms and reaches its sill slowly.

Each covariance function of the Cauchy family is a normal scale mixture.

Note that the Cauchy Family (see RMcauchy) is included in this family for the choice $\alpha = 2$ and $\beta = 2\gamma$.

Value

RMepscauchy returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. and Schlather, M. (2004) Stochastic models which separate fractal dimension and Hurst effect. *SIAM review* **46**, 269–282.

See Also

RMcauchy, RMcauchytbm, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMepscauchy(alpha=1.5, beta=1.5, scale=0.3, eps=0.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMexp *Exponential Covariance Model*

Description

RMexp is a stationary isotropic covariance model whose corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = e^{-r}.$$

Usage

```
RMexp(var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model is a special case of the Whittle covariance model (see RMwhittle) if $\nu = \frac{1}{2}$ and of the symmetric stable family (see RMstable) if $\nu = 1$. Moreover, it is the continuous-time analog of the first order autoregressive time series covariance structure.

The exponential covariance function is a normal scale mixture.

Value

RMexp returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Covariance model

- Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.

Tail correlation function

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

RMwhittle, RMstable, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMexp()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMexponential *Exponential operator*

Description

RMexponential yields a covariance model from a given variogram or covariance model. The covariance C is given as

$$C(h) = \frac{\exp(\phi(h)) - \sum_{k=0}^{n} \phi^k(h)/k!}{\exp(\phi(0)) - \sum_{k=0}^{n} \phi^k(0)/k!}$$

if ϕ is a covariance model, and as

$$C(h) = \exp(-\phi(h))$$

if ϕ is a variogram model.

Usage

```
RMexponential(phi, n, standardised, var, scale, Aniso, proj)
```

Arguments

phi	a valid RMmodel; either a variogram model or a covariance model
n	integer, see formula above. Default is -1.; if the multivariate dimension of the submodel is greater than 1 then only the default value is valid.
standardised	logical. If TRUE then the above formula holds. If FALSE then only the nominator of the above formula is returned. Default value is TRUE.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

If γ is a variogram, then $\exp(-\gamma)$ is a valid covariance.

Value

RMexponential returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

See, for instance,

- Berg, C., Christensen, J. P. R., Ressel, P. (1984) Harmonic Analysis on Semigroups. *Theory of Positive Definite and Related Functions.* Springer, New York.
- Sasvari, Z. (2013) *Multivariate Characteristic and Correlation Functions*. de Gruyter, Berlin.
- Schlather, M. (2010) *Some covariance models based on normal scale mixtures*, *Bernoulli* **16**, 780-797.
- Schlather, M. (2012) Construction of covariance functions and unconditional simulation of random fields. In Porcu, E., Montero, J. M., Schlather, M. *Advances and Challenges in Space-time Modelling of Natural Events*, Springer, New York.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMexponential(RMfbm(alpha=1))  ## identical to RMexp()
plot(RMexp(), model=model, type=c("p", "l"), pch=20)
```

RMfbm *Variogram Model of Fractal Brownian Motion*

Description

RMfbm is an intrinsically stationary isotropic variogram model. The corresponding centered semi-variogram only depends on the distance $r \geq 0$ between two points and is given by

$$\gamma(r) = r^{\alpha}$$

where $\alpha \in (0, 2]$.
By now, the model is implemented for dimensions up to 3.
For a generalized model see also RMgenfbm.

Usage

```
RMfbm(alpha, var, scale, Aniso, proj)
```

Arguments

alpha	numeric in $(0, 2]$; refers to the fractal dimension of the process
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above variogram remains unmodified.

Details

The variogram is unbounded and belongs to a non-stationary process with stationary increments. For $\alpha = 1$ and `scale=2` we get a variogram corresponding to a standard Brownian Motion.

For $\alpha \in (0, 2)$ the quantity $H = \frac{\alpha}{2}$ is called Hurst index and determines the fractal dimension D of the corresponding Gaussian sample paths

$$D = d + 1 - H$$

where d is the dimension of the random field (see Chiles and Delfiner, 1999, p. 89).

Value

`RMfbm` returns an object of class `RMmodel`.

Author(s)

Martin Schlather, `<schlather@math.uni-mannheim.de>`

References

- Chiles, J.-P. and P. Delfiner (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York, Chichester: John Wiley & Sons.
- Stein, M.L. (2002) Fast and exact simulation of fractional Brownian surfaces. *J. Comput. Graph. Statist.* **11**, 587–599

See Also

`RMgenfbm`, `RMmodel`, `RFsimulate`, `RFfit`.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMfbm(alpha=1)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

`RMfixed` *Fixed Effect Model*

Description

Expressions of the form `X@RMfixed(beta)` can be used within a formula of the type

$$response \tilde{} fixedeffects + randomeffects + errorterm$$

that specifies the Linear Mixed Model.

Important Remark: `RMfixed` is NOT a function although the parantheses notation is used to specify the vector of coefficients.

The matrix X is the design matrix and β is a vector of coefficients

Note that a fixed effect of the form X is interpreted as `X@RMfixed(beta=NA)` by default (and β is estimated provided that the formula is used in `RFfit`). Note that the `1` in an expression `1@RMfixed(beta)` is interpreted as the identity matrix.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

`RMmodel`, `RFformula`, `RFsimulate`,

Examples

```
## For examples see the help page of 'RFformula' ##
```

`RMflatpower` *Variogram Model Similar to Fractal Brownian Motion*

Description

`RMflatpower` is an intrinsically stationary isotropic variogram model. The corresponding centered semi-variogram only depends on the distance $r \geq 0$ between two points and is given by

$$\gamma(r) = r^2/(1+r^2)^\alpha$$

where $\alpha \in (0, 1]$.

For related models see `RMgenfbm`.

Usage

```
RMflatpower(alpha, var, scale, Aniso, proj)
```

Arguments

alpha	numeric in $(0, 1]$; refers to the fractal dimension of the process
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above variogram remains unmodified.

Details

The model is always smooth at the origin.

The parameter α only gives the tail behaviour and satisfies $\alpha \in (0, 1]$.

The variogram is unbounded and belongs to a non-stationary process with stationary increments.

Value

RMflatpower returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Oesting, M., Schlather, M., and Friederichs, P. (2014) Conditional Modelling of Extreme Wind Gusts by Bivariate Brown-Resnick Processes *arxiv* **1312.4584**.

See Also

RMgenfbm, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMflatpower(alpha=0.5)
x <- seq(0, 10, if (interactive()) 0.1 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMfractdiff *Fractionally Differenced Process Model*

Description

RMfractdiff is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given for integers $r \in \mathbf{N}$ by

$$C(r) = (-1)^r \frac{\Gamma(1-a/2)^2}{\Gamma(1-a/2+r)\Gamma(1-a/2-r)} r \in \mathbf{N}$$

and otherwise linearly interpolated. Here $a \in [-1, 1)$, Γ denotes the gamma function. It can only be used for one-dimensional random fields.

Usage

```
RMfractdiff(a, var, scale, Aniso, proj)
```

Arguments

a	$-1 \leq a < 1$
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is only valid for dimension $d = 1$. It stems from time series modelling where the grid locations are multiples of the scale parameter.

Value

RMfractdiff returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMfractdiff(0.5, scale=0.2)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMfractgauss *Fractal Gaussian Model Family*

Description

RMfractgauss is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = 0.5((r+1)^\alpha - 2r^\alpha + |r-1|^\alpha)$$

with $0 < \alpha \leq 2$. It can only be used for one-dimensional random fields.

Usage

```
RMfractgauss(alpha,var, scale, Aniso, proj)
```

Arguments

alpha	$0 < \alpha \leq 2$
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is only valid for dimension $d = 1$. It is the covariance function for the fractional Gaussian noise with self-affinity index (Hurst parameter) $H = \alpha/2$ with $0 < \alpha \leq 2$.

Value

RMfractgauss returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. and Schlather, M. (2004) Stochastic models which separate fractal dimension and Hurst effect. *SIAM review* **46**, 269–282.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMfractgauss(alpha=0.5, scale=0.2)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMgauss *Gaussian Covariance Model*

Description

RMgauss is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = e^{-r^2}$$

Usage

```
RMgauss(var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model is called Gaussian because of the functional similarity of the spectral density of a process with that covariance function to the Gaussian probability density function.

The Gaussian model has an infinitely differentiable covariance function. This smoothness is artificial. Furthermore, this often leads to singular matrices and therefore numerically instable procedures (cf. Stein, M. L. (1999), p. 29).

The Gaussian model is included in the symmetric stable class (see RMstable) for the choice $\alpha = 2$.

Value

RMgauss returns an object of class RMmodel

Note

The use of RMgauss is questionable from both a theoretical (analytical paths) and a practical point of view (e.g., speed of algorithms). Instead, RMgneiting should be used.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.

Stein, M. L. (1999) *Interpolation of Spatial Data.* New York: Springer-Verlag

See Also

RMstable and RMmatern for generalisations;
RMmodel, RFsimulate, RFfit.

Do not mix up with RPgauss or RRgauss.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMgauss(scale=0.4)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
lines(RMgauss(), col="red")
plot(RFsimulate(model, x=x))
```

RMgencauchy *Generalized Cauchy Family Covariance Model*

Description

RMgencauchy is a stationary isotropic covariance model belonging to the generalized Cauchy family. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (1 + r^{\alpha})^{(} - \beta/\alpha)$$

where $\alpha \in (0, 2]$ and $\beta > 0$. See also RMcauchy.

Usage

```
RMgencauchy(alpha, beta, var, scale, Aniso, proj)
```

Arguments

alpha	a numerical value; should be in the interval (0,2] to provide a valid covariance function for a random field of any dimension.
beta	a numerical value; should be positive to provide a valid covariance function for a random field of any dimension.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model has a smoothness parameter α and a paramater β which determines the asymptotic power law. More precisely, this model admits simulating random fields where fractal dimension D of the Gaussian sample and Hurst coefficient H can be chosen independently (compare also with RMlgd.): Here, we have

$$D = d + 1 - \alpha/2, \alpha \in (0, 2]$$

and

$$H = 1 - \beta/2, \beta > 0.$$

I. e. the smaller β, the longer the long-range dependence.

The covariance function is very regular near the origin, because its Taylor expansion only contains even terms and reaches its sill slowly.

Each covariance function of the Cauchy family is a normal scale mixture.

Note that the Cauchy Family (see RMcauchy) is included in this family for the choice $\alpha = 2$ and $\beta = 2\gamma$.

Value

RMgencauchy returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Covariance function

- Gneiting, T. and Schlather, M. (2004) Stochastic models which separate fractal dimension and Hurst effect. *SIAM review* **46**, 269–282.

Tail correlation function (for $\alpha \in (0, 1]$)

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

RMcauchy, RMcauchytbm, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMgencauchy(alpha=1.5, beta=1.5, scale=0.3)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMgenfbm *Generalized Fractal Brownian Motion Variogram Model*

Description

RMgenfbm is an intrinsically stationary isotropic variogram model. The corresponding centered semi-variogram only depends on the distance $r \geq 0$ between two points and is given by

$$\gamma(r) = (r^\alpha + 1)^{\beta/\alpha} - 1$$

where $\alpha \in (0, 2]$ and $\beta \in (0, 2]$.
See also RMfbm.

Usage

```
RMgenfbm(alpha, beta, var, scale, Aniso, proj)
```

Arguments

alpha	a numerical value; should be in the interval (0,2].
beta	a numerical value; should be in the interval (0,2].
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above variogram remains unmodified.

Details

Here the variogram of RMfbm is modified by the transformation $(\gamma + 1)^{\delta/-1}$ on variograms γ for $delta \in (0, 1]$. This original modification allows for further generalization, cf. RMbcw.

Value

RMgenfbm returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. (2002) Nonseparable, stationary covariance functions for space-time data, *JASA* **97**, 590-600.
- Schlather, M. (2010) On some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

See Also

RMbcw RMfbm, RMmodel, RMflatpower, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMgenfbm(alpha=1, beta=0.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMgengneiting *Gneiting-Wendland Covariance Models*

Description

RMgengneiting is a stationary isotropic covariance model family whose elements are specified by the two parameters κ and μ with n a non-negative integer and $\mu \geq \frac{d}{2}$ with d denoting the dimension of the random field (the models can be used for any dimension). A corresponding covariance function only depends on the distance $r \geq 0$ between two points. For the case $\kappa = 0$ the Gneiting-Wendland model equals the Askey model RMaskey,

$$C(r) = (1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \mu + 1/2 = \mu + 2\kappa + 1/2.$$

For $\kappa = 1$ the Gneiting model is given by

$$C(r) = (1+\beta r)\,(1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \mu + 2\kappa + 1/2.$$

If $\kappa = 2$

$$C(r) = \left(1 + \beta r + \frac{\beta^2 - 1}{3} r^2\right)(1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \mu + 2\kappa + 1/2.$$

In the case $\kappa = 3$

$$C(r) = \left(1 + \beta r + \frac{(2\beta^2 - 3)}{5} r^2 + \frac{(\beta^2 - 4)\beta}{15} r^3\right)(1-r)^{\beta} 1_{[0,1]}(r), \qquad \beta = \mu + 2\kappa + 1/2.$$

A special case of this model is RMgneiting. ℓ

Usage

```
RMgengneiting(kappa, mu, var, scale, Aniso, proj)
```

Arguments

kappa	$0, \ldots, 3$; it chooses between the three different covariance models above
mu	mu has to be greater than or equal to $\frac{d}{2}$ where d is the dimension of the random field.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This isotropic family of covariance functions is valid for any dimension of the random field.

A special case of this family is RMgneiting (with $s = 1$ there) for the choice $\kappa = 3, \mu = 3/2$.

Value

RMgengneiting returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. (1999) Correlation functions for atmospherical data analysis. *Q. J. Roy. Meteor. Soc* Part A **125**, 2449-2464.
- Wendland, H. (2005) *Scattered Data Approximation.* Cambridge Monogr. Appl. Comput. Math.

See Also

RMaskey, RMbigneiting, RMgneiting, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMgengneiting(kappa=1, mu=1.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))


## same models:
model2 <- RMgengneiting(kappa=3, mu=1.5, scale= 1 / 0.301187465825)
plot(RMgneiting(), model2=model2, type=c("p", "l"), pch=20)
```

RMgneiting *Gneiting Covariance Model*

Description

RMgneiting is a stationary isotropic covariance model which is only valid up to dimension 3, or 5 (see the argument orig). The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (1 + 8sr + 25s^2r^2 + 32s^3r^3)(1 - sr)^8$$

if $0 \geq r \geq \frac{1}{s}$ and

$$C(r) = 0$$

otherwise. Here, $s = 0.301187465825$. For a generalized model see also RMgengneiting.

Usage

```
RMgneiting(orig, var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.
orig	logical. if TRUE the above model is used. Otherwise the RMgengneiting model C(s r) with kappa=3 as above, but but with mu = 2.683509 and s=0.2745640815 is used. The latter has the advantage of being closer to the Gaussian model and it is valid up to dimension 5. Default: TRUE

Details

This isotropic covariance function is valid only for dimensions less than or equal to 3. It is 6 times differentiable and has compact support.

This model is an alternative to RMgauss as its graph is hardly distinguishable from the graph of the Gaussian model, but possesses neither the mathematical nor the numerical disadvantages of the Gaussian model.

It is a special case of RMgengneiting for the choice $kappa = 3, \mu = 1.5$.

Note that, in the original work by Gneiting (1999), a numerical value slightly deviating from the optimal one was used for $mu = 1.5$: $s = \frac{10\sqrt{(2)}}{47}$.

Value

RMgneiting returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

For the original version

- Gneiting, T. (1999) Correlation functions for atmospherical data analysis. *Q. J. Roy. Meteor. Soc* Part A **125**, 2449-2464.

For the version (orig=FALSE)

- this package **RandomFields**

See Also

RMbigneiting, RMgengneiting, RMgauss, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

plot(RMgneiting(), model2=RMgneiting(orig=FALSE), model3=RMgauss(),
     xlim=c(-3,3), maxchar=100)
plot(RMgneiting(), model2=RMgneiting(orig=FALSE), model3=RMgauss(),
     xlim=c(1.5,2.5), maxchar=100)

model <- RMgneiting(orig=FALSE, scale=0.4)
x <- seq(0,10, if (interactive()) 0.1 else 5)
z <- RFsimulate(model, x=x, y=x, z=x, T=c(1,1,5))
plot(z, MARGIN.slices=4, MARGIN.movie=3)
```

RMgneitingdiff *Gneiting Covariance Model Used as Tapering Function*

Description

RMgneitingdiff is a stationary isotropic covariance model which is only valid up to dimension 3. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(h) = C_0(h/t)W_\nu(h/s)$$

where C_0 is Gneiting's model RMgneiting and W_ν is the Whittle modelRMwhittle.

Usage

```
RMgneitingdiff(nu, taper.scale, scale, var, Aniso, proj)
```

Arguments

nu	see RMwhittle
taper.scale	is the paramter t in the above formula
scale	is the paramter s in the above formula
var,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model allows to a certain degree the smooth modelling of the differentiability of a covariance function with compact support.

Value

RMgneitingdiff returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. (1999) Correlation functions for atmospherical data analysis. *Q. J. Roy. Meteor. Soc* Part A **125**, 2449-2464.

See Also

RMbigneiting, RMgneiting, RMgengneiting, RMgauss, RMmodel, RMwhittle, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMgneitingdiff(nu=2, taper.scale=1, scale=0.2)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMhyperbolic *Generalized Hyperbolic Covariance Model*

Description

RMhyperbolic is a stationary isotropic covariance model called "generalized hyperbolic". The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = \frac{(\delta^2 + r^2)^{\nu/2} K_\nu(\xi(\delta^2 + r^2)^{1/2})}{\delta^\nu K_\nu(\xi\delta)}$$

where K_ν denotes the modifies Bessel function of second kind.

Usage

```
RMhyperbolic(nu, lambda, delta, var, scale, Aniso, proj)
```

Arguments

`nu, lambda, delta`
: numerical values; should either satisfy
$\delta \geq 0$, $\lambda > 0$ and $\nu > 0$, or
$\delta > 0$, $\lambda > 0$ and $\nu = 0$, or
$\delta > 0$, $\lambda \geq 0$ and $\nu < 0$.

`var,scale,Aniso,proj`
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This class is over-parametrized, i.e. it can be reparametrized by replacing the three parameters λ, δ and scale by two other parameters. This means that the representation is not unique.

Each generalized hyperbolic covariance function is a normal scale mixture.

The model contains some other classes as special cases; for $\lambda = 0$ we get Cauchy covariance function (see RMcauchy) with $\gamma = -\frac{\nu}{2}$ and scale=δ; the choice $\delta = 0$ yields a covariance model of type RMwhittle with smoothness parameter ν and scale parameter λ^{-1}.

Value

RMhyperbolic returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Shkarofsky, I.P. (1968) Generalized turbulence space-correlation and wave-number spectrum-function pairs. *Can. J. Phys.* **46**, 2133-2153.
- Barndorff-Nielsen, O. (1978) Hyperbolic distributions and distributions on hyperbolae. *Scand. J. Statist.* **5**, 151-157.
- Gneiting, T. (1997). Normal scale mixtures and dual probability densities. *J. Stat. Comput. Simul.* **59**, 375-384.

See Also

RMcauchy, RMwhittle, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMhyperbolic(nu=1, lambda=2, delta=0.2)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMiaco *Iaco-Cesare model*

Description

The space-time covariance function is

$$C(r,t) = (1.0 + r^{\nu} + t^{\lambda})^{\delta}$$

Usage

```
RMiaco(nu, lambda, delta, var, scale, Aniso, proj)
```

Arguments

nu,lambda	number in $(0, 2]$
delta	positive number
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMiaco returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- de Cesare, L., Myers, D.E., and Posa, D. (2002) FORPRAN programs for space-time modeling. *Computers \& Geosciences* **28**, 205-212.
- de Iaco, S.. Myers, D.E., and Posa, D. (2002) Nonseparable space-time covariance models: some parameteric families. *Math. Geol.* **34**, 23-42.

See Also

RMmodel,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMiaco(nu=1, lambda=2, delta=0.5)
plot(model, dim=2)

x <- seq(0, 10, if (interactive()) 0.1 else 5)
plot(RFsimulate(model, x=x, y=x))
```

RMid *Identical Model*

Description

RMid is the identical operator for objects of class RMmodel

Usage

```
RMid(phi, vdim, var, scale, Aniso, proj)
```

Arguments

phi	covariance function of class RMmodel.
vdim	for internal purposes
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMid returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMexp()
x <- 0:10
z <- RFsimulate(model, x)

model2 <- RMid(model)
z2 <- RFsimulate(model, x)
sum(abs(as.vector(z)- as.vector(z2))) == 0 # TRUE
```

RMintern *Internal models*

Description

Internal models that may appear in feedbacks from 'RandomFields'

Details

The following and many more internal models exist

- RF__Name__ : internal representation of certain functions RF__name__
- RO# : model for transforming coordinates within the cartesian system
- RO> : model for transforming earth coordinates to cartesian coordinates
- ROmissing : for error messages only
- RMmixed : internal representation of a mixed model
- RMselect : will be obsolete in future
- RMsetparam, RMptsGivenShape, RMstandardShape, RMstatShape : for max-stable processes and Poisson processes: models that combine shape functions with corresponding point processes
- RP__name__Intern : internal representations of some processes
- RPS, RPplusp, etc. : specific processes for RMS and RMplus etc. (For those covariance models that have specific simulation processes programmed.)
- RMS : internal representation of the modifying arguments var, scale, Aniso, proj

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## in the following 'try' the model 'RMtbmIntern' appears
model <- RPtbm(RMexp())
x <- seq(0, 10, 1)
try(RFsimulate(model, x=x)) ## fails
```

RMintexp *Integral exponential operator*

Description

RMintexp is a univariate stationary covariance model depending on a univariate variogram model ϕ. The corresponding covariance function only depends on the difference h between two points and is given by

$$C(h) = (1 - exp(-\phi(h)))/\phi(h)$$

Usage

```
RMintexp(phi, var, scale, Aniso, proj)
```

Arguments

phi	a variogram RMmodel.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMintexp returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M. (2012) Construction of covariance functions and unconditional simulation of random fields. *Lecture Notes in Statistics, Proceedings*, **207**, 25–54.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMintexp(RMfbm(alpha=1.5, scale=0.2))
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMintrinsic *Intrinsic Embedding Covariance Model*

Description

RMintrinsic is a univariate stationary isotropic covariance model which depends on a univariate stationary isotropic covariance model.

The corresponding covariance function C of the model only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = a_0 + a_2 r^2 + \phi(r), 0 \leq r \leq diameter$$

$$C(r) = b_0(rawRD - r)^3/(r), diameter \leq r \leq rawR * diameter$$

$$C(r) = 0, rawR * diameter \leq r$$

Usage

```
RMintrinsic(phi, diameter, rawR, var, scale, Aniso, proj)
```

Arguments

phi	an RMmodel; has to be stationary and isotropic
diameter	a numerical value; positive; should be the diameter of the domain on which simulation is done
rawR	a numerical value; greater or equal to 1
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The parameters a_0, a_2 and b_0 are chosen internally such that C becomes a smooth function. See formulas (3.8)-(3.10) in Gneiting et alii (2006). This model corresponds to the method Intrinsic Embedding. See also RPintrinsic.

NOTE: The algorithm that checks the given parameters knows only about some few necessary conditions. Hence it is not ensured that the Stein-model is a valid covariance function for any choice of ϕ and the parameters.

For certain models ϕ, i.e. `stable`, `whittle`, `gencauchy`, and the variogram model `fractalB` some sufficient conditions are known.

Value

RMintrinsic returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T., Sevecikova, H, Percival, D.B., Schlather M., Jiang Y. (2006) Fast and Exact Simulation of Large Gaussian Lattice Systems in R^2: Exploring the Limits. *J. Comput. Graph. Stat.* **15**, 483–501.
- Stein, M.L. (2002) Fast and exact simulation of fractional Brownian surfaces. *J. Comput. Graph. Statist.* **11**, 587–599

See Also

RPintrinsic, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x.max <- 10
model <- RMintrinsic(RMfbm(alpha=1), diameter=x.max)
x <- seq(0, x.max, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMkolmogorov *Identical Model*

Description

RMkolmogorov corresponds to a vector-valued random fields with covariance function

$$\gamma_{ij}(h) = \|h\|^{2/3} \left(\frac{4}{3}\delta_{ij} - \frac{1}{3}\frac{h_i h_j}{\|h\|^2} \right)$$

Usage

```
RMkolmogorov(var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMkolmogorov returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

The above formula is from eq. (6.32) of section 6.2 in

Pope, S.B. (2011) *Turbulent Flows.* **Cambridge:** Cambridge University Press.

See Also

RMmodel, RMcurlfree, RMdivfree, RMvector,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- y <- seq(-2, 2, len=if (interactive()) 20 else 2)
model <- RMkolmogorov()
plot(model, dim=3, MARGIN=1:2, fixed.MARGIN=1)

simu <- RFsimulate(model, x, y, z=0)
plot(simu, select.variables=list(c(1,2)), col=c("red"))
```

RMlgd *Local-Global Distinguisher Family Covariance Model*

Description

RMlgd is a stationary isotropic covariance model, which is valid only for dimensions $d = 1, 2$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = 1 - \beta^{-1}(\alpha + \beta)r^{\alpha}1_{[0,1]}(r) + \alpha^{-1}(\alpha + \beta)r^{-\beta}1_{r>1}(r)$$

where $\beta > 0$ and $0 < \alpha \leq (3 - d)/2$, with d denoting the dimension of the random field.

Usage

```
RMlgd(alpha, beta, var, scale, Aniso, proj)
```

Arguments

alpha	argument whose range depends on the dimension of the random field: $0 < \alpha \leq (3 - d)/2$.
beta	beta > 0.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is only valid for dimension $d = 1, 2$.

This model admits simulating random fields where fractal dimension *D* of the Gaussian sample and Hurst coefficient *H* can be chosen independently (compare also RMgencauchy.):

Here, the random field has fractal dimension

$$D = d + 1 - \alpha/2$$

and Hurst coefficient

$$H = 1 - \beta/2$$

for $0 < \beta \leq 1$.

Value

RMlgd returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. and Schlather, M. (2004) Stochastic models which separate fractal dimension and Hurst effect. *SIAM review* **46**, 269–282.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMlgd(alpha=0.7, beta=4, scale=0.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMma *Ma operator*

Description

RMma is a univariate stationary covariance model depending on a univariate stationary covariance model. The corresponding covariance function only depends on the difference h between two points and is given by

$$C(h) = (\theta/(1-(1-\theta)\phi(h)))^{\alpha}$$

Usage

```
RMma(phi, alpha, theta, var, scale, Aniso, proj)
```

Arguments

phi	a stationary covariance RMmodel.
alpha	a numerical value; positive
theta	a numerical value; in the interval $(0, 1)$
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMma returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Ma, C. (2003) Spatio-temporal covariance functions generated by mixtures. *Math. Geol.*, **34**, 965-975.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMma(RMgauss(), alpha=4, theta=0.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMmastein *Ma-Stein operator*

Description

RMmastein is a univariate stationary covariance model depending on a variogram or covariance model on the real axis. The corresponding covariance function only depends on the difference h between two points and is given by

$$C(h,t) = \frac{\Gamma(\nu+\phi(t))\Gamma(\nu+\delta)}{\Gamma(\nu+\phi(t)+\delta)\Gamma(\nu)} W_{\nu+\phi(t)}(\|h-Vt\|)$$

if ϕ is a variogram model. It is given by

$$C(h,t) = \frac{\Gamma(\nu+\phi(0)-\phi(t))\Gamma(\nu+\delta)}{\Gamma(\nu+\phi(0)-\phi(t)+\delta)\Gamma(\nu)} W_{\nu+\phi(t)}(\|h-Vt\|)$$

if ϕ is a covariance model.

Here Γ is the Gamma function; W is the Whittle-Matern model (RMwhittle).

Usage

```
RMmastein(phi, nu, delta, var, scale, Aniso, proj)
```

Arguments

phi	an RMmodel on the real axis
nu	numerical value; positive; smoothness parameter of the Whittle-Matern model (for $t = 0$)
delta	a numerical value; δ must be greater than or equal to half the dimension of h
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

See Stein (2005) formula (12).

Instead of the velocity parameter V in the original model description, a preceeding anisotropy matrix is chosen appropriately:

$$\begin{pmatrix} A & -V \\ 0 & 1 \end{pmatrix}$$

A is a spatial transformation matrix. (I.e. (x,t) is multiplied from left on the above matrix and the first elements of the obtained vector are intepreted as new spatial components and only these components are used to form the argument in the Whittle-Matern function.) The last component in the new coordinates is the time which is passed to ϕ. (Velocity is assumed to be zero in the new coordinates.)

Note, that for numerical reasons, $\nu+\phi+d$ may not exceed the value 80.0. If exceeded the algorithm fails.

Value

RMmastein returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Ma, C. (2003) Spatio-temporal covariance functions generated by mixtures. *Math. Geol.*, **34**, 965-975.
- Stein, M.L. (2005) Space-time covariance functions. *JASA*, **100**, 310-321.

See Also

RMwhittle, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make plotthem all random again
model <- RMmastein(RMgauss(), nu=1, delta=10)
plot(RMexp(), model.mastein=model, dim=2)

x <- seq(0, 10, if (interactive()) 0.1 else 3)
plot(RFsimulate(model, x=x, y=x))
```

RMmatrix *Matrix operator*

Description

RMmatrix is a multivariate covariance model depending on a multivariate covariance model ϕ. The corresponding covariance function is given by

$$C(h) = M\phi(h)M^t$$

Usage

```
RMmatrix(phi, M, var, scale, Aniso, proj)
```

Arguments

phi	a k-variate covariance RMmodel.
M	a k times k matrix
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMmatrix returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Strokorb, K. (2013) R package **RandomFields**: Analysis and simulation of multivariate random fields and more. *Submitted*

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
# bivariate Linear Model of Coregionalisation
model <- RMmatrix(M = c(0.9, 0.43), RMwhittle(nu = 0.3)) +
         RMmatrix(M = c(0.6, 0.8), RMwhittle(nu = 2))
x <- y <- seq(-10, 10, 0.2)
simu <- RFsimulate(model, x, y)
plot(simu)
```

RMmodel *Covariance and Variogram Models in* **RandomFields** *(RM commands)*

Description

Summary of implemented covariance and variogram models

Details

To generate a covariance or variogram model for use within **RandomFields**, calls of the form

$$RM_name(..., var, scale, Aniso, proj)$$

can be used, where _name_ has to be replaced by a valid model name,

- `...` can take model specific arguments.
- `var` is the optional variance argument v,
- `scale` the optional scale argument s,
- `Aniso` an optional anisotropy matrix A or given by `RMangle`, and
- `proj` is the optional projection vector which defines a diagonal matrix of zeros and ones and `proj` gives the positions of the ones (integer values).

With ϕ denoting the original model, the transformed model is $C(h) = v * \phi(A * h/s)$.

`RM_name_` must be a function of class `RMmodelgenerator`. The return value of all functions `RM_name_` is of class `RMmodel`.

The following models are available (cf. `RFgetModelNames`).

Basic stationary and isotropic models

`RMcauchy`	Cauchy family
`RMexp`	exponential model
`RMgencauchy`	generalized Cauchy family

RMgauss	Gaussian model
RMgneiting	differentiable model with compact support
RMmatern	Whittle-Matern model
RMnugget	nugget effect model
RMspheric	spherical model
RMstable	symmetric stable family or powered exponential model
RMwhittle	Whittle-Matern model, alternative parametrization

Variogram models (stationary increments/intrinsically stationary)

RMfbm	fractal Brownian motion

Basic Operations

RMmult, *	product of covariance models
RMplus, +	sum of covariance models or variograms

Basic models for mixed effect modelling

RMconstant	constant pre-defined covariance
RMfixed	fixed or trend effects; Caution: RMfixed is not a function and can be used only in formula notation

Others

RMtrend	trend
RMangle	defines a 2x2 anisotropy matrix by rotation and stretch arguments.

See RMmodelsAdvanced for many more, advanced models.

Author(s)

Alexander Malinowski, <malinowski@math.uni-mannheim.de>

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.
- Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

- Schlather, M. (2011) Construction of covariance functions and unconditional simulation of random fields. In Porcu, E., Montero, J.M. and Schlather, M., *Space-Time Processes and Challenges Related to Environmental Problems.* New York: Springer.
- Yaglom, A.M. (1987) *Correlation Theory of Stationary and Related Random Functions I, Basic Results.* New York: Springer.
- Wackernagel, H. (2003) *Multivariate Geostatistics.* Berlin: Springer, 3nd edition.

See Also

RC, RF, RP, RR, RFcov, RFformula, RMmodelsAdvanced, RMmodelsAuxiliary

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
RFgetModelNames(type="positive definite", domain="single variable",
                isotropy="isotropic", operator=FALSE)

## an example of a simple model
model <- RMexp(var=1.6, scale=0.5) + RMnugget(var=0) #exponential + nugget
plot(model)
```

RMmodel-class *Class* RMmodel

Description

Class for **RandomField**'s representation of explicit covariance models

Creating Objects

Objects are created by calling a function of class RMmodelgenerator

Slots

`call`: language object; the function call by which the object was generated

`name`: character string; nickname of the model, name of the function by which the object was generated

`submodels`: list; contains submodels (if existent)

`par.model`: list; conatins model specific arguments

`par.general`: list of 4; contains the four standard arguments `var`, `scale`, `Aniso` and `proj` that can be given for any model; if not specified by the user, the string `"RFdefault"` is inserted

Methods

+ `signature(x = "RMmodel")`: allows to sum up covariance models; internally calls `RMplus`.

***** `signature(x = "RMmodel")`: allows to multiply covariance models; internally calls `RMmult`.

plot `signature(x = "RMmodel")`: gives a plot of the covariance function or of the variogram model, for more details see `plot-method`.

points `signature(x = "RMmodel")`: adds a covariance plot to an existing plot, for more details see `plot-method`.

lines `signature(x = "RMmodel")`: adds a covariance plot to an existing plot, for more details see `plot-method`.

str `signature(x = "RMmodel")`: as the usual `str`-method for S4 objects but where only those entries of the 'par.general'-slot are shown that contain values different from `'RFdefault'`

show `signature(x = "RMmodel")`: returns the structure of `x`

print `signature(x = "RMmodel")`: identical with show-method, additional argument is `max.level`

[`signature(x = "RMmodel")`: enables accessing the slots via the `"["`-operator, e.g. `x["par.general"]`

[<- `signature(x = "RMmodel")`: enables replacing the slots via the `"["`-operator

Author(s)

Alexander Malinowski <malinows@math.uni-goettingen.de>

See Also

`RMmodelgenerator` RMmodel

Examples

```
# see RMmodel for introductory examples


# Compare:
model <- RMexp(scale=2) + RMnugget(var=3)
str(model)  ## S4 object as default in version 3 of RandomFields

model <- summary(model)
str(model)  ## list style as in version 2 of RandomFields
            ## see also 'spConform' in 'RFoptions' to make this style
            ## the default
```

RMmodelFit-class *Class "RMmodelFit"*

Description

Extension of Class RMmodel which additionally contains the likelihood of the data w.r.t. the covariance model represented by the "RMmodel" part, the estimated trend of the data if it is a constant trend, and the residuals of the data w.r.t. the model. Objects of this class only occur as slots in the output of "RFfit".

Creating Objects

Objects are only ment to be created by the function RFfit

Slots

AIC: the AIC value for the ml estimation

AICc: the corrected AIC value for the ml estimation

BIC: the BIC value for the ml estimation

call: see RMmodel.

likelihood: numeric; the likelihood of the data w.r.t. the covariance model

name: see RMmodel.

par.model: see RMmodel.

par.general: see RMmodel.

param: vector of estimated parameters

residuals: array or of class RFsp; residuals of the data w.r.t. the trend model

submodels: see RMmodel.

trend: numeric; the estimated mean of the data (if a constant mean was specified in the model)

variab: vector of estimated variables. Variables are used in the internal representation and can be a subset of the parameters.

Extends

Class "RMmodel", directly.

Methods

[signature(x = "RMmodelFit"): enables accessing the slots via the "["-operator, e.g. x["likelihood"]

[<- signature(x = "RMmodelFit"): enables replacing the slots via the "["-operator

show signature(x = "RFfit"): returns the structure of x

print signature(x = "RFfit"): identical with show-method

anova performs a likelihood ratio test base on a chisq approximation

summary gives a summary

Author(s)

Alexander Malinowski <malinows@math.uni-goettingen.de>, Martin Schlather, <schlather@math.uni-mannheim.de>
http://ms.math.uni-mannheim.de/de/publications/software

See Also

RMmodel RFfit

Examples

```
# see RFfit
```

RMmodelgenerator-class

Class RMmodelgenerator

Description

Class for all functions of this package with prefix RM, i.e. all functions that generate objects of class RMmodel; direct extension of class function

Creating Objects

Objects should not be created by the user!

Slots

.Data: function; the genuine funtion that generates an objects of class RMmodel

type: character string; specifies the cathegory of RMmodel-function, see Details

domain: character string; specifies whether the corresponding function(s) depend on 1 or 2 variables, see Details

isotropy: character string; specifies the type of isotropy of the corresponding covariance model, see Details

operator: logical; specifies whether the underlying covariance model is an operator, see Details

monotone: character string; specifies the kind of monotonicity of the model

finiterange: logical; specifies whether the underlying covariance model has finite range, see Details

simpleArguments: logical. If TRUE than all the parameters are real valued (or integer valued).

maxdim: numeric; the maximal dimension, in which the corresponding model is a valid covariance model, see Details

vdim: numeric; dimension of the value of the random fiels at a single fixed location, equals 1 in most cases, see Details

Extends

Class `function`, directly.

Methods

show `signature(x = "RMmodel")`: returns the structure of `x`

print `signature(x = "RMmodel")`: identical with show-method

[`signature(x = "RMmodelgenerator")`: enables accessing the slots via the "["-operator, e.g. x["maxdim"]

[<- `signature(x = "RMmodelgenerator")`: enables replacing the slots via the "["-operator

Details

`type`**:** can be one of the following strings:

- `'tail correlation function'`**:** indicates that the function returns a tail correlation function (a subclass of the set of positive definite functions)
- `'positive definite'`**:** indicates that the function returns a covariance function (positive definite function)
- `'negative definite'`**:** indicates that the function returns a variogram model (negative definite function)
- `'process'`**:** functions of that type determine the class of processes to be simulated
- `'method for Gauss processes'`**:** methods to simulate Gaussian random fields
- `'method for Brown-Resnick processes'`**:** methods to simulate Brown-Resnick fields
- `'point-shape function'`**:** functions of that type determine the distribution of points in space
- `'distribution family'`**:** e.g. (multivariate) uniform distribtion, normal distribution, etc., defined in **RandomFields**. See RR for a complete list.
- `'shape function'`**:** functions used in, e.g., M3 processes (RPsmith)
- `'trend'`**:** RMtrend or a mixed model
- `'interface'`**:** indicates internal models which are usually not visible for the users. These functions are the internal representations of `RFsimulate`, `RFcov`, etc.. See RF for a complete list.
- `'undefined'`**:** some models can take different types, depending on the parameter values and/or the submodels
- `'other type'`**:** very very special internal functions, not belonging to any of the above types.

`domain`**:** can be one of the following strings:

- `'single variable'`**:** Function depending on a single variable
- `'kernel'`**:** model refers to a kernel, e.g., an non-stationary convariance function
- `'framework dependent'`**:** domain depends on the calling model
- `'mismatch'`**:** this option is used only internaly and should never appear

`isotropy`**:** can be one of the following strings:

- `'isotropic'`**:** indicates that the model is isotropic
- `'space-isotropic'`**:** indicates that the spatial part of a spatio-temporal model is isotropic

'zero-space-isotropic': this property refers to space-time models; the model is called zerospaceisotropic if it is isotropic as soon as the time-component is zero

'vector-isotropic': multivariate vector model (flow fields) have a different notion of isotropy

'symmetric': the most basic property of any covariance function or variogram model

'cartesian system','earth system','spherical system','cylinder system': different coordinate systems

'non-dimension-reducing': the property $f(x) = f(-x)^\top$ does not hold

'parameter dependent': indicates that the type of isotropy of the model depends on the parameters passed to the model; in particular parameters may be submodels if an operator model is considered

'<mismatch>': this option is used only internaly and should never appear

operator: if TRUE, the model requires at least one submodel

monotone: 'mismatch in monotonicity': used if a statement on the monotonocity does not make sense, e.g. for RRmodels

'submodel dependent monotonicity': only for operators, e.g. RMS

'previous model dependent monotonicity': internal; should not be used

'parameter dependent monotonicity': some models change their properties according to the parameters

'not monotone': none of the above cathegories; either the function is not monotone or properties are not known

'monotone': isotone or antitone

'Gneiting-Schaback class': function belonging to Euclid's hat in Gneiting's 1999 paper

'normal mixture': scale mixture of the Gaussian model

'completely monotone': completely monotone function

'Bernstein': Bernstein function

Note that

- 'not monotone' includes 'monotone' and 'Bernstein'
- 'monotone' includes 'Gneiting-Schaback class'
- 'Gneiting-Schaback class' includes 'normal mixture'
- 'normal mixture' includes 'completely monotone'

finiterange: if TRUE, the covariance of the model has finite range

maxdim: if a positive integer, maxdim gives the maximum dimension in which the model is a valid covariance model, can be Inf; vdim=-1 means that the actual maxdim depends on the parameters; vdim=-2 means that the actual maxdim depends on the submodel(s)

vdim: if a positive integer, vdim gives the dimension of the random field, i.e. univariate, bi-variate, ...; vdim=-1 means that the actual vdim depends on the parameters; vdim=-2 means that the actual vdim depends on the submodel(s)

Author(s)

Alexander Malinowski <malinows@math.uni-goettingen.de>; Martin Schlather, <schlather@math.uni-mannheim.de>
http://ms.math.uni-mannheim.de/de/publications/software

References

- Gneiting, T. (1999) Radial positive definite functions generated by Euclid's hat, *J. Multivariate Anal.*, **69**, 88-119.

See Also

RMmodel, RFgetModelNames

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
RFgetModelNames(group="type")
```

RMmodelsAdvanced *Advanced features of the mdoels*

Description

Here, further models and advanced comments for RMmodel are given. See also RFgetModelNames.

Details

Further stationary and isotropic models

RMaskey	Askey model (generalized test or triangle model)
RMbessel	Bessel family
RMcircular	circular model
RMcauchy	modified Cauchy family
RMcubic	cubic model (see Chiles \& Delfiner)
RMdagum	Dagum model
RMdampedcos	exponentially damped cosine
RMqexp	Variant of the exponential model
RMfractdiff	fractionally differenced process
RMfractgauss	fractional Gaussian noise
RMgengneiting	generalized Gneiting model
RMgneitingdiff	Gneiting model for tapering
RMhyperbolic	generalised hyperbolic model
RMlgd	Gneiting's local-global distinguisher
RMma	one of Ma's model
RMpenta	penta model (see Chiles \& Delfiner)
RMpower	Golubov's model
RMwave	cardinal sine

Variogram models (stationary increments/intrinsically stationary)

RMdewijsian	generalised version of the DeWijsian model
RMgenfbm	generalized fractal Brownian motion
RMflatpower	similar to fractal Brownian motion but always smooth at the origin

General composed models (operators)

Here, composed models are given that can be of any kind (stationary/non-stationary), depending on the submodel.

RMbernoulli	Correlation function of a binary field based on a Gaussian field
RMexponential	exponential of a covariance model
RMintexp	integrated exponential of a covariance model (INCLUDES ma2)
RMpower	powered variograms
RMqam	Porcu's quasi-arithmetric-mean model
RMS	details on the optional transformation arguments (var, scale, Aniso, proj).

Stationary and isotropic composed models (operators)

RMcutoff	Gneiting's modification towards finite range
RMintrinsic	Stein's modification towards finite range
RMnatsc	practical range
RMstein	Stein's modification towards finite range
RMtbm	Turning bands operator

Stationary space-time models

Here, most of the models are composed models (operators).

RMave	space-time moving average model
RMcoxisham	Cox-Isham model
RMcurlfree	curlfree (spatial) field (stationary and anisotropic)
RMdivfree	divergence free (spatial) vector valued field, (stationary and anisotropic)
RMiaco	non-separabel space-time model
RMmastein	Ma-Stein model
RMnsst	Gneiting's non-separable space-time model
RMstein	Stein's non-separabel space-time model
RMstp	Single temporal process
RMtbm	Turning bands operator

Multivariate/Multivariable and vector valued models See also the vignette 'multivariate'.

RMbiwm	full bivariate Whittle-Matern model (stationary and isotropic)
RMbigneiting	bivariate Gneiting model (stationary and isotropic)
RMcurlfree	curlfree (spatial) vector-valued field (stationary and anisotropic)
RMdelay	bivariate delay effect model (stationary)
RMdivfree	divergence free (spatial) vector valued field, (stationary and anisotropic)
RMexponential	functional returning e^C
RMkolmogorov	Kolmogorov's model of turbulence

RMmatrix	trivial multivariate model
RMmqam	multivariate quasi-arithmetic mean (stationary)
RMparswm	multivariate Whittle-Matern model (stationary and isotropic)
RMschur	element-wise product with a positive definite matrix
RMtbm	turning bands operator
RMvector	vector-valued field (combining RMcurlfree and RMdivfree)

Non-stationary models

RMnonstwm	one of Stein's non-stationary Wittle-Matern model

Models related to max-stable random fields (tail correlation functions)

RMaskey	Askey model (generalized test or triangle model) with $\alpha \geq [dim/2] + 1$
RMbernoulli	Correlation function of a binary field based on a Gaussian field
RMbr2bg	Operator relating a Brown-Resnick process to a Bernoulli process
RMbr2eg	Operator relating a Brown-Resnick process to an extremal Gaussian process
RMbrownresnick	tail correlation function of Brown-Resnick process
RMgencauchy	generalized Cauchy family with $\alpha \leq 1/2$
RMm2r	shape functions related to max-stable processes
RMm3b	shape functions related to max-stable processes
RMmatern	Whittle-Matern model with $\nu \leq 1$
RMmps	shape functions related to max-stable processes
RMschlather	tail correlation function of the extremal Gaussian field
RMstable	symmetric stable family or powered exponential model with $\alpha \leq 1$
RMwhittle	Whittle-Matern model, alternative parametrization with $\nu \leq 1/2$

Other covariance models

RMuser	User defined model

Auxiliary models There are models or better function that are not covariance functions, but can be part of a model definition. See **Auxiliary RMmodels.**

Note

- Note that, instead of the named arguments, a single argument k can be passed. This is possible if all the arguments are scalar. Then k must have length equal to the number of arguments.
- If a argument equals NULL the argument is not set (but must be a valid name).
- Aniso can be given also by RMangle instead by a matrix
- Note also that a completely different possibility exists to define a model, namely by a list. This format allows for easy flexible models and modifications (and some few more options, as well as some abbreviations to the model names, see PrintModelList()). Here, the argument var, scale, Aniso and proj must be passed by the model RMS. For instance,

  - `model <- RMexp(scale=2, var=5)`
    is equivalent to
    `model <- list("RMS", scale=2, var=5, list("RMexp"))`
    The latter definition can be also obtained by
    `summary(RMexp(scale=2, var=5))`
  - `model <- RMnsst(phi=RMgauss(var=7), psi=RMfbm(alpha=1.5),scale=2, var=5)`
    is equivalent to
    `model <- list("RMS", scale=2, var=5,`
    `list("RMnsst", phi=list("RMS", var=7, list("RMgauss")),`
    `psi=list("RMfbm", alpha=1.5)) ).`

- Instead of a deterministic value, a distribution family might be given, see `RRmodels`. The latter starts with `RR` or is distribution family, e.g. `norm`, `exp`, or `unif`. Note that the effect of the distribution family varies between the different processes:
  - in Max-stable fields and `RPpoisson`, a new realisation of the distribution is drawn for each shape function
  - in all the other cases: a realisation is only drawn once. This effects, in particular, Gaussian fields with argument `n>1`, where all the realisations are based on the same realisation out of the distribution.

  MLE ist not programmed yet.

  Very advanced: In case of a distribution family, its arguments might be again given by a RM-model. Note that checking the validity of the arguments is rather limited for such complicated models, in general.

  See also RMmodelsAuxiliary and Baysian.

All models have secondary names that stem from **RandomFields** versions 2 and earlier and that can also be used as strings in the list notation. See `RFgetModelNames(internal=FALSE)` for the full list.

Author(s)

Alexander Malinowski, <malinowski@math.uni-mannheim.de>

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.
- Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.
- Schlather, M. (2011) Construction of covariance functions and unconditional simulation of random fields. In Porcu, E., Montero, J.M. and Schlather, M., *Space-Time Processes and Challenges Related to Environmental Problems.* New York: Springer.
- Yaglom, A.M. (1987) *Correlation Theory of Stationary and Related Random Functions I, Basic Results.* New York: Springer.
- Wackernagel, H. (2003) *Multivariate Geostatistics.* Berlin: Springer, 3nd edition.

See Also

RFformula, RMmodels, RMmodelsAuxiliary

'multivariate', a vignette for multivariate geostatistics

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
RFgetModelNames(type="positive", group.by=c("domain", "isotropy"))
```

RMmppplus *Mixture of shape functions*

Description

RMmppplus is a multivariate covariance model which depends on up to 10 submodels $C_0, C_1, ..., C_10$.

It is used to gether with RPsmith it allowed for mixed moving maxima with a finite number of shape functions.

Usage

```
RMmppplus(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, p)
```

Arguments

C0	an RMmodel.
C1,C2,C3,C4,C5,C6,C7,C8,C9	optional; each an RMmodel.
p	vector of probabilities for the shape functions. The probabilities should add up to 1. The length of the vector equals the number of given submodels

Value

RMmppplus returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMplus, RMmodel, RFsimulate, RFfit, RPsmith.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RMmqam *multivariate quasi-arithmetic mean*

Description

RMqam is a multivariate stationary covariance model depending on a submodel ϕ such that $\psi(\cdot) := \phi(\sqrt{(\cdot)})$ is completely monotone, and depending on further stationary covariance models C_i. The covariance is given by

$$C_{ij}(h) = \phi(\sqrt{(\theta_i(\phi^{-1}(C_i(h)))^2 + \theta_j(\phi^{-1}(C_j(h)))^2)})$$

where ϕ is a completely monotone function, C_i are suitable covariance functions and $\theta_i \geq 0$ such that $\sum_i \theta_i = 1$.

Usage

```
RMmqam(phi, C1, C2, C3, C4, C5, C6, C7, C8, C9, theta, var, scale, Aniso, proj)
```

Arguments

phi	a valid covariance RMmodel that is a normal scale mixture. See, for instance, RFgetModelNames(monotone="normal mixture")
C1, C2, C3, C4, C5, C6, C7, C8, C9	optional further stationary RMmodel
theta	is a vector of values in $[0, 1]$, summing up to 1.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Note that $\psi(\cdot) := \phi(\sqrt{(\cdot)})$ is completely monotone if and only if ϕ is a valid covariance function for all dimensions, e.g. RMstable, RMgauss, RMexponential.

Warning: RandomFields cannot check whether the combination of ϕ and C_i is valid.

Value

RMmqam returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Porcu, E., Mateu, J. & Christakos, G. (2009) Quasi-arithmetic means of covariance functions with potential applications to space-time data. *Journal of Multivariate Analysis*, **100**, 1830–1844.

See Also

RMqam, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RMmult *Multiplication of Random Field Models*

Description

RMmult is a multivariate covariance model which depends on up to 10 submodels $C_0, C_1, ..., C_10$. In general, realizations of the created RMmodel are pointwise product of independent realizations of the submodels.

In particular, if all submodels are given through a covariance function, the resulting model is defined through its covariance function, which is the product of the submodels' covariances.

Usage

```
RMmult(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, var, scale, Aniso, proj)
```

Arguments

C0	an RMmodel.
C1,C2,C3,C4,C5,C6,C7,C8,C9	optional; each an RMmodel.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above model remains unmodified.

Details

RMmodels can also be multiplied via the *-operator, e.g.: C0 * C1

The global arguments scale,Aniso,proj of RMmult are multiplied to the corresponding argument of the submodels (from the right side). E.g.,
RMmult(Aniso=A1, RMexp(Aniso=A2), RMspheric(Aniso=A3))
equals
RMexp(Aniso=A2 %*% A1), RMspheric(Aniso=A3 %*% A1)

In case that all submodels are given through a covariance function, the global argument var of RMmult is multiplied to the product covariance of RMmult.

Value

RMmult returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMplus, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

# separable, multiplicative model
model <- RMgauss(proj=1) * RMexp(proj=2, scale=5)
z <- RFsimulate(model=model, 0:10, 0:10, n=4)
plot(z)
```

RMnatsc *Natural scale*

Description

RMnatsc is a stationary isotropic covariance model that depends on a stationary isotropic covariance model ϕ. The covariance is given by

$$C(h) = \phi(h/s)$$

, where the argument s is chosen by RMnatsc such that the practical range or the mathematical range, if finite) is 1.

Usage

```
RMnatsc(phi, var, scale, Aniso, proj)
```

Arguments

phi	a stationary isotropic covariance RMmodel.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

For internal use only.

Value

RMnatsc returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMnatsc(RMexp())
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(RMexp(), model=model)
RFcov(model, 1)
```

RMnonstwm *RMnonstwm*

Description

RMnonstwm is a covariance model whose corresponding covariance C is given by

$$C(x,y) = \Gamma(\mu)\Gamma(\nu(x))^{-1/2}\Gamma(\nu(y))^{-1/2}W_\mu(|x-y|)$$

where $\mu = [\nu(x)+\nu(y)]/2$ and W_μ is the covariance of the RMwhittle model and ν is a positive function or a constant.

Usage

```
RMnonstwm(nu, var, scale, Aniso, proj)
```

Arguments

nu	positive argument defining the smoothness of the random field, or it is an arbitrary shape function.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMnonstwm returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Stein, M. (2005) Nonstationary Spatial Covariance Functions. Tech. Rep., 2005

See Also

RMwhittle, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RMnsst *Non-Separable Space-Time model*

Description

RMnsst is a univariate stationary spaceisotropic covariance model whose corresponding covariance is given by

$$C(h,u) = (\psi(u)+1)^{-\delta/2}\phi(h/\sqrt{(}\psi(u)+1))$$

Usage

```
RMnsst(phi, psi, delta, var, scale, Aniso, proj)
```

Arguments

phi	is normal mixture RMmodel, cf. RFgetModelNames(monotone="normal mixture")
psi	is a variogram RMmodel.
delta	a numerical value; must be greater than or equal to the spatial dimension of the field
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model is used for space-time modelling where the spatial component is isotropic.

Value

RMnsst returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T. (1997) Normal scale mixtures and dual probability densitites, *J. Stat. Comput. Simul.* **59**, 375-384.
- Gneiting, T. (2002) Nonseparable, stationary covariance functions for space-time data, *JASA* **97**, 590-600.
- Gneiting, T. and Schlather, M. (2001) Space-time covariance models. In El-Shaarawi, A.H. and Piegorsch, W.W.: *The Encyclopedia of Environmetrics.* Chichester: Wiley.
- Schlather, M. (2010) On some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMnsst(phi=RMgauss(), psi=RMfbm(alpha=1), delta=2)
x <- seq(0, 10, if (interactive()) 0.25 else 1)
plot(model, dim=2)
plot(RFsimulate(model, x=x, y=x))
```

RMnugget *Nugget Effect Covariance Model*

Description

RMnugget is a multivariate stationary isotropic covariance model called "nugget effect". The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given for i, j in $1, ...,$vdim by

$$C_{ij}(r) = \delta_i j 1_0(r),$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise.

Usage

```
RMnugget(tol, vdim, var, scale, Aniso, proj)
```

Arguments

tol	Only for advanced users. See RPnugget.
vdim	Only for advanced users. See RPnugget.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Note that the argument scale does not affect the covariance model; Aniso has an effect in case of zonal anisotropy.

The nugget effect belongs to Gaussian white noise and is often used for modeling measurement errors.

The locations at a distance less than or equal to nugget.tol are considered as being identical. This strategy applies to the simulation method and the covariance function itself. Hence, the covariance function is only positive definite if nugget.tol=0.0. However, if the anisotropy matrix does not have full rank and nugget.tol=0.0 then, the simulations are likely to be odd. The value of nugget.tol should be of order 10^{-15}.

Value

RMnugget returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMnugget(Aniso=matrix(1, nr=2, nc=2))
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(RFsimulate(model, x=x, y=x, tol=1e-10))
```

RMparswm *Parsimonious Multivariate Whittle Matern Model*

Description

RMparswm is a multivariate stationary isotropic covariance model whose corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given for $i, j \in \{1, 2\}$ by

$$C_{ij}(r) = c_{ij} W_{\nu_{ij}}(r).$$

Here W_ν is the covariance of the RMwhittle model.

RMparswmX ist defined as

$$\rho_{ij} C_{ij}(r)$$

where ρ_{ij} is any covariance matrix.

Usage

```
RMparswm(nudiag, var, scale, Aniso, proj)
RMparswmX(nudiag, rho, var, scale, Aniso, proj)
```

Arguments

nudiag	a vector of arbitrary length of positive values; each entry positive; the vector $(\nu_{11}, \nu_{22}, ...)$. The offdiagonal elements ν_{ij} are calculated as $0.5(\nu_{ii} + \nu_{jj})$.
rho	any positive definite $m \times m$ matrix; here m equals length(nudiag) For the calculation of c_{ij} see Details.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

In the equation above we have

$$c_{ij} = \rho_{ij} \sqrt{G_{ij}}$$

and

$$G_{ij} = \frac{\Gamma(\nu_{11} + d/2)\Gamma(\nu_{22} + d/2)\Gamma(\nu_{12})^2}{\Gamma(\nu_{11})\Gamma(\nu_{22})\Gamma(\nu_{12} + d/2)^2}$$

where Γ is the Gamma function and d is the dimension of the space.

Note that the definition of RMparswmX is RMschur(M=rho, RMparswm(nudiag, var, scale, Aniso, proj)) .

Value

RMparswm returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Gneiting, T., Kleiber, W., Schlather, M. (2010) Matern covariance functions for multivariate random fields *JASA*

See Also

RMbiwm, RMwhittle, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

rho <- matrix(nc=3, c(1, 0.5, 0.2, 0.5, 1, 0.6, 0.2, 0.6, 1))
model <- RMparswmX(nudiag=c(1.3, 0.7, 2), rho=rho)
x.seq <- y.seq <- seq(-10, 10, if (interactive()) 0.1 else 5)
z <- RFsimulate(model = model, x=x.seq, y=y.seq)
plot(z)
```

RMpenta *Penta Covariance Model*

Description

RMpenta is a stationary isotropic covariance model, which is valid only for dimensions $d \leq 3$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (1 - \frac{22}{3}r^2 + 33r^4 - \frac{77}{2}r^5 + \frac{33}{2}r^7 - \frac{11}{2}r^9 + \frac{5}{6}r^{11})1_{[0,1]}(r).$$

Usage

```
RMpenta(var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is only valid for dimension $d \leq 3$.

It has a 4 times differentiable covariance function with compact support (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 84).

Value

RMpenta returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMpenta()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMplus *Addition of Random Field Models*

Description

RMplus is an additive covariance model which depends on up to 10 submodels $C_0, C_1, ..., C_10$. In general, realizations of the created RMmodel are pointwise sums of independent realizations of the submodels.

In particular, if all submodels are given through a covariance function, the resulting model is defined through its covariance function, which is the sum of the submodels' covariances. Analogously, if all submodels are given through a variogram.

Usage

```
RMplus(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, var, scale, Aniso, proj)
```

Arguments

C0	an RMmodel.
C1,C2,C3,C4,C5,C6,C7,C8,C9	optional; each an RMmodel.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above model remains unmodified.

Details

RMmodels can also be summed up via the +-operator, e.g.: C0 + C1

The global arguments var,scale,Aniso,proj of RMplus are multiplied to the corresponding arguments of the submodels (from the right side).

Value

RMplus returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmult, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMplus(RMgauss(), RMnugget(var=0.1))
model2<- RMgauss() + RMnugget(var=0.1)
plot(model, "model.+"=model2, type=c("p", "l"), pch=20, xlim=c(0,3)) # the same
```

RMpolygon *RMpolygon*

Description

RMpolygon referes to the indicator function of a typical Poisson polygon, used for instance in the (mixed) Storm process.

Usage

```
RMpolygon(lambda)
```

Arguments

lambda	intensity of the hyperplan process creating the random shape function The default value is 1.

Author(s)

Felix Ballani, http://www.mathe.tu-freiberg.de/sto/mitarbeiter/felix-ballani

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Poission polygons / Poisson hyperplane tesselation

- Lantuejoul, C. (2002) *Geostatistical Simulation: Models and Algorithms.* Springer.

Poisson storm process

- Lantu\'ejoul, C., Bacro, J.N., Bel L. (2011) Storm processes and stochastic geometry. *Extremes*, **14**(4), 413-428.

Mixed Poisson storm process

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

RMball RMspheric, RFsimulate, RMmodel.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RMpower *Power operator for Variograms and Covariance functions*

Description

RMpower yields a variogram or covariance model from a given variogram or covariance model. The variogram γ of the model is given by

$$\gamma = \phi^\alpha$$

if ϕ is a variogram model. The covariance C of the model is given by

$$C(h) = \phi(0) - (\phi(0) - \phi(h))^\alpha$$

if ϕ is a covariance model.

Usage

```
RMpower(phi, alpha, var, scale, Aniso, proj)
```

Arguments

phi	a valid RMmodel; either a variogram model or a covariance model
alpha	a numerical value in the interval [0,1]
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

If γ is a variogram, then γ^α is a valid variogram for α in the interval [0,1].

Value

RMpower returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Schlather, M. (2012) Construction of covariance functions and unconditional simulation of random fields. In Porcu, E., Montero, J. M., Schlather, M. *Advances and Challenges in Space-time Modelling of Natural Events*, Springer, New York.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMpower(RMgauss(), alpha=0.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMqam *Quasi-arithmetic mean*

Description

RMqam is a univariate stationary covariance model depending on a submodel ϕ such that $\psi(\cdot) := \phi(\sqrt{(\cdot)})$ is completely monotone, and depending on further stationary covariance models C_i. The covariance is given by

$$C(h) = \phi(\sqrt{(\sum_i \theta_i(\phi^{-1}(C_i(h)))^2)})$$

Usage

```
RMqam(phi, C1, C2, C3, C4, C5, C6, C7, C8, C9, theta, var, scale, Aniso, proj)
```

Arguments

phi	a valid covariance RMmodel that is a normal scale mixture. See, for instance, RFgetModelNames(monotone="normal mixture")
C1, C2, C3, C4, C5, C6, C7, C8, C9	optional further univariate stationary RMmodel.
theta	a vector with positive entries
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Note that $\psi(\cdot) := \phi(\sqrt{(\cdot)})$ is completely monotone if and only if ϕ is a valid covariance function for all dimensions, e.g. RMstable, RMgauss, RMexponential.

Warning: RandomFields cannot check whether the combination of ϕ and C_i is valid.

Value

RMqam returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Porcu, E., Mateu, J. & Christakos, G. (2007) Quasi-arithmetic means of covariance functions with potential applications to space-time data. Submitted to Journal of Multivariate Analysis.

See Also

RMmqam, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMqam(phi=RMgauss(), RMexp(), RMgauss(),
               theta=c(0.3, 0.7), scale=0.5)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMqexp *Variant of the exponential model*

Description

The covariance function is

$$C(x) = (2e^{-x} - \alpha e^{-2x})/(2 - \alpha)$$

Usage

```
RMqexp(alpha, var, scale, Aniso, proj)
```

Arguments

alpha	value in $[0, 1]$
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMqexp returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- ?

See Also

RMmodel,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMqexp(alpha=0.95, scale=0.2)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

`RMrational` *Rational function*

Description

Defines a simple rational function.

$$f(h) = (a_1 + a_2 z(h))/(1 + z(h))$$

where

$$z(h) = h^\top A A^\top h$$

Usage

```
RMrational(A, a)
```

Arguments

`A`	a $d \times d$ matrix
`a`	a vector of one or two components; the second component has default value zero.

Value

`RMrational` returns an object of class `RMmodel`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

`RMmodel`, S10

Examples

```
# see S10
```

RMrotat *Rotation matrices*

Description

RMrotat and RMrotation are auxiliary space-time functions that create some rotation

$$f(h,t) = s(\cos(\phi t)h_1 + \sin(\phi t)h_2)/\|h\|$$

and

$$f(h,t) = (\cos(\phi t)h_1 + \sin(\phi t)h_2, -\sin(\phi t)h_1 + \cos(\phi t)h_2, t)$$

respectively

Usage

```
RMrotat(speed, phi)
RMrotation(phi)
```

Arguments

speed	real value s
phi	angle

Details

RMrotat and RMrotation are space-time models for two-dimensial space.

Value

RMrotat and RMrotation return an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, S10

Examples

```
# see S10
```

RMS *Scaling operator*

Description

RMS is an operator that modifies the variance and the coordinates or distances of a submodel ϕ by

$$C(h) = v * \phi(A * h/s).$$

Most users will never call RMS directly, see the details.

Usage

```
RMS(phi, var, scale, Aniso, proj, anisoT)
```

Arguments

phi	submodel
var	is the optional variance parameter v,
scale	scaling parameter s which is positive
Aniso	matrix or RMmodel. The optional anisotropy matrix A, multiplied from the right by a distance vector x, i.e. Ax
proj	is the optional projection vector which defines a diagonal matrix of zeros and ones and proj gives the positions of the ones (integer values between 1 and teh dimension of x).
anisoT	the transpose of the anisotropy matrix B, multiplied from the left by a distance vector x, i.e. $x^\top B$.

Details

The call in the usage section is equivalent to `phi(..., var, scale, anisoT, Aniso, proj)`, where phi has to be replaced by a valid RMmodel

Most user will never call RMS directly.

Value

RMS returns an object of class RMmodel.

Note

At most one of the arguments, Aniso, anisoT and proj may be given at the same time.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model1 <- RMS(RMexp(), scale=2)
model2 <- RMexp(scale=2)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
print(all(RFcov(model1, x) == RFcov(model2, x))) # TRUE
```

RMschlather *Covariance Model for binary field based on Gaussian field*

Description

RMschlather gives the tail correlation function of the extremal Gaussian process, i.e.

$$C(h) = 1 - \sqrt{(1 - \phi(h)/\phi(0))/2}$$

where ϕ is the covariance of a stationary Gaussian field.

Usage

```
RMschlather(phi, var, scale, Aniso, proj)
```

Arguments

phi	covariance function of class RMmodel.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model yields the tail correlation function of the field that is returned by RPschlather

Value

RMschlather returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RPschlather RMmodel, RFsimulate,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## This examples considers an extremal Gaussian random field
## with Gneiting's correlation function.

## first consider the covriance model and its corresponding tail
## corrlation function
model <- RMgneiting()
plot(model, model.tail.corr.fct=RMschlather(model),  xlim=c(0, 5))


## the extremal Gaussian field with the above underlying
## correlation function that has the above tail correlation function tcf
x <- seq(0, 10, if (interactive()) 0.1 else 3)
z <- RFsimulate(RPschlather(model), x)
plot(z)

## Note that in RFsimulate R-P-schlather was called, not R-M-schlather.
## The following lines give a Gaussian random field with corrlation
## function equal to the above tail correlation function.
z <- RFsimulate(RMschlather(model), x)
plot(z)
```

RMschur *Schur product*

Description

The covariance function is

$$C(x) = M * \phi(x)$$

where '*' denotes the Schur product, i.e. elementwise multiplication

Usage

```
RMschur(phi, M,  diag, rhored, var, scale, Aniso, proj)
```

Arguments

`phi`	covariance function of class `RMmodel`.
`M`	constant $n \times n$ covariance matrix of the same size as multivariate model `phi`
`diag,rhored`	alternative way of passing `M`: `diag` is a vector of variances, `rhored` is a vector containing the correlations of lower triangle of the `M`
`var,scale,Aniso,proj`	optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

Value

`RMschur` returns an object of class `RMmodel`

Author(s)

Martin Schlather, `<schlather@math.uni-mannheim.de>`

References

- ?

See Also

`RMmodel`, `RMmatrix`,

Examples

```
model <- RMschur(M=matrix(c(2, 1, 1, 1), ncol=2), RMparswm(nudiag=c(0.5, 2)))
plot(model)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(RFsimulate(model, x=x))
```

`RMsign` *Random sign*

Description

`RMsign` defines a random sign. It can be used as part of the model definition of a Poisson field.

Usage

```
RMsign(phi, p)
```

Arguments

`phi`	shape function of class `RMmodel`.
`p`	probability of keeping the sign

Details

RMsign changes the sign of the shape function phi with probability 1-p and keeps it otherwise.

Value

RMsign returns an object of class RMmodel

Note

Random univariate or multivariate objects are usually start with RR not with RM. This is an exception here, as it operates on shape functions.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel RR.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RPpoisson(RMsign(RMtent(), p=0.8))
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(RFsimulate(model, x=x))
```

RMspheric *Spherical Covariance Model*

Description

RMspheric is a stationary isotropic covariance model which is only valid up to dimension 3. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = \left(1 - \frac{3}{2}r + \frac{1}{2}r^3\right) 1_{[0,1]}(r)$$

Usage

```
RMspheric(var, scale, Aniso, proj)
```

Arguments

var,scale,Aniso,proj
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This covariance model is valid only for dimensions less than or equal to 3.

The covariance function has a finite range.

Value

RMspheric returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMspheric()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMstable *Stable Family / Powered Exponential Model*

Description

RMstable is a stationary isotropic covariance model belonging to the so called stable family. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = e^{-r^\alpha}$$

where $\alpha \in (0, 2]$.

Usage

```
RMstable(alpha, var, scale, Aniso, proj)
RMpoweredexp(alpha, var, scale, Aniso, proj)
```

Arguments

`alpha`	a numerical value; should be in the interval (0,2] to provide a valid covariance function for a random field of any dimension.
`var,scale,Aniso,proj`	optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

Details

The parameter α determines the fractal dimension D of the Gaussian sample paths:

$$D = d + 1 - \frac{\alpha}{2}$$

where d is the dimension of the random field. For $\alpha < 2$ the Gaussian sample paths are not differentiable (cf. Gelfand et al., 2010, p. 25).

Each covariance function of the stable family is a normal scale mixture.

The stable family includes the exponential model (see `RMexp`) $\alpha = 1$ and the Gaussian model (see `RMgauss`) for $\alpha = 2$.

The model is called stable, because in the 1-dimensional case the covariance is the characteristic function of a stable random variable (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 90).

Value

`RMstable` returns an object of class `RMmodel`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Covariance function

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.
- Diggle, P. J., Tawn, J. A. and Moyeed, R. A. (1998) Model-based geostatistics (with discussion). *Applied Statistics* **47**, 299–350.
- Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.

Tail correlation function (for $\alpha \in (0, 1]$)

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

`RMexp`, `RMgauss`, `RMmodel`, `RFsimulate`, `RFfit`.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMstable(alpha=1.9, scale=0.4)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMstein *Stein nonseparable space-time model*

Description

RMstein is a univariate stationary covariance model whose corresponding covariance function only depends on the difference h between two points and is given by

$$C(h,t) = W_\nu(y) - (< h, z > t)/((\nu-1)(2\nu+d)) * W_{\nu-1}(y)$$

Here W_ν is the covariance of the RMwhittle model with smoothness parameter ν; $y = \|(h,t)\|$ is the norm of the vector (h,t), d is the dimension of the space on which the random field is considered.

Usage

```
RMstein(nu, z, var, scale, Aniso, proj)
```

Arguments

nu	numerical value; greater than 1; smoothness parameter of the RMwhittle model
z	a vector; the norm of z must be less or equal to 1.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

See Stein (2005).

Value

RMstein returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Stein, M.L. (2005) Space-time covariance functions. *J. Amer. Statist. Assoc.* **100**, 310-321. Equation (8).

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMstein(nu=1.5, z=0.9)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(RFsimulate(model, x=x, y=x))
```

RMstp *Single temporal process*

Description

RMstp is a univariate covariance model which depends on a normal mixture submodel ϕ. The covariance is given by

$$C(x,y) = |S_x|^{1/4}|S_y|^{1/4}|A|^{-1/2}\phi(Q(x,y)^{1/2})$$

where

$$Q(x,y) = c^2 - m^2 + h^t(S_x + 2(m+c)M)A^{-1}(S_y + 2(m-c)M)h,$$

$$c = -z^t h + \xi_2(x) - \xi_2(y),$$

$$A = S_x + S_y + 4Mhh^tM$$

$$m = h^tMh$$

$$h = x - y$$

Usage

```
RMstp(xi, phi, S, z, M, var, scale, Aniso, proj)
```

Arguments

xi	arbitrary univariate function on R^d
phi	an RMmodel that is a normal mixture model, cf. RFgetModelNames(monotone="normal mixture")
S	functions that returns strictly positive definite $d \times d$
z	arbitrary vector, $z \in R^d$
M	an arbitrary, symmetric $d \times d$ matrix
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

See Schlather (2008) formula (13). The model allows for mimicking cyclonic behaviour.

Value

RMstp returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Paciorek C.J., and Schervish, M.J. (2006) Spatial modelling using a new class of nonstationary covariance functions, *Environmetrics* **17**, 483-506.
- Schlather, M. (2010) Some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

See Also

RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMstp(xi = RMrotat(phi= -2 * pi, speed=1),
               phi = RMwhittle(nu = 1),
               M=matrix(nc=3, rep(0, 9)),
               S=RMetaxxa(E=rep(1, 3), alpha = -2 * pi,
                          A=t(matrix(nc=3, c(2, 0, 0, 1, 1 , 0, 0, 0, 0))))
              )
x <- seq(0, 10, if (interactive()) 0.7 else 5)
plot(RFsimulate(model, x=x, y=x, z=x))
```

RMtbm *Turning Bands Method*

Description

RMtbm is a univariate stationary isotropic covariance model in dimension reduceddim which depends on a univariate stationary isotropic covariance ϕ in a bigger dimension fulldim. For formulas for the covariance function see details.

Usage

```
RMtbm(phi, fulldim, reduceddim, layers, var, scale, Aniso, proj)
```

Arguments

phi, fulldim, reduceddim, layers
: see RPtbm.

var,scale,Aniso,proj
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The turning bands method stems from the 1:1 correspondence between the isotropic covariance functions of different dimensions. See Gneiting (1999) and Strokorb and Schlather (2014).

The standard case reduceddim=1 and fulldim=3. If only one of the arguments are given, then the difference of the two arguments equals 2.

For d == n + 2, where n=reduceddim and d==fulldim the original dimension, we have

$$C(r) = \phi(r) + r\phi'(r)/n$$

which, for n=1 reduced to the standard TBM operator

$$C(r) = \frac{d}{dr} r\phi(r)$$

For d == 2 && n == 1 we have

$$C(r) = \frac{d}{dr} \int_0^r \frac{u\phi(u)}{\sqrt{r^2 - u^2}} du$$

'Turning layers' is a generalization of the turning bands method, see Schlather (2011).

Value

RMtbm returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Turning bands

- Gneiting, T. (1999) On the derivatives of radial positive definite function. *J. Math. Anal. Appl*, **236**, 86-99
- Matheron, G. (1973). The intrinsic random functions and their applications. *Adv . Appl. Probab.*, **5**, 439-468.
- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

Turning layers

- Schlather, M. (2011) Construction of covariance functions and unconditional simulation of random fields. In Porcu, E., Montero, J.M. and Schlather, M., *Space-Time Processes and Challenges Related to Environmental Problems.* New York: Springer.

See Also

RPtbm, RFsimulate.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- seq(0, 10, if (interactive()) 0.02 else 2)
model <- RMspheric()
plot(model, model.on.the.line=RMtbm(RMspheric()), xlim=c(-1.5, 1.5))

z <- RFsimulate(RPtbm(model), x, x)
plot(z)
```

RMtrafo *Transformation of coordinate systems*

Description

The functions transform a coordinate system into another. Currently, essentially only from the earth system to cartesian.

RMtrafo is the internal basic function that also allows to reduce vectors to their norm.

Usage

```
RMtrafo(isotropy)
RFearth2cartesian(coord, units=NULL)
RFearth2dist(coord, units=NULL, ...)
```

Arguments

isotropy	one of the values RC_ISOTROPIC, RC_SPACEISOTROPIC, RC_CARTESIAN_COORD,
coord	matrix or vector of earth coordinates
units	"km" or "miles"; if not given and RFoptions()$general$units != "" the latter is used. Otherwise "km".
...	the optional arguments of dist

Details

The functions transform between different coordinate systems.

Value

The function RMtrafo returns a matrix, in general. For fixed column, the results, applied to each row of the matrix, are returned.

The function RFearth2cartesian returns a matrix in one-to-one correspondance with coord assuming that the earth is an ellipsoid.

The function RFearth2dist calculates distances, cf. dist, assuming that the earth is an ellipsoid.

Note

Important options are units and coordinate_system, see RFoptions.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

For calculating the earth coordinates as ellipsoid:

- en.wikipedia.org/wiki/Geographic_coordinate_system
- nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

See Also

linkconstants, RMangle

Examples

```
data(weather)
(coord <- weather[1:5, 3:4])

(z <- RFfctn(RMtrafo(RC_CARTESIAN_COORD), coord))
(z1 <- RFearth2cartesian(coord)) ## equals t(z)
z1 - t(z) ## 0, i.e., z1 and t(z) are the same
dist(t(z))


(d <- RFearth2dist(coord))
d - dist(t(z)) ## 0, i.e., d and dist(t(z)) are the same
```

RMtrend *Trend Model*

Description

RMtrend is a pure trend model with covariance 0.

Usage

```
RMtrend(mean, plane, polydeg, polycoeff, arbitraryfct, fctcoeff)
```

Arguments

mean	optional; should be a vector of length p, where p is the number of variables taken into account by the corresponding multivariate random field $(Z_1(\cdot), \ldots, Z_p(\cdot))$; the i-th component of mean is interpreted as constant mean of $Z_i(\cdot)$.
plane	optional; should be a $d \times p$-matrix where d is the dimension of the random field and p the number of variables considered; corresponds to a trend described by p hyperplanes. The mean of $Z_i(x)$ with $x = (x_1, \ldots, x_d)$ is given by $$plane_{1i}x_1 + plane_{2i}x_2 + \ldots + plane_{di}x_d.$$
polydeg	optional; should be an integer vector of length p; indicates that the mean of $Z_i(\cdot)$ is given by a multivariate polynomial of degree less than or equal to the i-th component of polydeg; the coefficients are either assumed to be unknown or to be given by polycoeff.

`polycoeff` optional; should be a vector of length

$$\binom{polydeg_1 + d}{d} + \ldots + \binom{polydeg_p + d}{d}$$

which is the number of monomial basis functions up to degree `polydeg_1`, ..., `polydeg_p` in a d-dimensional space. Each component of `polycoeff` gives the coefficient to one basis function; these products are added providing p trend polynomials.
For the order of the monomial basis functions see details. CAUTION: This argument should be used by advanced users only as the order might be sophisticated. In many cases it is recommended to use `arbitraryfct` and `fctcoeff` instead.

`arbitraryfct` optional; should be a p-variate function; the i-th component of `arbitraryfct` describes the trend surface of $Z_i(\cdot)$; the arguments of this function should be location (and time) corresponding to the random field to be modelled. If `RMtrend` is used in the model definition of the functions `RFsimulate`, `RFfit` or `RFinterpolate`, the names of the arguments should be given by `x, y, z, T`.

`fctcoeff` optional; should be numerical; determines the coefficient belonging to `arbitraryfct`, i.e. the trend is given by `fctcoeff * arbitraryfct`. Note that the coefficient is the same for each component of `arbitraryfct`; `fctcoeff` is ignored if `arbitraryfct=NULL`.

Details

If different trend arguments are given, the corresponding trend components are added. Equivalently, `+` (see `RFformula`) or `RMplus` can be used to add trend terms.

Note that this function refers to trend surfaces in the geostatistical framework. Fixed effects in the mixed models framework are implemented, as well (see `RFformula`).

The order of the monomial basis functions and the corresponding coefficients given by `polycoeff` is the following:
The first $\binom{polydeg_1+d}{d}$ components belong to the trend polynomial of the first variable, the following $\binom{polydeg_2+d}{d}$ ones to the second one, and so on. Within one trend polynomial the monomial basis functions are ordered by the powers in an ascending way such that the power of the first component varies fastest; e.g. the monomial basis functions up to degree k in a two-dimensional space are given by

$$1, x, \ldots, x^k, y, xy, \ldots, x^{k-1}y, y^2, \ldots, xy^{k-1}, y^k.$$

Value

`RMtrend` returns an object of class `RMmodel`.

Author(s)

Marco Oesting, <oesting@math.uni-mannheim.de>

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Chiles, J. P., Delfiner, P. (1999) *Geostatistics: Modelling Spatial Uncertainty.* New York: John Wiley & Sons.

See Also

RMmodel, RFformula, RFsimulate, RMplus

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
###################################################
# Example 1: #
# Simulate from model with a plane trend surface #
###################################################

#trend: 1 + x - y, cov: exponential with variance 0.01
model <- ~ RMtrend(mean=1, plane = c(1,-1)) + RMexp(var=0.01)
#equivalent model:
model <- ~ RMtrend(polydeg=1,polycoeff=c(1,1,-1)) + RMexp(var=0.01)
#Simulation
x <- 0:10
simulated <- RFsimulate(model=model, x=x, y=x)
plot(simulated)
```

RMtruncsupport *Random sign*

Description

It may be used to truncate the support of a shape function when Poisson fields or M3 processes are created.

Usage

```
RMtruncsupport(phi, radius)
```

Arguments

phi	function of class RMmodel.
radius	truncation at radius

Value

RMtruncsupport returns an object of class RMmodel.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M. (2002) Models for stationary max-stable random fields. *Extremes* **5**, 33-44.

See Also

RMmodel, RMmatrix, RPpoisson

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

RMuser *User Defined Function*

Description

RMuser allows for a user defined covariance function, variogram model, or arbitrary function.

RMuser is very slow – users should avoid this model whenever possible.

Usage

```
RMuser(type, domain, isotropy, vdim, beta,
       variab.names = c("x", "y", "z", "T"), fctn, fst, snd, envir,
       var, scale, Aniso, proj)
```

Arguments

type	See RMmodelgenerator for the range of values of the arguments. Default: "shape function".
domain	See RMmodelgenerator for the range of values of the arguments. Default: XONLY.
isotropy	See RMmodelgenerator for the range of values of the arguments. Default: • 'isotropic' if type equals 'tail correlation function', 'positive definite' or 'negative definite'; • 'cartesian system' if type indicates a process or simulation method or a shape function.
vdim	multivariability. Default: vdim is identified from beta if given; otherwise the default value is 1.
beta	a fixed matrix that is multiplied to the return value of the given function; the dimension must match. Defining a vector valued function and beta as a vector, an arbitrary linear model can be defined. Estimation of beta is, however, not established yet.
variab.names	Just the names of the variables. More variable names might be given here than used in the function. See Detail for the interpretation of variables.
fctn, fst, snd	a user defined function and its first, second and third derivative, given as quote(myfunction(x)) or as quote(myfunction(x, y)), see Details an Examples below.
envir	the environment where the given function shall be evaluated
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Primarily, a function is expected that depends on a vector whose components, x, y, z, T, are given separately as scalar quantities.

Alternatively, the function might depend only on the first argument given by variab.names.

A kernel should depend on the first two arguments given by variab.names.

Value

RMuser returns an object of class RMmodel.

Note

- The use of RMuser is completely on the risk of the user. There is no way to check whether the expressions of the user are correct in any sense.
- Note that x, y, z and T are reserved argument names that define solely the coordinates. Hence, none of these names might be used for other arguments within these functions.
- In the user-defined functions the models of **RandomFields** are not recognised, so they cannot be included in the function definitions.
- RMuser may not be used in connection with obsolete commands of RandomFields.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RFfit, RMmodelgenerator RMmodel, RFsimulate, RC_ISOTROPY, RC_DOMAIN

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again


## Alternatively to 'model <- RMexp()' one may define the following
## (which is, however, much slower and cannot use all features of
## RandomFields)

## user-defined exponential covariance model
model <- RMuser(type="positive definite", domain="single variable",
                iso="isotropic", fctn=exp(-x))
x <- y <- seq(1, 10, len=10^(1+interactive()))
plot(model)
z <- RFsimulate(RPcirculant(model), x=x, y=y)
plot(z)

## the kernel, which is the scalar product (not programmed (yet) in
## RandomFields) %to do
model <- RMnugget(var=1e-5) +
         RMuser(type="positive definite", domain="kernel",
                iso="symmetric", fctn=sum(x * y))
x <- y <- seq(1, 10, len=7^(1+interactive()))
z <- RFsimulate(model, x=x, y=y, n=6)
plot(z)
```

RMvector *Vector Covariance Model*

Description

RMvector is a multivariate covariance model which depends on a univariate covariance model that is stationary in the first $Dspace$ coordinates h and where the covariance function phi(h,t) is twice differentiable in the first component h.

The corresponding matrix-valued covariance function C of the model only depends on the difference h between two points in the first component. It is given by

$$C(h,t) = (-0.5 * (a+1)\Delta + a\nabla\nabla^T)C_0(h,t)$$

where the operator is applied to the first component h only.

Usage

```
RMvector(phi, a, Dspace, var, scale, Aniso, proj)
```

Arguments

phi	an RMmodel; has two components h (2 or 3 dimensional and stationary) and t (arbitrary dimension)
a	a numerical value; should be in the interval $[-1, 1]$.
Dspace	an integer; either 2 or 3; the first $Dspace$ coordinates give the first component h
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

C_0 is either a spatiotemporal model (then t is the time component) or it is an isotropic model. Then, the first $Dspace$ coordinates are considered as h coordinates and the remaining ones as t coordinates. By default, $Dspace$ equals the dimension of the field (and there is no t component). If $a = -1$ then the field is curl free; if $a = 1$ then the field is divergence free.

Value

RMvector returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Scheuerer, M. and Schlather, M. (2012) Covariance Models for Divergence-Free and Curl-Free Random Vector Fields. *Stochastic Models* **28:3**.

See Also

RMcurlfree, RMdivfree, RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMvector(RMgauss(), scale=0.3)
x <- seq(0, 10, if (interactive()) 0.4 else 3)
plot(RFsimulate(model, x=x, y=x, z=0), select.variables=list(1:2))
```

RMwave *Wave Covariance Model / Cardinal Sine*

Description

RMwave is a stationary isotropic covariance model, which is valid only for dimension $d \leq 3$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = sin(r)/r 1_{r>0} + 1_{r=0}.$$

It is a special case of RMbessel.

Usage

```
RMwave(var, scale, Aniso, proj)
RMcardinalsine(var, scale, Aniso, proj)
```

Arguments

`var,scale,Aniso,proj`
: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is valid only for dimension $d \leq 3$. It is a special case of RMbessel for $\nu = 0.5$.

This covariance models a hole effect (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 92).

Value

RMwave returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.

See Also

RMbessel RMmodel, RFsimulate, RFfit.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMwave(scale=0.1)
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x))
```

RMwhittlematern *Whittle-Matern Covariance Model*

Description

RMmatern is a stationary isotropic covariance model belonging to the Matern family. The corresponding covariance function only depends on the distance $r \geq 0$ between two points.

The Whittle model is given by

$$C(r) = W_\nu(r) = 2^{1-\nu}\Gamma(\nu)^{-1} r^\nu K_\nu(r)$$

where $\nu > 0$ and K_ν is the modified Bessel function of second kind.

The Matern model is given by

$$C(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu} r)^\nu K_\nu(\sqrt{2\nu} r)$$

Usage

```
RMwhittle(nu, notinvnu, var, scale, Aniso, proj)

RMmatern(nu, notinvnu, var, scale, Aniso, proj)
```

Arguments

nu	a numerical value called "smoothness parameter"; should be greater than 0.
notinvnu	logical, if not given the model is defined as above. (default). This argument should not be set by users. See the Notes.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

RMwhittle and RMmatern are two alternative parametrizations of the same covariance function. The Matern model should be preferred as this model seperates the effects of scaling parameter and the shape parameter.

This is the normal scale mixture model of choice if the smoothness of a random field is to be parametrized: the sample paths of a Gaussian random field with this covariance structure are m times differentiable if and only if $\nu > m$ (see Gelfand et al., 2010, p. 24).

Furthermore, the fractal dimension (see also RFfractaldim) D of the Gaussian sample paths is determined by ν: we have

$$D = d + 1 - \nu, \nu \in (0, 1)$$

and $D = d$ for $\nu > 1$ where d is the dimension of the random field (see Stein, 1999, p. 32).

If $\nu = 0.5$ the Matern model equals RMexp.

For ν tending to ∞ a rescaled Gaussian model RMgauss appears as limit of the Matern model.

For generalisations see section 'seealso'.

Value

The function return an object of class RMmodel

Note

The Matern model called by $C(r\sqrt{(}2))$ equals the Handcock-Wallis (1994) parametrisation.

The model allows further to be reparameterized by substituting ν for ν^{-1} setting the argument `invnu=TRUE`. Note that the inversion of ν does not really make sense for the Whittle model. Due to this fact, if the argument `invnu` is given, the Whittle model changes its definition and becomes identical to the Matern model.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Covariance function

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty.* New York: Wiley.
- Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P. (eds.) (2010) *Handbook of Spatial Statistics.* Boca Raton: Chapman & Hall/CRL.
- Guttorp, P. and Gneiting, T. (2006) Studies in the history of probability and statistics. XLIX. On the Matern correlation family. *Biometrika* **93**, 989–995.
- Handcock, M. S. and Wallis, J. R. (1994) An approach to statistical spatio-temporal modeling of meteorological fields. *JASA* **89**, 368–378.
- Stein, M. L. (1999) *Interpolation of Spatial Data – Some Theory for Kriging.* New York: Springer.

Tail correlation function (for $\nu \in (0, 1/2]$)

- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

See Also

- RMexp, RMgauss for special cases of the model (for $\nu = 0.5$ and $\nu = \infty$, respectively)
- RMhyperbolic for a univariate generalization
- RMbiwm for a multivariate generalization
- RMnonstwm, RMstein for anisotropic (space-time) generalizations
- RMmodel, RFsimulate, RFfit for general use.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- seq(0, 1, len=if (interactive()) 100 else 3)
model <- RMwhittle(nu=1, Aniso=matrix(nc=2, c(1.5, 3, -3, 4)))
plot(model, dim=2, xlim=c(-1,1))
z <- RFsimulate(model=model, x, x)
plot(z)
```

RPbernoulli *Simulation of Binary Random Fields*

Description

Indicator or binary field which has the value 1, if an underfield field exceeds a given threshold, 0 otherwise.

Usage

```
RPbernoulli(phi, stationary_only, threshold)
```

Arguments

phi	the RMmodel. Either a model for a process or a covariance model must be specified. In the latter case, a Gaussian process RPgauss is tacitely assumed.
stationary_only	optional arguments; same meaning as for RPgauss. It is ignored if the submodel is a process definition.

threshold — real valued. RPbernoulli returns 1 if value of the random field given by phi is equal to or larger than the value of threshold, and 0 otherwise. In the multivariate case, a vector might be given. If the threshold is not finite, then the original field is returned.

threshold defaults value is 0.

Details

RPbernoulli can applied to any field. If only a covariance model is given, a Gaussian field is simulated as underlying field.

Value

The function returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

Auxiliary RMmodels, RP, RMbernoulli

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

x <- seq(0, 10, if (interactive()) 0.1 else 2)
model <- RPbernoulli(RMexp(), threshold=0)
z <- RFsimulate(model, x, x, n=4)
plot(z)

model <- RPbernoulli(RPbrownresnick(RMexp(), xi=1), threshold=1)
z <- RFsimulate(model, x, x, n=4)
plot(z)
```

RPchi2 — *Simulation of a Chi2 Random Fields*

Description

RPchi2 defines a chi2 fields.

Usage

```
RPchi2(phi, f)
```

Arguments

phi	the RMmodel. If a model for the distribution is not specified, RPgauss is used as default and a covariance model is expected.
f	integer. Degree of freedom.

Value

The function returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

Auxiliary RMmodels, RP, RPgauss

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RPchi2(RMexp(), f=2)
x <- seq(0, 10, if (interactive()) 0.01 else 2)
z <- RFsimulate(model=model, x, x, n=4)
plot(z)
```

RPgauss *Simulation of Gaussian Random Fields*

Description

This function is used to specify a Gaussian random field that is to be simulated or estimated. Returns an object of class RMmodel.

Usage

```
RPgauss(phi, stationary_only)
```

Arguments

phi	the RMmodel.
stationary_only	Logical or NA. Used for the automatic choice of methods. • TRUE: the simulation of non-stationary random fields is refused. In particular, the intrinsic embedding method is excluded and the simulation of Brownian motion is rejected. • FALSE: intrinsic embedding is always allowed, actually it's the first one considered in the automatic selection algorithm. • NA: the simulation of the Brownian motion allowed, but intrinsic embedding is not used for translation invariant ("stationary") covariance models. Default: NA

Value

The function returns an object of class RMmodel.

Note

In most cases, RPgauss need not be given explicitely as Gaussian random fields are assumed as default.

RPgauss may not find the fastest method neither the most precise one. It just finds any method among the available methods. (However it guesses what is a good choice.) See RFgetMethodNames for further information. Note that some of the methods do not work for all covariance or variogram models, see RFgetModelNames(intern=FALSE)

By default, all Gaussian random fields have zero mean. Simulating with trend can be done by including RMtrend in the model.

RPgauss allows to simulate different classes of random fields, controlled by the wrapping model:

If the submodel is a pure covariance or variogram model, i.e. of class RMmodel, a corresponding centered Gaussian field is simulated. Not only stationary fields but also non-stationary and anisotropic models can be used, e.g. zonal anisotropy, geometrical anisotropy, separable models, non-separable space-time smodels, multiplicative or nested models; see RMmodel for a list of all available models.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RP, Gaussian, RMmodel, RFoptions, RPbrownresnick, RPchi2, RPopitz. RPt, RPschlather.

Do not mix up with RMgauss or RRgauss.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMexp()
x <- seq(0, 10, if (interactive()) 0.02 else 1)
plot(model)
plot(RFsimulate(model, x=x, seed=0))
plot(RFsimulate(RPgauss(model), x=x, seed=0), col=2) ## the same
```

RPpoisson *Simulation of Random Fields*

Description

Shot noise model, which is also called moving average model, trigger process, dilution random field, and by several other names.

Usage

```
RPpoisson(phi, intensity)
```

Arguments

phi	the model, RMmodel, gives the shape function to be used
intensity	the intensity of the underlying stationary Poisson point process

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RMmodel RP, RPcoins

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

# example 1: Posson field based on disks with radius 1
x <- seq(0,25, if (interactive()) 0.02 else 5)
model <- RMball()
z <- RFsimulate(RPpoisson(model), x, intensity = 2)
plot(z)
par(mfcol=c(2,1))
plot(z@data[,1:min(length(z@data), 1000)], type="l")
hist(z@data[,1], breaks=0.5 + (-1 : max(z@data)))


# example 2: Poisson field based on the normal density function
# note that
# (i) the normal density as unbounded support that has to be truncated
# (ii) the intensity is high so that the CLT holds
x <- seq(0,10, if (interactive()) 0.01 else 1)
model <- RMtruncsupport(radius=5, RMgauss())
z <- RFsimulate(RPpoisson(model), x, intensity = 100)
plot(z)
```

RPprocess *Models for classes of random fields (RP commands)*

Description

Here, all the classes of random fields are described that can be simulated

Implemented processes

Gaussian Random fields	see Gaussian
Max-stable Random Fields	see Maxstable
Other Random Fields	Binary field
	chi2 field
	composed Poisson (shot noise, random coin)
	t field

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RC, RR, RM, RF

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- seq(0, 10, 0.1)
model <- RMexp()

## a Gaussian field with exponential covaraince function
z <- RFsimulate(model, x)
plot(z)

## a binary field obtained as a thresholded Gaussian field
b <- RFsimulate(RPbernoulli(model), x)
plot(b)

sum( abs((z@data$variabl1 >=0 ) - b@data$variable1)) == 0 ## TRUE,
## i.e. RPbernoulli is indeed a thresholded Gaussian process
```

RPt *Simulation of a T Random Fields*

Description

RPt defines a t fields.

Usage

```
RPt(phi, nu)
```

Arguments

phi	the RMmodel. If a model for the distribution is not specified, RPgauss is used as default and a covariance model is expected.
nu	non-negative number. Degree of freedom.

Value

The function returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

Related to the extremal t process

- T. Opitz (2012) A spectral construction of the extremal t process. *arxiv* **1207.2296**.

See Also

Auxiliary RMmodels, RP, RPgauss

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RPt(RMexp(), nu=2)
x <- seq(0, 10, if (interactive()) 0.1 else 2)
z <- RFsimulate(model, x, x, n=4)
plot(z)
```

RRdeterm *Random scaling used with balls*

Description

RRdeterm refers to the distribution of a deterministic variable.

Usage

```
RRdeterm(mean)
```

Arguments

mean	the deterministic value

Value

RRdeterm returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RRdistr, RRgauss,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
x <- seq(-2, 2, 0.001)
p <- RFpdistr(RRdeterm(mean=1), q=x)
plot(x, p, type="l")
```

RRdistr *RRdistr*

Description

RRdistr defines of distribution family given by fct. It is used to introduce Random parameters based on distributions defined on R.

Usage

```
RRdistr(fct, nrow, ncol, envir)
```

Arguments

fct	an arbitrary family of distribution. E.g. norm() for the family dnorm, pnorm, qnorm, rnorm.
nrow, ncol	The matrix size (or vector if ncol=1) the family returns. Except for very advanced modelling we always have nrow=ncol=1, which is the default.
envir	an environment; defaults to new.env().

Details

RRdistr returns an object of class RMmodel.

Note

RRdistr is the generic model introduced automatically when distribution families in R are used in the model definition. See the examples below.

Note

The use of RRdistr is completely on the risk of the user. There is no way to check whether the expressions of the user are statistically correct.

Further, RRdistr may not be used in connection of obsolete commands of RandomFields.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RMmodel, RR, RRdistr, RFsimulate, RRdistr

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
## here a model with random scale parameter
model <- RMgauss(scale=exp(rate=1))
x <- seq(0,10,0.02)
n <- if (interactive()) 10 else 1
for (i in 1:n) {
  readline(paste("Simulation no.", i, ": press return", sep=""))
  plot(RFsimulate(model, x=x, seed=i))
}

## another possibility to define exactly the same model above is
## model <- RMgauss(scale=exp())

## note that however, the following two definitions lead
## to covariance models with fixed scale parameter:
## model <- RMgauss(scale=exp(1))   # fixed to 2.7181
## model <- RMgauss(scale=exp(x=1)) # fixed to 2.7181


## here, just two other examples:
## fst
model <- RMmatern(nu=unif(min=0.1, max=2)) # random
for (i in 1:n) {
  readline(paste("Simulation no.", i, ": press return", sep=""))
  plot(RFsimulate(model, x=x, seed=i))
}

## snd
## note that the fist 'exp' refers to the exponential function,
## the second to the exponential distribution.
(model <- RMgauss(var=exp(3), scale=exp(rate=1)))
plot(z <- RFsimulate(model=model, x=1:100/10))
```

RRgauss *Random scaling used with balls*

Description

RRgauss defines the d-dimensional vector of independent Gaussian random variables.

Usage

```
RRgauss(mu, sd, log)
```

Arguments

mu, sd, log	see Normal. Here the components can be vectors, leading to multivariate distibution with independent components

Details

It has the same effect as `RRdistr(norm(mu=mu, sd=sd, log=log))`

Value

RRgauss returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RRdistr, RRunif.

Do not mix up RRgauss with RMgauss or RPgauss.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
r <- RFrdistr(RRgauss(mu=c(1,5)), n=1000, dim=2)
plot(r[1,], r[2, ])
```

RRloc *Random scaling used with balls*

Description

RRloc modifies location and scale of a distribution.

Usage

```
RRloc(phi, mu, scale, pow)
```

Arguments

phi	distribution RMmodel.
mu	location shift
scale	scale modification
pow	argument for internal use only

Details

It has the same effect as RRdistr(norm(mu=mu, sd=sd, log=log))

Value

RRloc returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RRdistr, RRgauss,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## empirical density of the distribution 'RRspheric'
model <- RRspheric(balldim=2)
hist(RFrdistr(model, n=1000), 50)

## empirical density of the distribution 'RRspheric', shifted by 3
model <- RRloc(mu=3, RRspheric(balldim=2))
hist(RFrdistr(model, n=1000), 50)
```

RRrectangular *Random scaling used with balls*

Description

Approximates an isotropic decreasing density function by a density function that is isotropic with respect to the l_1 norm.

Usage

```
RRrectangular(phi, safety, minsteplen, maxsteps, parts, maxit,
              innermin, outermax, mcmc_n, normed, approx, onesided)
```

Arguments

phi	a shape function; it is the user's responsability that it is non-negative. See details.
safety, minsteplen, maxsteps, parts, maxit, innermin, outermax, mcmc_n	Technical arguments to run an algorithm to simulate from this distribution. See RFoptions for the default values.
normed	logical. If FALSE then the norming constant c in the Details is set to 1. This affects the values the density function, the probability distribution and the quantile function, but not the simulation of random variables.
approx	logical. Default TRUE. If TRUE the isotropic distribution with respect to the l_1 norm is returned. If FALSE then the exact isotropic distribution with respect to the l_2 norm is simulated. Neither the density function, nor the probability distribution, not the quantile function will be availabe if approx=TRUE.
onesided	logical. Only for used for univariate distributions. if TRUE then the density is assumed to be non-negative only on the positive real axis. Otherwise the density is assumed to be symmetric.

Details

This models defines an isotropic density function \$f\$ with respect to the l_1 norm. i.e. $f(x) = c\phi(\|x\|_{l_1})$ with some function ϕ. Here, s is norming constant so that the integral of f equals one.

In case ϕ is monotoneously decreasing then rejection sampling is used, else MCMC.

The function ϕ might have a polynomial pole at the origin and asymptotically decreasing of the form $x^{\beta}exp(-x^{\delta})$.

Value

RRrectangular returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RRdistr, RRgauss,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

# simulation of Gaussian variables (in a not very straight forward way):
distr <- RRrectangular(RMgauss(), approx=FALSE)
z <- RFrdistr(distr, n=if (interactive()) 1000000 else 10)
hist(z, 200, freq=!TRUE)
x <- seq(-10, 10, 0.1)
lines(x, dnorm(x, sd=sqrt(0.5)))
```

```
#creation of random variables whose density is proportional
# to the spherical model:
distr <- RRrectangular(RMspheric(), approx=FALSE)
z <- RFrdistr(distr, n=if (interactive()) 1000000 else 10)
hist(z, 200, freq=!TRUE)
x <- seq(-10, 10, 0.01)
lines(x, 4/3 * RFcov(RMspheric(), x))
```

RRspheric *Random scaling used with balls*

Description

This model delivers the distribution of the **radius** of a ball obtained by the intersection of a balldim-dimensional ball with **diameter** R by a $spacedim$-dimensional hyperplane that has uniform distance from the center.

Usage

```
RRspheric(spacedim, balldim, R)
```

Arguments

spacedim	dimension of the hyperplane; defaults to 1.
balldim	the dimension of the ball
R	radius. Default: 1

Value

RRspheric returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RMball,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
hist(RFrdistr(RRspheric(balldim=2), n=1000), 50)
```

RRunif *Random scaling used with balls*

Description

The model refers to the d-dimensional univariate distribution on a rectangular window.

Usage

```
RRunif(min, max, normed)
```

Arguments

min,max	lower and upper corner of a rectangular window
normed	logical with default value TRUE.
	Advanced. If FALSE then the identicator function for the window is not normed to get a probability distribution. Nonetheless random drawing from the distribution still works.

Details

In the one-dimensional case it has the same effect as RRdistr(unif(min=min, max=max, log=log))

Value

RRunif returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

See Also

RMmodel, RRdistr, RRgauss, RRspheric,

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
## uniform distribution on [0,1] x [-2, -1]
RFrdistr(RRunif(c(0, -2), c(2, -1)), n=5, dim=2)
RFpdistr(RRunif(c(0, -2), c(2, -1)), q=c(1, -1.5), dim=2)
RFddistr(RRunif(c(0, -2), c(2, -1)), x=c(1, -1.5), dim=2)
```

S02 *Models for stationary max-stable random fields*

Description

Here the code of the paper on 'Models for stationary max-stable random fields' is given.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software;

References

- Schlather, M. (2002) Models for stationary max-stable random fields. *Extremes* **5**, 33-44.

Examples

```
RFoptions(seed=0, xi=1)
## seed = 0 : *ANY* simulation will have the random seed 0; set
##            RFoptions(seed=NA) to make them all random again
## xi = 0.5: Frechet margins with alpha=2

## Due to change in the handling the seeds here are different from the
## seed in the paper.


x <- seq(0, 10, length=if (interactive()) 128  else 4)

# Fig. 1-4
## Not run: \dontshow{plot(RFsimulate(RPsmith(RMgauss(s=1.5)), x, x))   # < 1 sec
plot(RFsimulate(RPsmith(RMball(s=RRspheric(2, 3,
R=3.3))), x, x)) # 30  sec
plot(RFsimulate(RPschlather(RMexp()), x, x))      #   1 sec
plot(RFsimulate(RPschlather(RMgauss()), x, x))    #  17 sec
}
## End(Not run)
```

S10 *On some covariance models based on normal scale mixtures*

Description

Here the code of the paper on 'On some covariance models based on normal scale mixtures' is given.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software;

References

- Schlather, M. (2010) On some covariance models based on normal scale mixtures. *Bernoulli*, **16**, 780-797.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

### Example 10 in Schlather (2010).
## The field below has more than 80 million points. So the simulation
## takes a while
y <- x <- seq(0, 10, len=if (interactive()) 256 else 5)
T <- c(0, 0.02, if (interactive()) 1275 else 4)
col <- c(topo.colors(300)[1:100], cm.colors(300)[c((1:50) * 2, 101:150)])
model <- RMcoxisham(mu=c(1, 1), D=matrix(nr=2, c(1, 0.5, 0.5, 1)),
                    RMwhittle(nu=1))
z <- RFsimulate(model, x, y, T=T, sp_lines=1500, every=10)
plot(z, MARGIN.slices=3, col=col)
```

Description

Here the code of the paper on 'On some covariance models based on normal scale mixtures' is given.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software;

References

- Strokorb, K., Ballani, F. and Schlather, M. (2014) Systematic co-occurrence of tail correlation functions among max-stable processes. Work in progress.

Examples

`Smith` *(Mixed) Moving Maxima*

Description

`RPsmith` defines a moving maximum process or a mixed moving maximum process with finite number of shape functions.

Usage

```
RPsmith(shape, tcf, xi, mu, s)
```

Arguments

`shape`	an `RMmodel` giving the spectral function
`tcf`	an `RMmodel` specifying the extremal correlation function; either `shape` or `tcf` must be given. If `tcf` is given a shape function is tried to be constructed via the `RMm2r` construction of deterministic, monotone functions.
`xi,mu,s`	the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.

Details

The argument `xi` is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical value or given a `RMmodel`, in particular by a `RMtrend` model. The default values of mu and s are 1 and ξ, respectively.

It simulates max-stable processes Z that are referred to as "Smith model".

$$Z(x) = \max_{i=1}^{\infty} X_i Y_i(x - W_i),$$

where (W_i, X_i) are the points of a Poisson point process on $\mathsf{R}^d \times (0, \infty)$ with intensity $dw * c/x^2 dx$ and $Y_i \sim Y$ are iid measurable random functions with $E[\int \max(0, Y(x))dx] < \infty$. The constant c is chosen such that Z has standard Frechet margins.

Note

IMPORTANT: for consistency reasons with the geostatistical definitions in this package the scale argument differs froms the original definition of the Smith model! See the example below.

`RPsmith` depends on `RRrectangular` and its arguments.

Advanced options are `maxpoints` and `max_gauss`, see `RFoptions`.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Haan, L. (1984) A spectral representation for max-stable processes. *Ann. Probab.*, **12**, 1194-1204.
- Smith, R.L. (1990) Max-stable processes and spatial extremes Unpublished Manuscript.

See Also

Advanced RMmodels, Auxiliary RMmodels, RMmodel, RPbernoulli, RPgauss, maxstable maxstableAdvanced

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

model <- RMball()
x <- if (interactive()) seq(0, 1000, 0.02) else seq(0, 100, 10)
z <- RFsimulate(RPsmith(model, xi=0), x)
plot(z)
hist(z@data$variable1, 50, freq=FALSE)
curve(exp(-x) * exp(-exp(-x)), from=-3, to=8, add=TRUE)

## 2-dim
x <- seq(0, 10, if (interactive()) 0.05 else 1)
z <- RFsimulate(RPsmith(model, xi=0), x, x)
plot(z)

## original Smith model
x <- seq(0, 10, if (interactive()) 0.05 else 1)
model <- RMgauss(scale = sqrt(2)) # !! cf. definition of RMgauss
z <- RFsimulate(RPsmith(model, xi=0), x, x)
plot(z)


## for some more sophisticated models see 'maxstableAdvanced'
```

---

soil *Soil data of North Bavaria, Germany*

Description

Soil physical and chemical data collected on a field in the Weissenstaedter Becken, Germany

Usage

```
data(soil)
```

Format

This data frame contains the following columns:

x.coord x coordinates given in cm

y.coord y coordinates given in cm

nr number of the samples, which were taken in this order

moisture moisture content [Kg/Kg * 100%]

NO3.N nitrate nitrogen [mg/Kg]

Total.N total nitrogen [mg/Kg]

NH4.N ammonium nitrogen [mg/Kg]

DOC dissolved organic carbon [mg/Kg]

N20N nitrous oxide [mg/Kg dried substance]

Details

For technical reasons some of the data were obtained as differences of two measurements (which are not available anymore). Therefore, some of the data have negative values.

Source

The data were collected by Wolfgang Falk, Soil Physics Group, http://www.geo.uni-bayreuth.de/bodenphysik/Welcome.html, University of Bayreuth, Germany.

References

Falk, W. (2000) *Kleinskalige raeumliche Variabilitaet von Lachgas und bodenchemischen Parameters [Small Scale Spatial Variability of Nitrous Oxide and Pedo-Chemical Parameters]*, Master thesis, University of Bayreuth, Germany.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again



################################################################
## ##
## a geostatistical analysis that demonstrates ##
## features of the package `RandomFields' ##
## ##
################################################################
```

```
data(soil)
str(soil)
soil <- RFspatialPointsDataFrame(
 coords = soil[, c("x.coord", "y.coord")],
 data = soil[, c("moisture", "NO3.N", "Total.N",
 "NH4.N", "DOC", "N20N")],
 RFparams=list(vdim=6, n=1)
 )
data <- soil["moisture"]



## plot the data first
colour <- rainbow(100)
plot(soil["moisture"], col=colour)


## fit by eye
RFgui.model <-  RFgui(soil["moisture"])


## fit by ML
model <- ~1 + RMplus(RMwhittle(scale=NA, var=NA, nu=NA), RMnugget(var=NA))
fit <- RFfit(model, data=data)
plot(fit, fit.method=c("ml", "plain", "sqrt.nr", "sd.inv"),
     model = RFgui.model, col=1:8)

## Kriging ...
x <- seq(min(data@coords[, 1]), max(data@coords[, 1]),
         l=if (interactive()) 121 else 4)
k <- RFinterpolate(fit["ml"], x=x, y=x, data=data)
plot(x=k, col=colour)
plot(x=k, y=data, col=colour)


## what is the probability that at no point of the
## grid given by x and y the moisture is greater than 24 percent?

cs <- RFsimulate(model=fit["ml"], x=x, y=x, data=data,
      n=if (interactive()) 50 else 3)
plot(cs, col=colour)
plot(cs, y=data, col=colour)
Print(mean(apply(as.matrix(cs@data) <= 24, 2, all))) ## about 40 percent
##...
```

---

sp2RF *Transformation of an 'sp' object to an 'RFsp' object*

Description

The function transforms an 'sp' object to an 'RFsp' object.

This explicite transformation is only necessary if several variables are repeated measurements are given.

Usage

```
sp2RF(sp, param=list(n=1, vdim=1))
```

Arguments

sp	an 'sp' object
param	n number of repetitions; vdim the number of variables (multivariability)

Value

sp2RF returns an object of class RFsp.

Note

The two options varnames and coordnames, cf. Section 'coords' in RFoptions, might be useful.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

See Also

RFsp

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

p <- 100
n <- 5
x <- runif(p, 0, 1)
y <- runif(p, 0, 1)
z <- RFsimulate(RMexp(), x=x, y=y, n=n)
z1 <- z2 <- as.data.frame(z)
coordinates(z2) <- ~coords.x1 + coords.x2

emp.var <- RFempiricalvariogram(data=z)
emp.var1 <- RFempiricalvariogram(data=z1)
emp.var2 <- RFempiricalvariogram(data=sp2RF(z2, param=list(n=n, vdim=1)))

## results are all equal, except for the calls:
emp.var@call <- emp.var1@call <- emp.var2@call <- NULL
```

```
stopifnot(all.equal(emp.var, emp.var1))
stopifnot(all.equal(emp.var, emp.var2))
```

`Specific` *Methods that are specific to certain covariance models*

Description

This model determines that the (Gaussian) random field should be modelled by a particular method that is specific to the given covariance model.

Usage

```
RPspecific(phi)
```

Arguments

`phi`	object of class `RMmodel`; specifies the covariance model to be simulated.

Details

`RPspecific` is used for specific algorithms or specific features for simulating certain covariance functions

- `RMplus` is able to simulate separately the fields given by its summands. This is necessary, e.g., when a `RMtrend` is involved.
- `RMmult` for Gaussian random fields only. `RMmult` simulates the random fields of all the components and multiplies them. This is repeated several times and averaged.
- `RMS` Then, for instance, `sqrt(var)` is multiplied onto the (Gaussian) random fields after the field has been simulated. Hence, when `var` is random, then, for each realisation of the Gaussian field (for `n>1` in `RFsimulate`) a new realisation of `var` is used.

  Further, new coordinates are created where the old coordinates have been devided by the `scale` and/or multiplied with the `Aniso` matrix or a `projection` has been performed.

  `RPspecific(RMS())` is called internally when the user wants to simulate `Anisotropic` fields with isotropic methods, e.g. `RPtbm`.
- `RMmppplus`
- `RMtrend`

Note that `RPspecific` applies only to the first model or operator in argument `phi`.

Value

`RPspecific` returns an object of class `RMmodel`

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

See Also

RP, RPcoins, RPhyperplane, RPspectral, RPtbm.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
## example for implicite use
model <- RMgauss(var=10, s=10) + RMnugget(var=0.1)
plot(model)
plot(RFsimulate(model=model, 0:10, 0:10, n=4))
## The following function shows the internal structure of the model.
## In particular, it can be seen that RPspecific is applied to RMplus.
RFgetModelInfo(level=0, which="internal")

## example for explicite used
model <- RPspecific(RMS(var=unif(min=0, max=10), RMgauss()))
n <- if (interactive()) 6 else 1
k <- if (interactive()) 10 else 1
x <- seq(0,10,0.02)
for (i in 1:k) {
  readline(paste("Simulation no.", i, ": press return", sep=""))
  plot(RFsimulate(model, x=x, n=n, seed=i), ylim=c(-5,5))
}
```

Spectral *Spectral turning bands method*

Description

The spectral turning bands method is a simulation method for stationary Gaussian random fields (Mantoglou and Wilson, 1982). It makes use of Bochners's theorem and the corresponding spectral measure Ξ for a given covariance function $C(h)$. For $x \in \mathbf{R}^d$, the field

$$Y(x) = \sqrt{2} cos(< V, x > +2\pi U)$$

with $V \tilde{} \Xi$ and $U \tilde{} U f o((O,1))$ is a random field with covariance function $C(h)$. A scaled superposition of many independent realizations of Y gives a Gaussian field, acoording to the central limit theorem. For details see Lantuejoul (2002). The standard method allows for the simulation of 2-dimensional random fields defined on arbitrary points or arbitrary grids.

Usage

```
RPspectral(phi, sp_lines, sp_grid, prop_factor, sigma)
```

Arguments

phi	object of class RMmodel; specifies the covariance model to be simulated.
sp_lines	Number of lines used (in total for all additive components of the covariance function). Default: 2500.
sp_grid	Logical. The angle of the lines is random if grid=FALSE, and $k\pi/$sp_lines for k in 1:sp_lines, otherwise. This argument is only considered if the spectral measure, not the density is used. Default: TRUE.
prop_factor	positive real value. Sometimes, the spectral density must be samples by MCMC. Let p the average rejection rate. Then the chain is sampled every nth point where $n = \|log(p)\| *$prop_factor Default: 50.
sigma	real. Considered if the Metropolis algorithm is used. It gives the standard deviation of the multivariate normal distribution of the proposing distribution. If sigma is not positive thenRandomFields tries to find a good choice for sigma itself. Default: 0.

Details

old args:

- ergodic In case of an additive model and ergodic=FALSE, the additive component are chosen proportional to their variance. In total sp_lines are simulated. If ergodic=TRUE, the components are simulated separately and then added.
  Default: FALSE.
- identical Default: 1e-25.

Value

RPspectral returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Lantuejoul, C. (2002) *Geostatistical Simulation: Models and Algorithms.* Springer.
- Mantoglou, A. and J. L. Wilson (1982), *The Turning Bands Method for simulation of random fields using line generation by a spectral method.* Water Resour. Res., 18(5), 1379-1394.

See Also

RPcoins, RPhyperplane, RPspectral, RPtbm.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RPspectral(RMmatern(nu=1))
y <- x <- seq(0,10,len=if (interactive()) 400 else 3)
z <- RFsimulate(model, x, y, n=2)
plot(z)
```

Square roots *Methods relying on square roots of the covariance matrix*

Description

Methods relying on square roots of the covariance matrix

Usage

```
RPdirect(phi, root_method, svdtolerance, max_variab)

RPsequential(phi, max_variables, back_steps, initial)
```

Arguments

phi	object of class RMmodel; specifies the covariance model to be simulated.
root_method	Decomposition of the covariance matrix. If root_method=1 or 3, Cholesky decomposition will not be attempted, but singular value decomposition performed instead. In case of a multivariate random field, root_method = 2 or 3 orders the covariance such that first all components are considered for the first variable, then all components for the second one, and so on. If root_method = 0 or 1 it starts with the first component of all locations, then the second components follow, etc. Default: 0 .
svdtolerance	If SVD decomposition is used for calculating the square root of the covariance matrix then the absolute componentwise difference between the covariance matrix and square of the square root must be less than svdtolerance. No check is performed if svdtolerance is negative. Default: 1e-12 .

max_variab	If the number of variables to generate is greater than maxvariables, then any matrix decomposition method is rejected. It is important that this option is set conveniently to avoid great losses of time during the automatic search of a simulation method (method="any method"). Default: 8192
max_variables	The maximum size of the conditional covariance matrix (default to 5000)
back_steps	Number of previous instances on which the algorithm should condition. If less than one then the number of previous instances equals max / (number of spatial points). Default: 5 .
initial	First, N=(number of spatial points) * back_steps number of points are simulated. Then, sequentially, all spatial points for the next time instance are simulated at once, based on the previous back_steps instances. The distribution of the first N points is the correct distribution, but differs, in general, from the distribution of the sequentially simulated variables. We prefer here to have the same distribution all over (although only approximatively the correct one), hence do some initial sequential steps first. If initial is non-negative, then initial first steps are performed. If initial is negative, then back_steps - initial initial steps are performed. The latter ensures that none of the very first N variables are returned. Default: -10 .

Details

RPdirect is based on the well-known method for simulating any multivariate Gaussian distribution, using the square root of the covariance matrix. The method is pretty slow and limited to about 8000 points, i.e. a 20x20x20 grid in three dimensions. This implementation can use the Cholesky decomposition and the singular value decomposition. It allows for arbitrary points and arbitrary grids.

RPsequential is programmed for spatio-temporal models where the field is modelled sequentially in the time direction conditioned on the previous k instances. For $k = 5$ the method has its limits for about 1000 spatial points. It is an approximative method. The larger k the better. It also works for certain grids where the last dimension should contain the highest number of grid points.

Value

RPsequential returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

- Schlather, M. (1999) *An introduction to positive definite functions and to unconditional simulation of random fields.* Technical report ST 99-10, Dept. of Maths and Statistics, Lancaster University.

See Also

RP, RPcoins, RPhyperplane, RPspectral, RPtbm.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMgauss(var=10, s=10) + RMnugget(var=0.01)
plot(model, xlim=c(-25, 25))

z <- RFsimulate(model=RPdirect(model), 0:10, 0:10, n=4)
plot(z)
```

SS12 *Covariance Models for Random Vector Fields*

Description

Here the code of the paper on 'Covariance Models for Random Vector Fields' is given.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software;

References

- Scheuerer, M. and Schlather, M. (2012) Covariance Models for Random Vector Fields. *Stochastic Models*, **82**, 433-451.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
```

Strokorb's Functions *Tail correlation function of the Brown-Resnick process*

Description

The models define various shape functions for max-stable processes for a given tail correlation function

Usage

```
RMm2r(phi)
RMm3b(phi)
RMmps(phi)
```

Arguments

phi	a model for a tail correlation function belonging to the Gneiting class H_d

Details

RMm2r used with RPsmith defines a monotone shape function that corresponds to a tail correlation function belonging to Gneiting's class H_d. Currently, the function is implemented for dimensions 1 and 3. Called as such it returns the corresponding monotone function.

RMm3b used with RPsmith defines balls with random *radius* that corresponds to a tail correlation function belonging to Gneiting's class H_d. Currently, the function is implemented for dimensions 1 and 3. (Note that in Strokorb et al. (2014) the density function for twice the radius is considered.) Called as such it returns the corresponding density function for the radius of the balls.

RMmps used with RPsmith defines random hyperplane polygons that corresponds a tail correlaton function belonging to Gneiting's class H_d. It currently only allows for RMbrownresnick(RMfbm(alpha=1)) and dimension 2. Called as such it returns the tcf defined by the submodel – this definition may change in future.

Value

object of class RMmodel

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

References

- Strokorb, K. (2013) *Properties of the Extremal Coefficient Functions.* Univ. Goettingen. PhD thesis.
- Strokorb, K., Ballani, F. and Schlather, M. (2014) In Preparation.

See Also

RFsimulate, RMmodel.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again
model <- RMbrownresnick(RMfbm(alpha=1.5, s=0.2))
plot(RMm2r(model))

x <- seq(0, 10, if (interactive()) 0.005 else 1)
z <- RFsimulate(RPsmith(RMm2r(model), xi=0), x)
plot(z, type="p", pch=20)
```

Tbm *Turning Bands method*

Description

The Turning Bands method is a simulation method for stationary, isotropic random fields in any dimension and defined on arbitrary points or arbitrary grids. It performs a multidimensional simulation by superposing lower-dimensional fields. In fact, the Turning Bands method is called with the Turning Bands model, see RMtbm.
For details see RMtbm.

Usage

```
RPtbm(phi, fulldim, reduceddim, layers, lines,
      linessimufactor, linesimustep, center, points)
```

Arguments

phi	object of class RMmodel; specifies the covariance function to be simulated; a univariate stationary isotropic covariance model (see RFgetModelNames(type="positive definite", which is valid in dimension fulldim.
fulldim	a positive integer. The dimension of the space of the random field to be simulated
reduceddim	a positive integer; less than fulldim. The dimension of the auxiliary hyperplane (most frequently a line, i.e. reduceddim=1 used in the simulation.
layers	a boolean value; for space time model. If TRUE then the turning layers are used whenever a time component is given. If NA the turning layers are used only when the traditional TBM is not applicable. If FALSE then turning layers may never be used. Default: TRUE.
lines	Number of lines used. Default: 60.

`linessimufactor`

`linessimufactor` or `linesimustep` must be non-negative; if `linesimustep` is positive then `linesimufactor` is ignored. If both arguments are naught then `points` is used (and must be positive). The grid on the line is `linesimufactor`-times finer than the smallest distance. See also `linesimustep`.

Default: `2.0`.

`linesimustep` If `linesimustep` is positive the grid on the line has lag `linesimustep`. See also `linesimufactor`.

Default: `0.0`.

`center` Scalar or vector. If not `NA`, the `center` is used as the center of the turning bands for `fulldim`. Otherwise the center is determined automatically such that the line length is minimal. See also `points` and the examples below.

Default: `NA`.

`points` integer. If greater than 0, `points` gives the number of points simulated on the TBM line, hence must be greater than the minimal number of points given by the size of the simulated field and the two paramters `linesimufactor` and `linesimustep`. If `points` is not positive the number of points is determined automatically. The use of `center` and `points` is highlighted in an example below.

Default: `0`.

Details

- 2-dimensional case
  It is generally difficult to use the turning bands method (`RPtbm`) directly in the 2-dimensional space. Instead, 2-dimensional random fields are frequently obtained by simulating a 3-dimensional random field (using `RPtbm`) and taking a 2-dimensional cross-section. See also the arguments `fulldim` and `reduceddim`.
- 4-dimensional case
  The turning layers can be used for the simulations with a (formal) time component. It works for all isotropic models, some special models such as `RMnsst`, and multiplicate models with that separate the time component.

Value

`RPtbm` returns an object of class `RMmodel`

Note

Both the precision and the simulation time depend heavily on `linesimustep` and `linesimufactor`. For covariance models with larger values of the scale parameter, `linesimufactor=2` is too small.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>

References

Turning bands

- Lantuejoul, C. (2002) *Geostatistical Simulation: Models and Algorithms.* Springer.
- Matheron, G. (1973). The intrinsic random functions and their applications. *Adv. Appl. Probab.*, **5**, 439-468.
- Strokorb, K., Ballani, F., and Schlather, M. (2014) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, Submitted.

Turning layers

- Schlather, M. (2011) Construction of covariance functions and unconditional simulation of random fields. In Porcu, E., Montero, J.M. and Schlather, M., *Space-Time Processes and Challenges Related to Environmental Problems.* New York: Springer.

See Also

RMtbm, RP, RPspectral

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                   RFoptions(seed=NA) to make them all random again

## isotropic example that forces the use of the turning bands method
model <- RPtbm(RMstable(s=1, alpha=1.8))
x <- seq(-3, 3, 0.1)
z <- RFsimulate(model=model, x=x, y=x)
plot(z)

## anisotropic example that forces the use of the turning bands method
model <- RPtbm(RMexp(Aniso=matrix(nc=2, rep(1,4))))
z <- RFsimulate(model=model, x=x, y=x)
plot(z)

## isotropic example that uses the turning layers method
model <- RMgneiting(orig=FALSE, scale=0.4)
x <- seq(0,10, if (interactive()) 0.1 else 5)
z <- RFsimulate(model, x=x, y=x, z=x, T=c(1,1,5))
plot(z, MARGIN.slices=4, MARGIN.movie=3)
```

`weather` *Pressure and temperature forecast errors over the Pacific Northwest*

Description

Meteorological dataset, which consists of difference between forecasts and observations (forcasts minus observations) of temperature and pressure at 157 locations in the North American Pacific Northwest.

Usage

```
data(weather)
```

Format

The data frame `weather` contains the following columns:

pressure in units of Pascal

temperature in units of degree Celcius

lon longitudinal coordinates of the locations

lat latitude coordinates of the locations

Furthermore, some results obtained from the data analysis in `jss14` are delivered that are `pars.model`, `pars`, `whole.model`, `whole`.

Finally, the variable `information` contains packing information (the date and the version of **RandomFields**)

Details

The forecasts are from the GFS member of the University of Washington regional numerical weather prediction ensemble (UWME; Grimit and Mass 2002; Eckel and Mass 2005); they were valid on December 18, 2003 at 4 pm local time, at a forecast horizon of 48 hours.

Source

The data were obtained from Cliff Mass and Jeff Baars in the University of Washington Department of Atmospheric Sciences.

References

- Eckel, A. F. and Mass, C. F. (2005) Aspects of effective mesoscale, short-range ensemble forecasting *Wea. Forecasting* **20**, 328-350.
- Gneiting, T., Kleiber, W. and Schlather, M. (2010) Matern cross-covariance functions for multivariate random fields *J. Amer. Statist. Assoc.* **105**, 1167-1177.
- Grimit, E. P. and Mass, C. F. (2002) Initial results of a mesoscale short-range forecasting system over the Pacific Northwest *Wea. Forecasting* **17**, 192-205.

See Also

A reanalysis has been performed in Section 5 of the jss14 paper

Examples

```
## see 'jss14'
```

Index