

Package ‘RandomFields’

February 6, 2020

Version 3.3.8

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], Olga Moreva [aut], Jonas Auel[ctr], Peter Menck [ctr], Sebastian Gross [ctr], Ulrike Ober [ctb], Paulo Ribeiro [ctb], Brian D. Ripley [ctb], Richard Singleton [ctb], Ben Pfaff [ctb], R Core Team [ctb]

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

LinkingTo RandomFieldsUtils

Depends R (>= 3.5.0), sp, RandomFieldsUtils (>= 0.5.1)

Imports graphics, methods, grDevices, stats, utils

Suggests colorspace, RColorBrewer, mvtnorm, raster, tcltk2, tcltk, tkrplot, spam, tools, minqa, soma, optimx, nloptr, pso, GenSA

Description Methods for the inference on and the simulation of Gaussian fields are provided, as well as methods for the simulation of extreme value random fields. Main geostatistical parts are based on the books by Christian Lantuejoul <doi:10.1007/978-3-662-04808-5>, Jean-Paul Chiles and Pierre Delfiner <doi:10.1002/9781118136188> and Noel A. Cressie <doi:10.1002/9781119115151>. For extreme value random fields see Oesting, Schlather, Schillings (2019) <doi.org/10.1002/sta4.228> and Schlather (2002) <doi.org/10.1023/A:1020977924878>.

Collate aaa_auto.R aaa_globals.R basic.fctns.R bigdata.R Class.R convert_new.R convert.R Crossvalidation.R datasets.R D.H.R fitbernoulli.R fitbr.R fitgauss.R fitmaxstable.R fitpoisson.R getNset.R helmholz.R internal_use.R kriging.R Likelihoodratio.R Methods-aux.R Methods-RFsp.R obsolete.R RFempvario-Methods-plots.R RFempvario.R RFfit.R RFgui.R rf-plots.R rf.R RMmodels-Methods-plots.R RMmodelsBasics.R RMmodels.R RMmodelsConvenience.R RMmodelsSpecial.R RQmodels.R generatemodels.R zzz.R

License GPL (>= 3)

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-02-06 06:20:37 UTC

R topics documented:

RandomFields-package	6
Brown-Resnick-Specific	10
BrownResnick	12
ca20	14
Changings	15
Circulant Embedding	16
Coins	20
Constants	21
conventional2RFspDataFrame	22
Coordinate systems	24
Distribution Families	27
Extremal t	29
ExtremalGaussian	30
fitgauss	32
GaussianFields	33
GSPSJ06	35
Hierarchical Modelling	36
Hyperplane	38
Independent Variables	39
Internal functions	41
jss14	43
Major Revisions	47
Mathematial C functions	49
Max-stable random fields	54
Max-stable random fields, advanced	56
Obsolete Functions Version 2	56
Obsolete Functions Version 3	58
Others	59
papers	60
plot-method	61
PrintModelList	67
RFboxcox	68
RFcov	70
RFcovmatrix	73
RFcrossvalidate	74
RFdistr	77
RFempVariog-class	78
RFfctn	81
RFfit	82
RFfit-class	85
RFfitoptimiser	89
RFformula	91

RFformulaAdvanced	93
RFfractaldim	95
RFfunction	98
RFgetMethodNames	100
RFgetModel	104
RFgetModelInfo	105
RFgetModelNames	107
RFgridDataFrame-class	111
RFgui	113
RFhurst	114
RFinterpolate	117
RFlinearpart	121
RFloglikelihood	123
RFmadogram	126
RFoldstyle	129
RFoptions	130
RFoptionsAdvanced	151
RFpar	153
RFpointsDataFrame-class	154
RFpseudomadogram	156
RFpseudovariogram	158
RFratiotest	160
RFsimulate	163
RFsimulate.more.examples	167
RFsimulate.sophisticated.examples	168
RFsimulateAdvanced	168
RFsp-class	173
RFspatialGridDataFrame-class	175
RFspatialPointsDataFrame-class	178
RFvariogram	180
RMangle	183
RMaskey	184
RMave	186
RMball	188
RMbcw	189
RMbernoulli	190
RMbessel	191
RMbicauchy	193
RMbigneiting	194
RMbistable	196
RMbiwm	198
RMblend	200
RMbr2bg	201
RMbr2eg	202
RMbrownresnick	204
RMbubble	205
RMcauchy	208
RMcauchytbm	210

RMchoquet	211
RMcircular	212
RMconstant	213
RMcov	214
RMcovariate	216
RMcoxisham	217
RMcubic	219
RMcurlfree	220
RMcutoff	221
RMdagum	223
RMdampedcos	224
RMdeclare	225
RMdelay	227
RMderiv	228
RMdewijsian	229
RMdivfree	231
RMeaxxa	232
RMepscauchy	233
RMexp	235
RMexponential	236
RMfbm	238
RMfixcov	239
RMfixed	241
RMflatpower	241
RMfractdiff	243
RMfractgauss	244
RMgauss	245
RMgencauchy	246
RMgenfbm	248
RMgengneiting	249
RMgennsst	251
RMgneiting	252
RMgneitingdiff	253
RMhyperbolic	255
RMiaco	256
RMid	257
RMidmodel	258
RMintern	259
RMintexp	260
RMintrinsic	261
RMkolmogorov	262
RMlgd	263
RMlsfbm	265
RMma	266
RMmastein	267
RMmatrix	269
RMmodel	271
RMmodel-class	273

RMmodelFit-class	275
RMmodelgenerator-class	277
RMmodels Overview	280
RMmodelsAdvanced	281
RMmodelsMultivariate	284
RMmodelsNonstationary	286
RMmodelsSpacetime	287
RMmppplus	288
RMmqam	289
RMmult	291
RMmultiquad	292
RMnatsc	293
RMnonstwm	295
RMnsst	296
RMnugget	297
RMparswm	298
RMpenta	300
RMplus	301
RMpolygon	302
RMpolynome	303
RMpower	304
RMprod	305
RMqam	306
RMqexp	308
RMrational	309
RMrotat	310
RMS	311
RMSadvanced	312
RMScale	313
RMSchlather	315
RMSchur	316
RMSign	317
RMSinepower	318
RMSpheric	319
RMstable	321
RMstein	322
RMstp	323
RMsum	325
RMtbm	326
RMtrafo	328
RMtrend	330
RMtruncsupport	332
RMuser	333
RMvector	335
RMwave	336
RMwhittlematern	337
RPbernoulli	340
RPchi2	341

RPgauss	342
RPpoisson	343
RPprocess	344
RPt	345
RRdeterm	347
RRdistr	347
RRgauss	349
RRloc	350
RRmcmc	351
RRrectangular	352
RRspheric	354
RRunif	355
S02	356
S10	356
SBS14	357
Sequential	359
Smith	360
soil	362
sp2RF	364
Specific	365
Spectral	367
Spherical models	368
Square root	370
SS12	371
Stokorb's Functions	372
Tail Correlation Functions	373
Tbm	375
Trend Modelling	378
weather	379

Index	381
--------------	------------

RandomFields-package *Simulation and Analysis of Random Fields*

Description

The package RandomFields offers various tools for

1. **model estimation (ML) and inference (tests)** for regionalized variables and data analysis,
2. **simulation** of different kinds of random fields, including
 - multivariate, spatial, spatio-temporal, and non-stationary 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).

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

Details

The following features are provided by the package:

1. Bayesian Modelling

- See [Bayesian Modelling](#) for an introduction to hierarchical modelling.

2. Coordinate systems

- Cartesian, earth and spherical coordinates are recognized, see [coordinate systems](#) for details.
- A list of valid models is given by [spherical models](#).

3. 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)

4. 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
- [RFvariogram](#) : calculates the empirical variogram
- [RFcov](#) : calculates the empirical (auto-)covariance function

5. Graphics

- Fitting a covariance function manually [RFgui](#)
- the generic function [plot](#)
- global graphical parameters with [RFpar](#)

6. Inference (for Gaussian random fields)

- [RFCrossvalidate](#) : cross validation
- [RFlikelihood](#) : likelihood
- [RFratiotest](#) : likelihood ratio test
- [AIC](#), [AICc](#), [BIC](#), [anova](#), [logLik](#)

7. Models

- For an introduction and general properties, see [RMmodels](#).
- For an overview over classes of covariance and variogram models –e.g. for **geostatistical** purposes– see [RM](#). More sophisticated models and covariance function operators are included.
- [RFformula](#) reports a new style of passing a model since version 3.3.
- definite models are evaluated by [RFcov](#), [RFvariogram](#) and [RFcovmatrix](#). For a quick impression use [plot\(model\)](#).
- non-definite models are evaluated by [RFfctn](#) and [RFcalc](#)
- [RFlinearpart](#) returns the linear part of a model
- [RFboxcox](#) deals explicitly with Box-Cox transformations. In many cases it is performed implicitly.

8. Prediction (for second-order random fields)

- `RFinterpolate` : kriging, including imputing

9. Simulation

- `RFsimulate`: Simulation of random fields, including conditional simulation. For a list of all covariance functions and variogram models see `RM`. Use `plot` for visualisation of the result.

10. 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 option 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 explicit transformation of `sp` objects to S4 objects of `RandomFields`.
- Further generic functions are available for fitted models, see ‘Inference’ above.

11. Xtended features, especially for 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.

`RandomFields` should be rather stable when running it with `parallel`. However `RandomFields` might crash severely if an error occurs when running in parallel. When used with `parallel`, you might set `RFoptions(cores = 1)`. Note that `RFoptions(cores = ...)` with more than 1 core uses another level of parallelism which will be in competetions with `parallel` during runtime.

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; RFvariogram, 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; RFvariogram, 2011-12)

Ulrike Ober (Univ. Goettingen; help pages, testing, 2011-12)
 Marco Oesting (Univ. Mannheim; Brown-Resnick processes, Kriging, Trend, 2011-13)
 Paulo Ribeiro (Universidade Federal do Parana; code adopted from **geoR**, 2014)
 Kirstin Storkorb (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 Ribeiro : 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 choices for the optimizer (instead of `optim`) 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>

References

- Singleton, R.C. (1979). In *Programs for Digital Signal Processing* Ed.: Digital Signal Processing Committee and IEEE Acoustics, Speech, and Signal Processing Committee (1979) IEEE press.

- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Strokorb, K. (2015) Analysis, simulation and prediction of multivariate random fields with package **RandomFields**. *Journal of Statistical Software*, **63** (8), 1-25, url = 'http://www.jstatsoft.org/v63/i08/'
- see also the corresponding [vignette](#).

See Also

See also [RF](#), [RM](#), [RP](#), [RR](#), [RC](#), [R](#).

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, 0.1)
z <- RFsimulate(model, x, x, n=6)

## select some data from the simulated data
xy <- coordinates(z)
pts <- sample(nrow(xy), min(100, nrow(xy) / 2))
dta <- matrix(nrow=nrow(xy), as.vector(z))[pts, ]
dta <- cbind(xy[pts, ], dta)
plot(z, dta)

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

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

Brown-Resnick-Specific

Simulation methods for Brown-Resnick processes

Description

These models define particular ways to simulate Brown-Resnick processes.

Usage

```

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

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

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

RPloggaussnormed(variogram, prob, optimize_p, nth, burn.in, rejection)

```

Arguments

<code>phi, variogram</code>	object of class RMmodel ; specifies the covariance model to be simulated.
<code>tcf</code>	the extremal correlation function; either <code>phi</code> or <code>tcf</code> must be given.
<code>xi, mu, s</code>	the shape parameter, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details .
<code>lambda</code>	positive constant factor in the intensity of the Poisson point process used in the M3 representation, cf. Thm. 6 and Remark 7 in Oesting et. al (2012); can be estimated by setting <code>optim_mixed</code> if unknown. Default value is 1.
<code>areamat</code>	vector of values in $[0, 1]$. The value of the k th component represents the portion of processes whose maximum is located at a distance d with $k - 1 \leq d < k$ from the origin taken into account for the simulation of the shape function in the M3 representation. <code>areamat</code> can be used for isotropic models only; can be optimized by setting <code>optim_mixed</code> if unknown. Default value is 1.
<code>meshsize, vertnumber, optim_mixed, optim_mixed_tol, variobound</code>	further arguments for simulation via the mixed moving maxima (M3) representation; see RFOptions .
<code>prob</code>	to do
<code>optimize_p</code>	to do
<code>nth</code>	to do
<code>burn.in</code>	to do
<code>rejection</code>	to do

Details

The argument `xi` is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical values or given an [RMmodel](#), in particular by an [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 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@mathematik.uni-siegen.de>, <https://www.uni-siegen.de/fb6/src/scheffler/mitarbeiter/oesting>; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>

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

## currently does not work
```

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 argument ξ is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical values or (in future!) be given by an [RMmodel](#), in particular by an [RMtrend](#) model.

For $\xi = 0$, the default values of μ and s are 0 and 1, respectively. For $\xi \neq 0$, the default values of μ and s are 1 and $|\xi|$, respectively, so that it defaults to the standard Frechet case if $\xi > 0$.

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 phi.

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@mathematik.uni-siegen.de>, <https://www.uni-siegen.de/fb6/src/scheffler/mitarbeiter/oesting>; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>

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 'maxstableAdvanced'
```

ca20

Calcium content in soil samples

Description

This data set contains the calcium content measured in soil samples taken from the 0-20cm layer at 178 locations within a certain study area divided in three sub-areas. The elevation at each location was also recorded.

The first region is typically flooded during the rain season and not used as an experimental area. The calcium levels would represent the natural content in the region. The second region has received fertilisers a while ago and is typically occupied by rice fields. The third region has received fertilisers recently and is frequently used as an experimental area.

Usage

```
data(ca20)
```

Format

The object `ca20` belongs to the class `geodata` and is a list with the following elements:

`coords` a matrix with the coordinates of the soil samples.

`data` calcium content measured in $mmol_c/dm^3$.

`covariate` a data-frame with the covariates

`altitude` a vector with the elevation of each sampling location, in meters (m).

`area` a factor indicating the sub area to which the locations belongs.

`borders` a matrix with the coordinates defining the borders of the area.

reg1 a matrix with the coordinates of the limits of the sub-area 1.

reg1 a matrix with the coordinates of the limits of the sub-area 2.

reg1 a matrix with the coordinates of the limits of the sub-area 3.

Source

The data was collected by researchers from PESAGRO and EMBRAPA-Solos, Rio de Janeiro, Brasil and provided by Dra. Maria Cristina Neves de Oliveira.

Capeche, C.L.; Macedo, J.R.; Manzatto, H.R.H.; Silva, E.F. (1997) Caracterização pedológica da fazenda Angra - PESAGRO/RIO - Estação experimental de Campos (RJ). (compact disc). In: Congresso BRASILEIRO de Ciência do Solo. 26., Informação, globalização, uso do solo; Rio de Janeiro, 1997. trabalhos. Rio de Janeiro: Embrapa/SBCS.

References

Oliveira, M.C.N. (2003) *Métodos de estimação de parâmetros em modelos geoestatísticos com diferentes estruturas de covariâncias: uma aplicação ao teor de cálcio no solo*. Tese de Doutorado, ESALQ/USP/Brasil.

Further information on the package **geoR** can be found at:

<http://www.leg.ufpr.br/geoR>.

Description

- Version 3.3
 - RFempiricalvariogram, RFempiricalcovariance and RFempiricalmadogram became obsolete. Use [RFvariogram](#), [RFcov](#), [RFmadogram](#) instead.
 - RFOptions(grDefault=FALSE) returns to the old style of graphical device handling. Otherwise there is no handling.
 - C code is started to be parallelized.
 - Some new [Multivariate RMmodels](#)
 - New way of passing models, see [RFformula](#), which allows connections (formulae) between parameters, e.g. one parameter value might be twice as large as another parameter value. Also dummy variables can be [RMdeclared](#).
- Options getting obsolete (Version 3 and older)
 - oldstyle is becoming warn_oldstyle
 - newstyle is becoming warn_newstyle
 - 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**

- several changes in `ROptions()`\$graphics in version 3.1.11
- `pdfnumber` became in version 3.0.42 `filenumber`
- `pdfonefile` became in version 3.0.42 `onefile`
- `pdffile` became in version 3.0.42 `file`
- `tbdim` 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`

Author(s)

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

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 guaranteed 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 restrictions.

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 [RPsirculant](#) method, so that exact simulation is guaranteed for further covariance models, e.g. the [RMwhittle](#) model.

In fact, the circulant embedding is called with the cutoff hypermodel, see [RMcutoff](#). Cutoff halves 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 [RPsirculant](#) 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, boxcox, force, mmin, strategy,
            maxGB, maxmem, tolIm, tolRe, trials, useprimes, dependent,
            approx_step, approx_maxgrid)
```

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

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

Arguments

phi	See RPgauss .
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
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 (<code>force=TRUE</code>) after <code>trials</code> number of trials. Default: <code>FALSE</code> .
mmin	Scalar or vector, integer if positive. <code>CE.mmin</code> determines the initial size of the circulant matrix. If <code>CE.mmin=0</code> the minimal starting size is determined automatically according to the dimensions of the grid. If <code>CE.mmin>0</code> then the absolute starting size is given. If <code>CE.mmin<0</code> then the automatically determined matrix size is multiplied by $ CE.mmin $; here, <code>CE.mmin</code> must be smaller than <code>-1</code> ; the value <code>-1</code> takes over the minimal starting size. Note: in any cases, the initial size might be increased according to <code>CE.useprimes</code> . Default: <code>0</code> .
strategy	Logical. <code>0</code> : If the circulant matrix has negative eigenvalues then the size in each direction is doubled; <code>1</code> : 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 <code>trials</code> is probably too small in that case. In some cases <code>strategy=0</code> works better, in other cases <code>strategy=1</code> . Just try. Clearly, if the field is isotropic and a square grid should be simulated, then <code>strategy=0</code> is the better choice. Default: <code>0</code> .

maxGB	Maximal memory used for the circulant matrix in units of GB. If this argument is set then maxmem is set to MAXINT. Default: 1.
maxmem	Integer. maximal number of entries in a row of the circulant matrix. The total amount of memory needed for the internal calculations is $32 (= 4 * \text{sizeof}(\text{double}))$ times 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 lie 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>, <http://ms.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 \mathbb{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

[Gaussian, RP](#)

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=2)
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 a 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 approaches a Gaussian random field with covariance function C .

Usage

```
RPCoins(phi, shape, boxcox, intensity, method)
```

```
RPaverage(phi, shape, boxcox, intensity, method)
```

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.
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
intensity	positive number, intensity of the underlying Poisson point process.
method	integer. Default is the value 0 which addresses the current standard procedure. There might be further methods implemented mainly for internal purposes.

Value

`RPcoins` returns an object of class `RMmodel`.

Author(s)

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

References

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

See Also

`Gaussian`, `RP`, `RPhyperplane`, `RPspectral`, `RPtbn`.

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_NAMES = c("tail correlation", "positive definite", "variogram", "negative definite", "point-shape
function", "shape function", "trend", "distribution or shape", "of manifold type", "process", "method
for Gauss process", "normed process (non-negative values with maximum value being 0 or
1)", "method for Brown-Resnick process", "Smith", "Schlather", "Poisson", "PoissonGauss", "distribution
family", "interface", "mathematical operator", "other type")
```

```
RC_DOMAIN_NAMES = c("single variable", "kernel", "framework dependent", "submodel dependent", "parameter
dependent", "<keep copy>", "mismatch")
```

```
RC_ISO_NAMES = c("isotropic", "space-isotropic", "vector-isotropic", "symmetric", "cartesian
system", "gnomonic projection", "orthographic projection", "spherical isotropic", "spherical
symmetric", "spherical system", "earth isotropic", "earth symmetric", "earth system", "cylinder
system", "non-dimension-reducing", "framework dependent", "submodel dependent", "parameter
dependent", "<internal keep copy>", "<mismatch>")
```

```
RC_MONOTONE_NAMES = c("not set", "mismatch in monotonicity", "submodel dependent monotonicity", "previous
model dependent monotonicity", "parameter dependent monotonicity", "not monotone", "monotone", "Gneiting-S
class", "normal mixture", "completely monotone", "Bernstein")
```

RC_ISOTROPIC gives the numerical code for option "isotropic"

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

RC_CARTESIAN_COORD gives the numerical code for option "cartesian system"

RC_GNOMONIC_PROJ gives the numerical code for the gnomonic projection, see also zenit in [RFoptions](#).

RC_ORTHOGRAPHIC_PROJ gives the numerical code for the orthographic projection, see also zenit in [RFoptions](#).

RC_EARTH_COORDS gives the numerical code for option "earth coordinates"

RC_EARTH_ISOTROPIC gives the numerical code for option "earth isotropic"

RC_SPHERICAL_COORDS gives the numerical code for option "earth coordinates"

RC_OPTIMISER_NAMES and RC_NLOPTR_NAMES give the names for the options optimiser and algorithm, respectively, `RFfitoptimiser`.

RC_LIKELIHOOD_NAMES = c("auto", "full", "composite", "tesselation") gives the names of the ML variants: (i) internal choice according to the number of data, (ii) full likelihood, (iii) (pair-wise) composite likelihood, and (iv) composite likelihood based on a tessellation of the space.

Author(s)

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

See Also

[RF](#), [RM](#), [RP](#), [RR](#), [R](#)., [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_ISO_NAMES
RC_ISO_NAMES[RC_ISOTROPIC:RC_CARTESIAN_COORD + 1]
## Not run:
RFgetModelNames(isotropy=RC_ISO_NAMES[RC_ISOTROPIC:RC_CARTESIAN_COORD +
1])

## End(Not run)
```

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, T=NULL, vdim_close_together)
```

Arguments

data	array; of dimension $c(vdim, space-time-dim, n)$; contains the values of the random field
coords	matrix of coordinates
gridTopology	3-row-matrix or of class <code>GridTopology</code> ; specifies the grid vectors; either <code>coords</code> or <code>gridTopology</code> must be <code>NULL</code>
n	number of iid copies of the random field, default is 1
vdim	number of dimensions of the values of the random field, default is 1
T	time component if any. The length of the temporal grid is needed by <code>as.array</code> if the spatial locations are randomly scattered.
vdim_close_together	logical. Currently, only <code>vdim_close_together=FALSE</code> 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, Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.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)
```

Description

Implemented Coordinate Systems.

Implemented coordinate systems

- Cartesian coordinate system
- Earth coordinate systems
The earth is considered as an ellipsoid; The first angle takes values in $[0, 360)$, the second angle takes values in $[-90, 90]$.
- Spherical coordinate systems
The earth is considered as an ellipsoid; The first angle takes values in $[0, 2\pi)$, the second angle takes values in $[-\pi/2, \pi/2]$.

Transformations between the system

- Earth to cartesian
The 3-dimensional resulting coordinates are either given in 'km' or in 'miles'.
- Gnomonic and orthographic projections
The 2-dimensional resulting coordinates are either given in 'km' or in 'miles'. The projection direction is given by the zenit.
- Earth to spherical
In this case the Earth is considered as a ball.

Cartesian systems cannot be transformed to earth or spherical coordinate systems, nor a spherical system to earth coordinates.

Options

`coord_system` character. One of the values "auto", "cartesian", "earth"

If "auto", then the coordinates are considered as "cartesian" except the names of the given coordinates indicate a different system. Currently, only "longitude" and "latidute" (or abbreviations of them) are excepted as names for given coordinates and indicate an earth coordinate systems. See the examples below.

Default: "auto"

`coordidx` integer vector of column numbers of the variables in the data frame. `varidx` can be set alternatively to `coordnames`. This parameter gives the coordinate columns in a data frame by starting column and ending column or the sequence. An NA in the second component means 'until the end'.

`coordnames` vector of characters that can be set alternatively to `coordidx`. This parameter gives the coordinate columns in a data frame by names. If it is "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: ""

`new_coord_system` One of the values "keep", "cartesian", "earth", "plane".

1. "keep"

The `coord_system` is kept (except an explicit transformation is given, see [RMtrafo](#)).

Note that some classes of models, e.g. completely monotone functions and compactly supported covariance models with range less than π are valid models on a sphere. In this case the models are considered as models on the sphere. See [spherical models](#) for lists.

2. "cartesian"

If `coord_system` is "earth" the coordinates are transformed to cartesian coordinates before any model is considered.

3. "orthographic", "genomic"

If `coord_system` is "earth" the locations are projected to a plane before any model is considered.

Default: "keep"

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

Default: ""

`polar_coord` logical. If FALSE the spherical coordinates agree with the earth coordinate parametrization, except that radians are used for spherical coordinates instead of degrees for the earth coordinates.

If TRUE the spherical coordinates signify polar coordinates.

Default: FALSE

`varidx` integer vector of length 2. `varidx` can be set alternatively to `varnames`. This parameter gives the data columns in a data frame, either by starting column and ending column. An NA in the second component means 'until the end'.

`varnames` vector of characters that can be set alternatively to `varidx`. This parameter gives the data columns in a data frame by names.

if `varnames` equals "NA" then for keywords 'data', 'value' and 'variable' will be searched for keywords. 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.

`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 kernel definition, not as xyz notation

TRUE: xyz notation used

`zenit` two angles of the central projection direction for the gnomonic projection (http://en.wikipedia.org/wiki/Gnomonic_projection, http://de.wikipedia.org/wiki/Gnomonische_Projektion) and the orthographic projection, (http://en.wikipedia.org/wiki/Orthographic_projection_in_cartography, http://de.wikipedia.org/wiki/Orthografische_Azimutalprojektion).

If any(`is.na(zenit)`) then either the value of either of the components may not be NA, whose value will be denoted by p .

If $p = 1$ then the mean of the locations is calculated; if $p = Inf$ then the mean of the range is calculated.

Default: `c(1, NA)`

References

Covariance models in a cartesian system

- 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.

Covariance models on a sphere

- Gneiting, T. (2013) Strictly and non-strictly positive definite functions on spheres. *Bernoulli*, **19**, 1327-1349.

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

[RMtrafo](#), [RFearth2cartesian](#), [RPdirect](#), [models valid on a sphere](#), [Rfoptions](#).

Examples

```

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

z <- 1:4
x <- cbind(z, 0)
y <- cbind(0, z)
model <- RMwhittle(nu=0.5)
Rfcov(model, x, y, grid=FALSE) ## standard is (cartesian) model

## same as above, but explicit:
Rfcov(model, x, y, grid=FALSE, coord_sys="cartesian")

## model is not valid on a sphere; x,y coordinates are
## transformed from earth coordinates to spherical coordinates
Rfcov(model, x, y, grid=FALSE, coord_sys="earth")

## now the scale is chosen such that the covariance
## values are comparable to those in the cartesian case
Rfcov(RMS(model, s= 1 / 180 * pi), x, y, grid=FALSE,
      coord_sys="earth")

```

```

## projection onto a plane first. Then the scale is interpreted
## in the usual, i.e. cartesian, sense, i.e. the model does not
## really make sense
RFoptions(zenit = c(2.5, 2.5))
RFcov(model, x, y, grid=FALSE,
      coord_sys="earth", new_coord_sys="orthographic")

## again, here the scale is chosen to be comparable to the cartesian case
## here the (standard) units are [km]
(z1 <- RFcov(RMS(model, s= 6350 / 180 * pi), x, y, grid=FALSE,
            coord_sys="earth", new_coord_sys="orthographic"))

## as above, but in miles
(z2 <- RFcov(RMS(model, s= 6350 / 1.609344 / 180 * pi), x, y, grid=FALSE,
            coord_sys="earth", new_coord_sys="orthographic",
            new_coordunits="miles"))
stopifnot(all.equal(z1, z2))

## again, projection onto a plane first, but now using the
## gnomonic projection
## here the (standard) units are [km]
(z1 <- RFcov(RMS(model, s= 6350 / 180 * pi), x, y, grid=FALSE,
            coord_sys="earth", new_coord_sys="gnomonic"))

## as above, but in miles
(z2 <- RFcov(RMS(model, s= 6350 / 1.609344 / 180 * pi), x, y, grid=FALSE,
            coord_sys="earth", new_coord_sys="gnomonic",
            new_coordunits="miles"))
stopifnot(all.equal(z1, z2, tol=1e-5))

```

Distribution Families *Distribution families (RR commands)*

Description

Distribution families to specify random parameters in the model definition.

Details

See [Bayesian Modelling](#) for a less technical introduction to hierarchical modelling.

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 realization of the random parameters are performed. See the examples below.

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

These functions will allow for Bayesian modelling. (Future project).

Implemented models

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

Note

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

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>

See Also

[RC](#), [RF](#), [RM](#), [RP](#), [Other models](#), [RFdistr](#), [RMmodelgenerator](#), [R](#).

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 the 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 \code{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 field, 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 values or (in future!) be given by an [RMmodel](#), in particular by an [RMtrend](#) model.

For $x_i = 0$, the default values of μ and s are 0 and 1, respectively. For $x_i \neq 0$, the default values of μ and s are 1 and $|\xi|$, respectively, so that it defaults to the standard Frechet case if $\xi > 0$.

Author(s)

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

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

```
## sorry, does not work savely yet
```

ExtremalGaussian	<i>Extremal Gaussian process</i>
------------------	----------------------------------

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 values or (in future!) be given by an `RMmodel`, in particular by an `RMtrend` model.

For $xi = 0$, the default values of mu and s are 0 and 1, respectively. For $xi \neq 0$, the default values of mu and s are 1 and $|\xi|$, respectively, so that it defaults to the standard Frechet case if $\xi > 0$.

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

It simulates an 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 `phi`, and c is chosen such that Z has standard Frechet margins. `phi` 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>

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-stable random field has extreme value index 0
x <- seq(0, 2, 0.01)

## 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 following definition is incorrect
try(RFsimulate(model=RPschlather(RMgauss(var=2)), x=x))

## check whether the marginal distribution (Gumbel) is indeed correct:
model <- RMgauss()
z <- RFsimulate(RPschlather(model, xi=0), x, n=100)
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, including Box-Cox transformation

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 minimize

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

over all parametrized 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>

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

Here, 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
RPdirect	through the square root of the covariance matrix
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
RPtbn	turning bands

Computing demand for simulations

Assume at n locations in d dimensions a v -variate field has to be simulated. Let

$$f(n, d) = 2^d n \log(n)$$

The following table gives in particular the time and memory needed for the specific simulation method.

	grid	v	d	time	memory	comments
RPCirculant	yes	any	≤ 13	$O(v^3 f(n, d))$	$O(v^2 f(n, d))$	
	no	any	≤ 13	$O(v^3 f(k, d))$	$O(v^2 f(k, d))$	$k \sim \text{approx_step}^{-d}$
RPCutoff						see RPCirculant above
RPCoins	yes	1	≤ 4	$O(kn)$	$O(n)$	$k \sim (\text{latticespacing})^{-d}$
	no	1	≤ 4	$O(kn)$	$O(n)$	k depends on the geometry
RPdirect	any	any	any	$O(1)..O(v^2 n^2)$	$O(v^2 n^2)$	effort to investigate the covariance matrix, if
				$O(vn)$	$O(vn)$	covariance matrix is diagonal
				see spam	$O(z + vn)$	covariance matrix is sparse matrix with z non-
				$O(v^3 n^3)$	$O(v^2 n^2)$	arbitrary covariance matrix (preparation)
				$O(v^2 n^2)$	$O(v^2 n^2)$	arbitrary covariance matrix (simulation)
RPgauss	any	any	any	$O(1) \dots O(v^3 n^3)$	$O(1) \dots O(n^2)$	only the selection process; $O(1)$ if first metho
RPhyperplane	any	1	2	$O(n/s^d)$	$O(n/s^d)$	$s = \text{scale}$
RPintrinsic						see RPCirculant above
RPnugget	any	any	any	$O(vn)$	$O(vn)$	
RPsequential	any	1	any	$O(S^3 b^3)$	$O(S^2 b^2)$	$n = ST$; S and T the number of spatial and t
				$O(nSb^2)$	$O(S^2 b^2) + O(n)$	(simulation)
RPspectral	any	1	≤ 2	$O(C(d)n)$	$O(n)$	$C(d)$: large constant increasing in d
RPtbn	any	1	≤ 4	$O(C(d)(n + L))$	$O(n + L)$	$C(d)$: large constant increasing in d ; L is the
RPspecific						only the specific part
** RMplus	any	any	any	$O(v n)$	$O(v n)$	
** RMS	any	any	any	$O(1)$	$O(v n)$	
** RMult	any	any	any	$O(v n)$	$O(v n)$	

Computing demand for interpolation

Assume v -variate data are given at n locations in d dimensions. To interpolate at k locations RandomFields needs

grid	v	d	time	memory	comments
any	any	any	$O(1)..O(v^2 n^2)$	$O(v^2 n^2)$	effort to investigate the covariance matrix, if matrix_methods
			$O(v^2 nk)$	$O(v(n + k))$	covariance matrix is diagonal
			see spam + $O(v^2 nk)$	$O(z + v(n + k))$	covariance matrix is sparse matrix with z non-zeros
			$O(v^3 n^3 + v^2 nk)$	$O(v^2 n^2 + v * k)$	arbitrary covariance matrix

Computing demand for conditional simulation

Assume v -variate data are given at n locations x_1, \dots, x_n in d dimensions. To conditionally simulate at k locations y_1, \dots, y_k , the computing demand equals the sum of the demand for interpolating and the demand for simulating on the $k + n$ locations. (Grid algorithms for simulating will apply if the k locations y_1, \dots, y_k are defined by a grid and the n locations x_1, \dots, x_n are a subset of y_1, \dots, y_k , a situation typical in image analysis.)

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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. (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, 3rd edition.

See Also

[RP](#), [Other models](#), [RMmodel](#), [RFgetMethodNames](#), [RFsimulateAdvanced](#).

Examples

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

set.seed(1)
x <- runif(90, 0, 500)
z <- RFsimulate(RMspheric(), x)
z <- RFsimulate(RMspheric(), x, max_variab=10000)
```

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>

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

```

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

## Figure 1 (pretty time consuming)
stabletest <- function(alpha, theta, size=512) {
  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(modelname="RMcutoff", level=3)$storage$R_theor
  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, 0.02)
thetas <- seq(0.05, 3.5, 0.05)

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))
}

```

Hierarchical Modelling

Bayesian Spatial Modelling

Description

RandomFields provides Bayesian modelling to some extent: (i) simulation of hierarchical models at arbitrary depth; (ii) estimation of the parameters of a hierarchical model of depth 1 by means of maximizing the likelihood.

Details

A Bayesian approach can be taken for scalar, real valued model parameters, e.g. the shape parameter ν in the [RMmatern](#) model. A random parameter can be passed through a distribution of an existing family, e.g. (dnorm, pnorm, qnorm, rnorm) or self-defined. It is passed without the leading letter d,

p , q , r , but as a function call e.g. `norm()`. This function call may contain arguments that must be named, e.g. `norm(mean=3, sd=5)`.

Usage:

- `exp()` denotes the exponential distribution family with rate 1,
- `exp(3)` is just the scalar e^3 and
- `exp(rate=3)` is the exponential distribution family with rate 3.

The family can be passed in three ways:

- implicitly, e.g. `RMwhittle(nu=exp())` or
- explicitly through `RRdistr`, e.g. `RMwhittle(nu=RRdistr(exp()))`.
- by use of `RRmodels` of the package.

The first is more convenient, the second more flexible and slightly safer.

Note

- While simulating any depth of hierarchical modelling is possible, estimation is currently restricted to one level of hierarchy.
- The effect of the distribution family varies between the different processes:
 - in max-stable fields and `RPpoisson`, a new realization of the prior distribution(s) is drawn for each shape function
 - in all other cases: a realization of the prior(s) is only drawn once. This effects, in particular, Gaussian fields with argument $n>1$, where all realizations are based on the same realization out of the prior distribution(s).

Note that checking the validity of the arguments is rather limited for such complicated models, in general.

Author(s)

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

## the following model defines the argument nu of the Whittle-Matern
## model to be an exponential random variable with rate 5.
model <- ~ 1 + RMwhittle(scale=NA, var=NA, nu=exp(rate=5)) + RMnugget(var=NA)

data(soil)
fit <- RFfit(model, x=soil$x, y=soil$y, data=soil$moisture, modus="careless")
print(fit)
```

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

```
RHyperplane(phi, boxcox, superpos, maxlines, mar_distr, mar_param ,additive)
```

Arguments

phi	object of class RMmodel ; specifies the covariance function to be simulated; only exponential covariance functions and scale mixtures of it are allowed.
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
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.
additive	logical. If TRUE independent realizations are added, else the maximum is taken. Default: TRUE.

Value

RHyperplane returns an object of class [RMmodel](#).

Author(s)

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

References

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

See Also

[Gaussian, RP](#).

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                RFoptions(seed=NA) to make them all random again
model <- RHyperplane(RMexp(s=2), superpos=1)
x <- seq(0, 3, 0.04)
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. (Only for advanced users)

Usage

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

Arguments

phi	object of class RMmodel ; specifies the covariance model to be simulated. The only possible model for phi is RMnugget .
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
tol	points at a distance less than or equal to <code>nugget.tol</code> 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 <code>nugget.tol=0.0</code> . However, if the anisotropy matrix does not have full rank and <code>nugget.tol=0.0</code> , then the simulations are likely to be odd. The value of <code>nugget.tol</code> should be of order 10^{-15} . Default: <code>0.0</code>
vdim	positive integer; the model is treated <code>vdim</code> -variate, <code>vdim=1</code> (default) corresponds to a univariate random field. Mostly, the value of <code>vdim</code> is set automatically. Default is that it takes the value of the submodel phi.

Details

General This method only allows [RMnugget](#) as a submodel.

Anisotropy 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 kernel of the matrix is important.

Points close together The locations at a distance less than or equal to the [RFoptions](#) `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} .

Repeated measurements Measurement errors are mathematically not distinguishable from spatial nugget effects as long as measurements are not repeated at the very same space-time location. So there is no need to distinguish the spatial nugget effect from a measurement error. This is the default, see `allow_duplicated_locations` in [RFoptions](#).

In case several measurement have been taken in single space-time locations, measurement errors can be separated from spatial noise. In this case `RMnugget()` models the measurement error (which corresponds to a non-stationary model in an abstract space) by default and the measurement error model cannot be extended beyond the given locations. On the other hand `RMnugget(Aniso=something)` and `RMnugget(proj=something)` model the spatial nugget effect (with and without zonal anisotropy in case `Aniso` has low and full rank respectively).

Role of RPnugget Even for advanced users, there is no need to call `RPnugget` directly, as this is done internally when the [RMnugget](#) is involved in the covariance model.

Value

`RPnugget` returns an object of class [RMmodel](#).

Author(s)

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

[Gaussian](#), [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
```



```

x <- y <- 1:2
xy <- as.matrix(expand.grid(x, y)) ## we get 4 locations
## Standard use of the nugget effect
model <- RMnugget(var = 100)
Rfcovmatrix(model, x=xy)
as.vector(RFsimulate(model, x=x, y=x, tol=1e-10))

## zonal nugget effect, which is not along the axes
model <- RMnugget(Aniso=matrix(1, nr=2, nc=2))
Rfcovmatrix(model, x=xy)
as.vector(RFsimulate(model, x=x, y=x, tol=1e-10))

## All the following examples refer to repeated measurements
RFOptions(allow_duplicated_locations = TRUE)
(xy <- rbind(xy, xy)) ## now, the 4 locations are repeated twice

## standard situation: the nugget is interpreted as measurement error:
model <- RMnugget()
Rfcovmatrix(model, x=xy)
as.matrix(RFsimulate(model, x=xy))

## any anisotropy matrix with full rank: spatial nugget effect
model <- RMnugget(Aniso=diag(2))
Rfcovmatrix(model, x=xy)
as.matrix(RFsimulate(model, x=xy))

## anisotropy matrix with lower rank: zonal nugget effect
model <- RMnugget(Aniso=matrix(c(1, 0, 0, 0), nc=2))
Rfcovmatrix(model, x=xy)
as.matrix(RFsimulate(model, x=xy))

## same as before: zonal nugget effect
model <- RMnugget(Aniso=t(c(1,0)))
Rfcovmatrix(model, x=xy)
as.matrix(RFsimulate(model, x=xy))

```

Internal functions *Internal functions*

Description

These functions are internal and should not be used.

Usage

```

rfGenerateModels(package = "RandomFields", assigning,
                 RFpath = "~/svn/RandomFields/RandomFields",
                 RMmodels.file=paste(RFpath, "R/RMmodels.R", sep="/"),
                 PL = RFOptions()$basic$printlevel)

```

```

rfGenerateConstants(package="RandomFields", aux.package = "RandomFieldsUtils",
  RFpath = paste0("~/svn/",package, "/", package),
  RCauto.file = paste(RFpath, "R/aaa_auto.R", sep="/"),
  header.source =
  c(if (length(aux.package) > 0) paste0("../..", aux.package,"/",
aux.package, "/src/Auto", aux.package, ".h"),
  paste0("src/Auto",package,".h")),
  c.source = paste0("src/Auto", package, ".cc"))

rfGenerateTest(package = "RandomFields", files = NULL,
  RFpath = paste0("~/svn/", package, "/", package))

rfGenerateMaths(package = "RandomFields",
  files = "/usr/include/tgmath.h",
  do.cfile = FALSE,
  ## copy also in ../private/lit
  Cfile = "QMath",
  Rfile = "RQmodels",
  RFpath = paste0("~/svn/",package,"/", package))

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

ScreenDevice(height, width)

FinalizeExample()
StartExample(reduced = TRUE, save.seed = TRUE)

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, ...)

maintainers.machine()

```

Arguments

package, assigning, RFpath, RMmodels.file, PL
internal

```

aux.package,RCauto.file,header.source,c.source
      internal
files      internal
Cfile, Rfile, do.cfile
      internal
exclude, include, ask, echo, halt, ignore.all, path, read.rd.files, libpath, single.runs
      internal; ignore.all refers 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 searched in the environments) are
      run.
height,width  window sizes
data, factor, xlim, ylim, col, fill, ...
      internal
reduced, save.seed, local
      internal

```

Author(s)

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

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`’ is given.

Author(s)

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

References

Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Strokorb, K. (2015) Analysis, simulation and prediction of multivariate random fields with package **RandomFields**. *Journal of Statistical Software*, **63** (8), 1-25, url = ‘<http://www.jstatsoft.org/v63/i08/>’

See Also

[weather](#), [SS12](#), [S10](#).

Examples

```
## Not run:
#####
## SECTION 4: UNCONDITIONAL SIMULATION ##
#####

RFOptions(seed = 0, height = 4)
## 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_device=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 latent 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 coregionalization
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))
simu <- RFsimulate(model, x, y)
plot(simu)
```

```
## Fig. 7: random vector field whose paths 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 fourth 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] ## full data set takes more than
##                       half an hour to be analysed
## transformation of earth coordinates to Euclidean coordinates
Dist.mat <- as.vector(RFearth2dist(weather[ , 3 : 4]))
```

```

All <- TRUE
\dontshow{if(RFoptions()$internal$examples_reduced){warning("reduced data set")}
All <- 1:10
PT <- weather[All , 1 : 2]
Dist.mat <- as.vector(RFearth2dist(weather[All , 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  ##
#####
## 'parsimonious model'
## fitting takes a while ( > 10 min)
pars <- RFFit(pars.model, distances = Dist.mat, dim = 3, data = PT)
print(pars)
print(pars, full=TRUE)
RFratiotest(pars)
#RFCrossvalidate(pars, x = weather[All , 3 : 4], data = PT, full = TRUE)

## 'whole model'
## fitting takes a while ( > 10 min)
whole <- RFFit(whole.model, distances = Dist.mat, dim = 3, data = PT)
print(whole, full=TRUE)
RFratiotest(whole)
#RFCrossvalidate(whole, x = weather[All , 3 : 4], data = PT, full = TRUE)

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

#####
## kriging              ##
#####
## First, the coordinates are projected on a plane
a <- colMeans(weather[All , 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[All , 3 : 4])
plane <- (t(x) %*%P)[ , 1 : 2]

## here, kriging is performed on a rectangular that covers
## the projected points above. The rectangular is approximated
## by a grid of length 200 in each direction.
n <- 200
r <- apply(plane, 2, range)
dta <- cbind(plane, weather[All , 1 : 2])
z <- RFinterpolate(pars, data = dta, 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)

## End(Not run)
```

Description

This man page documents some major changings in RandomFields.

Changes done in 3.1.0 (Summer 2015)

- full (univariate) trend modelling
- error in particular in [RFfit](#) corrected
- [RFfit](#) runs much faster now
- effects of modus operandi changed for estimating

Corrections done in 3.0.56 (Jan 2015)

- log Gauss field corrected (has been a log log Gauss field)
- [RMconstant](#) is now called [RMfixcov](#)

Corrections done in 3.0.55 (Jan 2015)

- Conditional simulation: several severe typos corrected.

Major Revision: changings from Version 2 to Version 3 (Jan 2014)

- **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 than 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 of random effect models with spatial covariance structure
 - (ii) automatic estimation of 10 and more arguments in multivariate and/or space-time models
 - **RFvariogram** is now based on an fft algorithm if the data are on a grid, even allowing for missing values.
 - **RFratiotest** has been added.
- **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 be included which will be extended to **Bayesian** modelling in future.

Author(s)

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

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, 0.1)
model <- RMexp()
plot(RFsimulate(model, x)) ## S4
plot(RFsimulate(model, x, spConform=FALSE)) ## no class

```

Mathematical C functions

Transformation of coordinate systems

Description

The functions provide mathematical c functions as [RMmodels](#)

Usage

```

RFcalc(model, params, ...)
R.minus(x, y, factor)
R.plus(x, y, factor)
R.div(x, y, factor)
R.mult(x, y, factor)
R.const(x)
R.c(a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, q, ncol, factor)
R.p(proj, new, factor)
R.is(x, is, y)
R.lon()
R.lat()

R.gamma(x)
R.acos(x)
R.asin(x)
R.atan(x)
R.atan2(y, x)
R.cos(x)
R.sin(x)
R.tan(x)
R.acosh(x)
R.asinh(x)
R.atanh(x)
R.cosh(x)
R.sinh(x)

```

```
R.tanh(x)
R.exp(x)
R.log(x)
R.expm1(x)
R.log1p(x)
R.exp2(x)
R.log2(x)
R.pow(x, y)
R.sqrt(x)
R.hypot(x, y)
R.cbrt(x)
R.ceil(x)
R.fabs(x)
R.floor(x)
R.fmod(x, y)
R.round(x)
R.trunc(x)
R.erf(x)
R.erfc(x)
R.lgamma(x)
R.remainder(x, y)
R.fdim(x, y)
R.fmax(x, y)
R.fmin(x, y)

## S4 method for signature 'ANY,RMmodel'
e1 %% e2
## S4 method for signature 'RMmodel,ANY'
e1 %% e2
## S4 method for signature 'RMmodel,character'
e1 * e2
## S4 method for signature 'character,RMmodel'
e1 * e2
## S4 method for signature 'RMmodel,character'
e1 + e2
## S4 method for signature 'RMmodel,factor'
e1 + e2
## S4 method for signature 'RMmodel,list'
e1 + e2
## S4 method for signature 'character,RMmodel'
e1 + e2
## S4 method for signature 'data.frame,RMmodel'
e1 + e2
## S4 method for signature 'factor,RMmodel'
e1 + e2
## S4 method for signature 'RMmodel,character'
e1 - e2
## S4 method for signature 'character,RMmodel'
```

```
e1 - e2
## S4 method for signature 'RMmodel,character'
e1 / e2
## S4 method for signature 'character,RMmodel'
e1 / e2
## S4 method for signature 'ANY,RMmodel'
e1 ^ e2
## S4 method for signature 'RMmodel,ANY'
e1 ^ e2
## S4 method for signature 'RMmodel,character'
e1 ^ e2
## S4 method for signature 'character,RMmodel'
e1 ^ e2
## S4 method for signature 'RMmodel'
abs(x)
## S4 method for signature 'RMmodel'
acosh(x)
## S4 method for signature 'RMmodel'
asin(x)
## S4 method for signature 'RMmodel'
asinh(x)
## S4 method for signature 'ANY,RMmodel'
atan2(y,x)
## S4 method for signature 'RMmodel,ANY'
atan2(y,x)
## S4 method for signature 'RMmodel'
atan(x)
## S4 method for signature 'RMmodel'
atanh(x)
## S4 method for signature 'RMmodel'
ceiling(x)
## S4 method for signature 'RMmodel'
cos(x)
## S4 method for signature 'RMmodel'
cosh(x)
## S4 method for signature 'RMmodel'
exp(x)
## S4 method for signature 'RMmodel'
expm1(x)
## S4 method for signature 'RMmodel'
floor(x)
## S4 method for signature 'RMmodel'
lgamma(x)
## S4 method for signature 'RMmodel'
log1p(x)
## S4 method for signature 'RMmodel'
log2(x)
## S4 method for signature 'RMmodel'
```

```

log(x)
## S4 method for signature 'RMmodel,missing'
round(x,digits)
## S4 method for signature 'RMmodel'
sin(x)
## S4 method for signature 'RMmodel'
sinh(x)
## S4 method for signature 'RMmodel'
sqrt(x)
## S4 method for signature 'RMmodel'
tan(x)
## S4 method for signature 'RMmodel'
tanh(x)
## S4 method for signature 'RMmodel'
trunc(x)

```

Arguments

<code>model, params</code>	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument <code>params</code> is a list that specifies free parameters in a formula description, see RMformula . <code>model</code> is usually a R. <code>model</code> given here.
<code>e1, e2, x, y, a, b, c, d, e, f, g, h, i, j, l, m, n, o, p, q</code>	constant or object of class RMmodel , in particular R. <code>model</code>
<code>ncol</code>	in contrast to <code>c</code> , R. <code>c</code> also allows for defining matrices; <code>ncol</code> gives the number of columns
<code>factor</code>	constant factor multiplied with the function. This is useful when linear models are built
<code>is</code>	one of " <code>==</code> ", " <code>!=</code> ", " <code><=</code> ", " <code><</code> ", " <code>>=</code> ", " <code>></code> "
<code>proj</code>	selection of a component of the vector giving the location. Default value is 1.
<code>new</code>	coordinate system or other kind of isotropy which is supposed to be present at this model. It should always be given if the coordinates are not cartesian.
<code>digits</code>	number of digits. Does not work with a RMmodel
<code>...</code>	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If <code>params</code> is given, then <code>...</code> may include also the variables used in <code>params</code> .

Details

R.plus adds two values

R.minus subtracts two values

R.mult multiplies two values

R.div divides two values

R.const defines a constant

R.c builds a vector

R.is evaluates equalities and inequalities; note that TRUE is returned if the equality or inequality holds up to a tolerance given by `RFoptions()$nugget$tol`

R.p takes a component out of the vector giving the location

R.lon, R.lat longitudinal and latitudinal coordinate, given in the *spherical system*, i.e. in radians. (earth system is in degrees).

R.round Note that R. round rounds away from zero.

For the remaining models see the corresponding C functions for their return value. (For any 'R.model' type 'man model' under Linux.)

Value

Formally, the functions return an object of class `RMmodel`, except for `RFcalc` that returns a scalar. Neither vectors nor parentheses are allowed.

Note

Instead of `R.model` the standard function can be used in case there is no ambiguity, i.e., `c(...)`, `asin(x)`, `atan(x)`, `atan2(y, x)`, `cos(x)`, `sin(x)`, `tan(x)`, `acosh(x)`, `asinh(x)`, `atanh(x)`, `cosh(x)`, `sinh(x)`, `tanh(x)`, `exp(x)`, `log(x)`, `expm1(x)`, `log2(x)`, `log1p(x)`, `exp2(x)`, `^`, `sqrt(x)`, `hypot(a, b)`, `cbrt(x)`, `ceiling(x)`, `abs(x)`, `floor(x)`, `round(x)`, `trunc(x)`, `erf(x)`, `erfc(x)`, `lgamma(x)`. See the examples below.

The function `RFcalc` is intended for simple calculations only and it is not excessively tested. Especially, binary operators should be used with caution.

Note that all the functions here are NOT recognized as being positive definite (or negative definite), e.g. `cos` in R^1 :

1. please use the functions given in `RMmodels` for definite functions (for `cos` see `MBessel`)
2. Using uncapsulated subtraction to build up a covariance function is ambiguous, see the example in `RMtrend`

Author(s)

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

See Also

`RMmodel`, `RFctn`, `RMtrend`

Examples

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

## simple calculation
RFcalc(3 + R.sin(pi/4))
```

```

## calculation performed on a field
RFfctn(R.p(1) + R.p(2), 1:3, 1:3)
RFfctn(10 + R.p(2), 1:3, 1:3)

## calculate the distances between two vectors
print(RFfctn(R.p(new="iso"), 1:10, 1:10))

## simulation of a non-stationary field where
## anisotropy by a transform the coordinates (x_1^2, x_2^1.5)
x <- seq(0.1, 6, 0.12)
Aniso <- R.c(R.p(1)^2, R.p(2)^1.5)
z <- RFsimulate(RMexp(Aniso=Aniso), x, x)

## calculating norms can be abbreviated:
x <- seq(-5, 5, 5) #0.1)
z2 <- RFsimulate(RMexp() + -40 + exp(0.5 * R.p(new="isotropic")), x, x)
z1 <- RFsimulate(RMexp() + -40 + exp(0.5 * sqrt(R.p(1)^2 + R.p(2)^2)), x, x)
stopifnot(all.equal(z1, z2))
plot(z1)

```

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.

Implemented models and methods

Models

RPbrownresnick	Brown-Resnick process using an automatic choice of the 3 RPbr* methods below
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>

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

### currently not programmed

## 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)
z <- RFsimulate(RPsmith(model, xi=0), x, n=1000, 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()
x <- seq(0, 10, 0.05)
z <- RFsimulate(RPschlather(model, xi=0), x, n=1000)
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
```

Max-stable random fields, advanced

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>, <http://ms.math.uni-mannheim.de>

References

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

See Also

[RPmaxstable](#)

Examples

Obsolete Functions Version 2

Obsolete functions Version 2

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 = NULL,
  Distances, fctcall = c("Cov", "Variogram", "CovMatrix"))
CovarianceFct(x, y = NULL, model, param = NULL, dim = NULL,
  Distances, fctcall = c("Cov", "Variogram", "CovMatrix"))
CovMatrix(x, y = NULL, model, param = NULL, dim = NULL, 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 = NULL, Distances)
InitMaxStableRF(x, y = NULL, z = NULL, grid=NULL, model, param, maxstable,
  method = NULL, register = 0, gridtriple = FALSE)
MaxStableRF(x, y = NULL, z = NULL, grid=NULL, model, param, maxstable,
  method = NULL, n = 1, register = 0, gridtriple = FALSE, ...)
EmpiricalVariogram(x, y = NULL, z = NULL, T=NULL, data, grid=NULL, bin,
  gridtriple = FALSE, phi, theta, deltaT)
Kriging(krige.method, x, y=NULL, z=NULL, T=NULL, grid=NULL, 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=NULL, 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()$basic$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()$basic$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, gridtriple, di
 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>

See Also

The functions that should be used instead are, for instance, [RFcov](#), [RFcovmatrix](#), [RFvariogram](#), [RFsimulate](#), [RFinterpolate](#), [RFvariogram](#), [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
```

Obsolete Functions Version 3

Obsolete functions Version 3

Description

Some functions of RandomFields Version 3 have been replaced by more powerful functions

Usage

```
RFempiricalvariogram(...)
RFempiricalcovariance(...)
RFempiricalmadogram(...)
```

Arguments

... See for the recent functions given in the Details

Details

RFempiricalvariogram see [RFvariogram](#)

RFempiricalcovariance see [RFcov](#)

RFempiricalmadogram see [RFmadogram](#)

Storkorb's M3/M4 functions are called [RMm2r](#), [RMm3b](#), [RMmps](#)

Value

see the respective recent function

Author(s)

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

See Also

[RFCov](#), [RFCovmatrix](#), [RFvariogram](#), [RFpseudovariogram](#), [RFmadogram](#), [RFpseudomadogram](#)

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](#).

Mathematical functions See [R](#).

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
RMidmodel	model identity
RMid	identity but interpretation turns from a coordinate to a model value
RMtrafo	allows to model the identity within the set of coordinates
RMrotation	shape function used in the Bernoulli paper given in the references

Other models

[RMuser](#) User defined covariance model

Author(s)

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

See Also

[RM](#)

Examples

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

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>

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. (2015) Analysis, simulation and prediction of multivariate random fields with package **RandomFields**. *Journal of Statistical Software*, **63** (8), 1-25, url = 'http://www.jstatsoft.org/v63/i08/',
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](#)).

The plot methods not described here are described together with the class itself, for instance, [RFfit](#), [RFempVariog](#) [RMmodel](#).

Usage

```
RFplotSimulation(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, file=NULL, speed = 0.3,
  height.pixel=300, width.pixel=height.pixel,
  ..., plotmethod="image")
```

```
RFplotSimulation1D(x, y, nmax=6,
  plot.variance=!is.null(x@.RFparams$has.variance) && x@.RFparams$has.variance,
  legend=TRUE, ...)
```

```
## S4 method for signature 'RFgridDataFrame,missing'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFpointsDataFrame,missing'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFspatialGridDataFrame,missing'
```

```
plot(x, y, ...)
```

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

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFgridDataFrame,matrix'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFpointsDataFrame,matrix'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFspatialGridDataFrame,matrix'
```

```
plot(x, y, ...)
```

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

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFgridDataFrame,data.frame'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFpointsDataFrame,data.frame'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFspatialGridDataFrame,data.frame'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFspatialPointsDataFrame,data.frame'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFgridDataFrame,RFgridDataFrame'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFgridDataFrame,RFpointsDataFrame'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFpointsDataFrame,RFgridDataFrame'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RFpointsDataFrame,RFpointsDataFrame'
```

```
plot(x, y, ...)
```

```

## S4 method for signature 'RFspatialGridDataFrame,RFspatialGridDataFrame'
plot(x, y, ...)
## S4 method for signature 'RFspatialGridDataFrame,RFspatialPointsDataFrame'
plot(x, y, ...)
## S4 method for signature 'RFspatialPointsDataFrame,RFspatialGridDataFrame'
plot(x, y, ...)
## S4 method for signature 'RFspatialPointsDataFrame,RFspatialPointsDataFrame'
plot(x, y, ...)

## S4 method for signature 'RFspatialGridDataFrame'
persp(x, ..., zlab="")
## S3 method for class 'RFspatialGridDataFrame'
contour(x, ...)

```

Arguments

x	object of class <code>RFsp</code> or <code>RMmodel</code> ; 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; in case of <code>RFplotSimulation</code> data might be given
MARGIN	vector of two; two integer values giving the coordinate dimensions w.r.t. whether 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.
n.slices	integer value. The number of slices to be plotted; if <code>n.slices</code> >1, <code>nmax</code> is set to 1. Or <code>n.slices</code> is a vector of 3 elements: start, end, length. Currently only works for grids.
nmax	the maximal number of the <code>x@.RFparams\$n</code> 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.
file, speed, height.pixel, width.pixel	In case MARGIN.movie and file is given an 'avi' movie is stored using the mencoder command with speed argument speed. As temporary files <code>file_###.png</code> of size <code>width.pixel</code> x <code>height.pixel</code> are created.
...	arguments to be passed to methods; mainly graphical arguments, or further models in case of class <code>CLASS_CLIST</code> , see Details.
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, <code>length(select.variables)</code> gives the number of pictures shown (excluding the plot for the variances, if applicable). If <code>select.variables</code> is a vector of integers then exactly these components are shown. If <code>select.variables</code> is a list, each element should be a vector up to length $l \leq 3$:

	<ul style="list-style-type: none"> • $l = 1$ 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 components 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 with the value 'joint' indicating that all plotted components 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.
plotmethod	string or function. Internal.
zlab	character. See persp

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 `CLASS_CLIST`. 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.

Author(s)

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

See Also

[RFpar](#).

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)
image(model, xlim=c(-10, 20), fct.type="Variogram")
persp(model, xlim=c(-10, 20), fct.type="Variogram")

#####
## Plot of simulation results

## define the locations:
from <- 0
step <- .1
len <- 50 # nicer if len=100 %ok

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(model, x=cbind(x))
ev1 <- RFvariogram(data=X, bin=bin)
plot(ev1)

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

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

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

#####
## plotting vector-valued results
model <- RMdivfree(RMgauss(), scale=4)

```

```

x <- y <- seq(-10,10, 0.5)
simulated <- RFsimulate(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")

```

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 also be considered.
internal	logical. Flag whether internal models should also be 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>

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()
```

RFboxcox

Linear part of [RMmodel](#)

Description

`RFboxcox` performs the Box-Cox transformation: $\frac{(x+\mu)^\lambda - 1}{\lambda}$

Usage

```
RFboxcox(data, boxcox, vdim = 1, inverse=FALSE, ignore.na=FALSE)
```

Arguments

<code>data</code>	matrix or list of matrices.
<code>boxcox</code>	the one or two parameters (λ, μ) of the box cox transformation, in the univariate case; if μ is not given, then μ is set to 0. If not given, the globally defined parameters are used, see Details. In the m -variate case <code>boxcox</code> should be a $2 \times m$ matrix. If $\lambda = \infty$ then no transformation is performed.
<code>vdim</code>	the multivariate dimensionality of the field;
<code>inverse</code>	logical. Whether the inverse transformation should be performed.
<code>ignore.na</code>	logical. If FALSE an error message is returned if any value of <code>boxcox</code> is NA. Otherwise the data are returned without being transformed.

Details

The Box-Cox transformation `boxcox` can be set globally through `RFoptions`. If it is set globally the transformation applies in the **Gaussian** case to `RFfit`, `RFsimulate`, `RFinterpolate`, `RFvariogram`. Always first, the Box-Cox transformation is applied to the data. Then the command is performed. The result is back-transformed before returned.

If the first value of the transformation is `Inf` no transformation is performed (and is identical to `boxcox = c(1, 0)`). If `boxcox` has length 1, then the transformation parameter μ is set to 0, which is the standard case.

Value

`RFboxcox` returns a list of three components, `Y`, `X`, `vdim` returning the deterministic trend, the design matrix, and the multivariability, respectively. If `set` is positive, `Y` and `X` contain the values for the `set`-th set of coordinates. Else, `Y` and `X` are both lists containing the values for all the sets.

Author(s)

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

References

For the likelihood correction see

- Konishi, S., and Kitagawa, G. (2008) *Information criteria and statistical modeling*. Springer Science & Business Media. Section 4.9.

See Also

`Bayesian`, `RMmodel`, `RFsimulate`, `RFlikelihood`.

Examples

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

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)
)
dta <- soil["moisture"]

model <- ~1 + RMplus(RMwhittle(scale=NA, var=NA, nu=NA), RMnugget(var=NA))

## main Parameter in the Box Cox transformation to be estimated
```

```
print(fit <- RFfit(model, data=dta, boxcox=NA))
```

RFcov

(Cross-)Covariance function

Description

Calculates both the empirical and the theoretical (cross-)covariance function.

Usage

```
RFcov(model, x, y = NULL, z = NULL, T=NULL, grid, params, distances, dim, ...,
      data, bin=NULL, phi=NULL, theta = NULL, deltaT = NULL, vdim=NULL)
```

Arguments

model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
data	matrix, data.frame or object of class RFsp ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
bin	a vector giving the borders of the bins; If not specified an array describing the empirical (pseudo-)(cross-) covariance function in every direction is returned.
phi	an integer defining the number of sectors one half of the X/Y plane shall be divided 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 divided 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 <code>seq(from=0, to=deltaT[1], by=deltaT[2])</code>
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .

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.
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If params is given, then ... may include also the variables used in params.

Details

RFcov computes the empirical cross-covariance function for given (multivariate) spatial data.

The empirical (cross-)covariance function of two random fields X and Y is given by

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

where $t_{i,j} := t_i - t_j$, $N(r)$ denotes the number of pairs of data points with distancevector $t_{i,j} = r$ and where $m_X := \frac{1}{N(r)} \sum_{(t_i, t_j) | t_{i,j} = r} X_{t_i}$ and $m_Y := \frac{1}{N(r)} \sum_{(t_i, t_j) | t_{i,j} = r} Y_{t_i}$ denotes the mean of data points with distancevector $t_{i,j} = r$.

The spatial coordinates x , y , z should be vectors. For random fields of spatial dimension $d > 3$ write all vectors as columns of matrix x . In this case do neither use y , nor z and write the columns 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 **RFcov** to calculate the pseudocovariance function (see **RFoptions**).

Value

RFcov returns objects of class **RFempVariog**.

Author(s)

Jonas Auel; Sebastian Engelke; Johannes Martini; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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

RFvariogram, **RFmadogram**, **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, 0.02)
z <- RFsimulate(model, x=x, n=n)
emp.vario <- RFcov(data=z)
plot(emp.vario, model=model)

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

## space-time model
model <- RMnsst(phi=RMexp(), psi=RMfbm(alpha=1), delta=2)
x <- seq(0, 10, 0.05)
T <- c(0, 0.1, 100)
z <- RFsimulate(x=x, T=T, model=model, n=n)
emp.vario <- RFcov(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, 0.1)
z <- RFsimulate(model, x=x, y=x, n=n)
emp.vario <- RFcov(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, 0.1)
dta <- RFsimulate(model, x, x, n=n)
ev <- RFcov(data=dta, phi=4)
plot(ev, model=model, boundaries=FALSE)

```

RFcovmatrix	<i>Covariance matrix</i>
-------------	--------------------------

Description

`RFcovmatrix` returns the covariance matrix for a set of points;

Usage

```
RFcovmatrix(model, x, y = NULL, z = NULL, T = NULL, grid, params,
            distances, dim,...)
```

Arguments

model, params	object of class <code>RMmodel</code> , <code>RFformula</code> or <code>formula</code> ; best is to consider the examples below, first. The argument <code>params</code> is a list that specifies free parameters in a formula description, see <code>RMformula</code> .
x	vector of x coordinates, or object of class <code>GridTopology</code> or <code>raster</code> ; for more options see <code>RFsimulateAdvanced</code> .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually <code>grid</code> need not be given. See also <code>RFsimulateAdvanced</code> .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see <code>RFsimulateAdvanced</code> .
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by <code>RFoptions</code> . If <code>params</code> is given, then ... may include also the variables used in <code>params</code> .

Details

`RFcovmatrix` returns a covariance matrix. Here, a matrix of coordinates (x) or a vector or a matrix of distances is expected.

`RFcovmatrix` also allows for variogram models. Then the negative of the variogram matrix is returned.

Value

`RFcovmatrix` returns a covariance matrix.

Author(s)

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

See Also

[RMmodel](#), [RFsimulate](#), [RFfit](#), [RFfctn](#), [RFcalc](#), [RFCov](#), [RFpseudovariogram](#), [RFvariogram](#).

Examples

```
#####
# Example: 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)
```

RFcrossvalidate	<i>Fitting model parameters to spatial data (regionalised variables) and to linear (mixed) models</i>
-----------------	---

Description

The function estimates arbitrary parameters of a random field specification with various methods. Currently, the models 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

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

Arguments

model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
data	matrix, data.frame or object of class RFsp ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
lower	list or vector. Lower bounds for the parameters. If lower 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 lower has to be maintained. A component being NA means that no manual lower bound for the corresponding parameter is set. If lower is a list, lower has to be of (exactly) the same structure of the model.
upper	list or vector. Upper bounds for the parameters. See lower.
method	Single method to be used for estimating, either one of the methods or one of the sub.methods see RFfit
users.guess	User's guess of the parameters. All the parameters must be given using the same rules as for lower (except that no NA's should be contained).
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .
optim.control	control list for optim , which uses 'L-BFGS-B'. However parscale may not be given.
transform	obsolete for users; use param instead. <code>transform=list()</code> will return structural information to set up the correct function.
full	logical. If TRUE then cross-validation is also performed for intermediate models used in RFfit (if any).
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If params is given, then ... may include also the variables used in params.

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	<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.</p> <p>A method for summary returns summary statistics for the errors and standard errors similar to geoR.</p> <p>If <code>cross_refit = TRUE</code> and <code>detailed_output = TRUE</code> the returned object also contains a <code>fitted</code> which is a list of fitted models.</p>

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>

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
```

```
## currently disabled!
```

RFdistr

Evaluating distribution families

Description

Through [RRdistr](#) distribution families can be passed to **RandomFields** to create distributions available in the [RMmodel](#) definitions.

Usage

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

Arguments

model,params	an RRmodel .
x	the location where the density is evaluated
q	the location where 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>

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)))
}
```

RFempVariog-class

Class RFempVariog

Description

Class for RandomFields' representation of empirical variograms

Usage

```

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

```

Arguments

x	object of class RFempVariog
y	unused
model	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .. Or a list of such models. It gives the covariance or variogram models that are to be plotted into the same plot as the empirical variogram (and the fitted models)
nmax.phi	even integer; only for <code>class(x)=="RFempVariog"</code> ; the number of bins of angle phi that are to be plotted.
nmax.theta	integer; only for <code>class(x)=="RFempVariog"</code> ; the number of bins of angle theta that are to be plotted
nmax.T	integer; only for <code>class(x)=="RFempVariog"</code> ; the maximal number of different time bins that are to be plotted
plot.nbin	logical; only for <code>class(x)=="RFempVariog"</code> ; indicates whether the number of pairs per bin are to be plotted
plot.sd	logical; only for <code>class(x)=="RFempVariog"</code> ; indicates whether the calculated standard deviation (<code>x@sd</code>) is to be plotted (in form of arrows of length $\pm 1 * sd$)
method	character. Currently restricted to "ml" for maximum-likelihood method.
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 <code>class(x)=="RFempVariog"</code> 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 <code>boundaries=TRUE</code> 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 empirical variogram nor does the mean values need to be close the empirical variogram.
...	arguments to be passed to methods; mainly graphical arguments.

Slots

centers: the bin centres of the spatial distances
empirical: 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 empirical variograms only)
theta.centers: centres of the bins with respect to the second angle (for anisotropic empirical variograms in 3D only)
T: the bin centres of the time axis
vdim: the multivariate dimension
coordunits: string giving the units of the coordinates, see also option coordunits of [RFoptions](#).
varunits: 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
method: integer; variogram (0), covariance (2), madogram (4)

Methods

plot signature(x = "RFempVariog"): gives a plot of the empirical variogram, for more details see [plot-method](#).
plot signature(x = "RFempVariog", 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.
as signature(x = "RFempVariog"): converts into other formats, only implemented for target class [list](#).
show signature(x = "RFfit"): returns the structure of x
persp signature(obj = "RFempVariog"): 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, Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>

See Also

[RFvariogram](#), [plot-method](#)

Examples

```
# see 'RFvariogram'
```

 RFfctn

Evaluate Covariance and Variogram Functions

Description

[RFfctn](#) returns the values of a shape function.

Usage

```
RFfctn(model, x, y = NULL, z = NULL, T = NULL, grid, params, distances, dim,...)
```

Arguments

model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If params is given, then ... may include also the variables used in params.

Details

[RFcovmatrix](#) also allows for variogram models. Then the negative of the variogram matrix is returned.

Value

`RFfctn` returns a vector.

Author(s)

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

See Also

`RMmodel`, `RFsimulate`, `RFfit`, `RFcalc`, `RFcov`, `RFcovmatrix`, `RFpseudovariogram`, `RFvariogram`.

Examples

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

model <- RMexp() - 1
RFfctn(model, 1:10)
```

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. Currently, the models 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=NULL, data,
      lower = NULL, upper = NULL, methods,
      sub.methods, optim.control = NULL, users.guess = NULL,
      distances = NULL, dim, transform = NULL, params=NULL, ...)
```

Arguments

model, params	<p>object of class RMmodel, RFformula or formula; best is to consider the examples below, first.</p> <p>The argument params is a list that specifies free parameters in a formula description, see RMformula.</p> <p>All parameters that are set to NA will be estimated; see the examples below.</p> <p>Type RFgetModelNames(type="variogram") to get all options for model.</p>
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
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; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
data	<p>matrix, data.frame or object of class RFsp;</p> <p>If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.</p>
lower	<p>list or vector. Lower bounds for the parameters. If lower 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 lower has to be maintained. A component being NA means that no manual lower bound for the corresponding parameter is set.</p> <p>If lower is a list, lower has to be of (exactly) the same structure of the model.</p>
upper	list or vector. Upper bounds for the parameters. See lower.
methods	Main methods to be used for estimating. If several methods are given, estimation will be performed with each method and the results reported.
sub.methods	variants of the least squares fit of the variogram. 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 lower (except that no NA's should be contained).
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .
optim.control	control list for optim , which uses 'L-BFGS-B'. However parscale may not be given.
transform	obsolete for users; use param instead. transform=list() will return structural information to set up the correct function.
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If params is given, then ... may include also the variables used in params.

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 NAs 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 created. 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

- An important optional argument is [boxcox](#) which indicates a Box-Cox transformation; see [boxcox](#) in [RFoptions](#) and [RFboxcox](#) for details.
- Instead of [optim](#), other optimisers can be used, see [RFfitOptimiser](#).
- Several advanced options can be found in sections ‘General options’ and ‘fit’ of [RFoptions](#).
- In particular, [boxcox](#), [boxcox_lb](#), [boxcox_ub](#) allow Box-Cox transformation.
- 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>

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

[RFfitOptimiser](#), [RFlikelihood](#), [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")
```

```
#####
## simulate some data first                                     ##
points <- 100
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=100) #1000

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

#####
## coupling alpha and beta                                     ##
estmodel <- RMgencauchy(var=NA, scale=NA, alpha=NA, beta=NA) +
  RMtrend(NA)
RFfit(estmodel, data=d, transform = NA) ## just for information
trafo <- function(a) c(a[1], rep(a[2], 2))
fit <- RFfit(estmodel, data=d,
  transform = list(c(TRUE, TRUE, FALSE), trafo))
print(fit)
print(fit, full=TRUE)
```

RFfit-class

Class RFfit

Description

Class for RandomFields' representation of model estimation results

Usage

```
## S4 method for signature 'RFfit'
residuals(object, ..., method="ml", full=FALSE)
## S4 method for signature 'RFfit'
summary(object, ..., method="ml")
## S4 method for signature 'RFfit,missing'
plot(x, y, ...)
```

```
## S3 method for class 'RFfit'
contour(x, ...)
## S3 method for class 'RFempVariog'
contour(x, ...)

RFhessian(model)
```

Arguments

object	see the generic function;
...	<ul style="list-style-type: none"> • plot: arguments to be passed to methods; mainly graphical arguments, or further models in case of class CLASS_CLIST, see Details. • summary: see the generic function • contour : see RFplotEmpVariogram
method	character; only for <code>class(x)=="RFfit"</code> ; a vector of slot names for which the fitted covariance or variogram model is to be plotted; should be a subset of <code>slotNames(x)</code> for which the corresponding slots are of class CLASS_FIT; by default, the maximum likelihood fit ("ml") will be plotted
full	logical. if TRUE submodels are reported as well (if available).
x	object of class RFsp or RFempVariog 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	unused
model	<code>class(x)=="RF_fit"</code> or <code>class(x)=="RFfit"</code> , obtained from RFfit

Details

for the definition of plot see [RFplotEmpVariogram](#).

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 transformation has been estimated

coordunits: string giving the units of the coordinates, see also option `coordunits` of [RFoptions](#).

deleted: integer vector. Positions of the parameters that have been deleted to get the set of variables, used in the optimization.

ev: list; list of objects of class [RFempVariog](#), contains the empirical variogram estimates of the data

fixed: list of two vectors. The first 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
m1: 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 functions, e.g. `print.RMmodelFit`, use 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'
m1: RMmodelFit; analog to slot 'autostart'; with maximum likelihood method
v.proj: vector of integers. The components selected in one of the submodels
varunits: string giving the units of the variables, see also option `varunits` of `RFoptions`.
x.proj: logical or integer. If logical, 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 signature(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

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.

Further 'methods'

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

Author(s)

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

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](#), [RFvariogram](#), [RMmodel-class](#), [RMmodelFit-class](#), [plot-method](#).

Examples

```
# see RFfit
```


Description

See [RFfit](#) for a detailed description of the fitting procedure.

Details

Two parameters, see also [RFoptions](#) can be passed to [RFfit](#) that allow for choosing an optimiser different from `optim`:

`optimiser` takes one of the values "optim", "optimx", "soma", "nloptr", "GenSA", "minqa", "pso" or "DEoptim", see the corresponding packages for a description.

If `optimiser="nloptr"`, then the additional parameter `algorithm` must be given which takes the values "NLOPT_GN_DIRECT", "NLOPT_GN_DIRECT_L", "NLOPT_GN_DIRECT_L_RAND", "NLOPT_GN_DIRECT_NOSCAL", "NLOPT_GN_DIRECT_L_NOSCAL", "NLOPT_GN_DIRECT_L_RAND_NOSCAL", "NLOPT_GN_ORIG_DIRECT", "NLOPT_GN_ORIG_DIRECT_L", "NLOPT_LN_PRAXIS", "NLOPT_GN_CRS2_LM", "NLOPT_LN_COBYLA", "NLOPT_LN_NELDERMEAD", "NLOPT_LN_SBPLX", "NLOPT_LN_BOBYQA", "NLOPT_GN_ISRES", see **nloptr** for a description.

Author(s)

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

See Also

[RFfit](#), [RFoptions](#)

Examples

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

## Not run:
## Here some alternative optimisers to 'optim' are considered.
## All but the \pkg{nloptr} algorithms are largely slower than 'optim'.
## Only a few of them return results as good as 'optim'.

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)
)
dta <- soil["moisture"]
\dontshow{if (RFoptions())$internal$examples_red) {
```

```

warning("data have been reduced !")
All <- 1:7
rm(soil)
data(soil)
soil <- RFspatialPointsDataFrame(
  coords = soil[All, c("x.coord", "y.coord")],
  data = soil[All, c("moisture", "N03.N", "Total.N",
    "NH4.N", "DOC", "N20N")],
  RFparams=list(vdim=6, n=1)
)
dta <- soil["moisture"]
}}

model <- ~1 + RMwhittle(scale=NA, var=NA, nu=NA) + RMnugget(var=NA)
\dontshow{if (RFOptions()$internal$examples_red){model<~1+RMwhittle(scale=NA, var=NA, nu=1/2)}}
## standard optimiser 'optim'
print(system.time(fit <- RFfit(model, data=dta)))
print(fit)

opt <- "optimx" # 30 sec; better result
print(system.time(fit2 <- try(RFfit(model, data=dta, optimiser=opt))))
print(fit2)

\dontshow{\dontrun{
opt <- "soma" # 450 sec
print(system.time(fit2 <- try(RFfit(model, data=dta, optimiser=opt))))
print(fit2)
}}

opt <- "minqa" # 330 sec
print(system.time(fit2 <- try(RFfit(model, data=dta, optimiser=opt))))
print(fit2)

opt <- "nloptr"
algorithm <- RC_NLOPTR_NAMES
\dontshow{if(!interactive()) algorithm <- RC_NLOPTR_NAMES[1]}
for (i in 1:length(algorithm)) {
  print(algorithm[i])
  print(system.time(fit2 <- try(RFfit(model, data=dta, optimiser=opt,
    algorithm=algorithm[i])))
  print(fit2)
}

if (interactive()) {
## the following two optimisers are too slow to be run on CRAN.

opt <- "pso" # 600 sec
print(system.time(fit2 <- try(RFfit(model, data=dta, optimiser=opt))))
print(fit2)

```

```

opt <- "GenSA" # 10^4 sec
print(system.time(fit2 <- try(RFfit(model, data=dta, optimiser=opt))))
print(fit2)
}

## End(Not run)

```

RFformula

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

Description

It is described how to create a formula, which, for example, can be used as an argument of [RFsimulate](#) and [RFfit](#) to simulate and to fit data according to the model described by the formula.

In general, the created formula serves two purposes:

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

Thereby, fixed effects and trend surfaces can be addressed via the expression [RMfixed](#) and the function [RMtrend](#). In simple cases, the trend can also be given in a very simple, see the examples below. The covariance structures of the zero-mean multivariate normally distributed random field components are addressed by objects of class [RMmodel](#), which allow for a very flexible covariance specification.

See [RFformulaAdvanced](#) for rather complicated model definitions.

Details

The formula should be of the type

$$\text{response fixedeffects} + \text{errorterm}$$

or

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

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](#)).

- error term/nugget effect
optional, should be of the form `RMnugget(...)`. `RMnugget` describes a vector of iid Gaussian random variables.

IMPORTANT

Note that in formula constants are interpreted as part of a linear model, i.e. the corresponding parameter has to be estimated (e.g. $\sim 1 + \dots$) whereas in models not given as formula the parameters to be estimated must be given explicitly.

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>, <http://ms.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
M <- 3
model <- RMexp(var=V, scale=S) + M
```

```
x <- y <- seq(1, 3, 0.1)

simulated <- RFsimulate(model = model, x=x, y=y)
plot(simulated)

# an alternative code to the above code:
model <- ~ Mean + RMexp(var=Var, scale=Sc)
simulated2 <- RFsimulate(model = model,x=x, y=y, Var=V, Sc=S, Mean=M)
plot(simulated2)

# a third way of specifying the model using the argument 'param'
# the initials of the variables do not be captical letters
model <- ~ M + RMexp(var=var, scale=sc)
simulated3 <- RFsimulate(model = model,x=x, y=y,
                        param=list(var=V, sc=S, M=M))
plot(simulated3)

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

# compare sample mean of data with ML estimate, which is very similar:
mean(simulated@data[,1])
fitted
```

Description

Here examples for much more advanced formula are given

Note

NaN, in contrast to NA, signifies a unknown parameter that can be calculated from other (unknown) parameters.

Examples

```

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

#####
### the following definitions are needed in all the examples ###
#####
V <- 10
S <- 0.3
M <- 3
x <- y <- seq(1, 3, 0.1)

#####
###      Example 1: simple example      ###
#####

## the following to definitions of a model and call of RFsimulate
## give the same result:
model <- RMexp(var=V, scale=S) + M
z1 <- RFsimulate(model = model, x=x, y=y)
plot(z1)

model <- ~ M + RMexp(var=var, scale=sc)
p <- list(var=V, sc=S, M=M)
z2 <- RFsimulate(model = model,x=x, y=y, param=p)
plot(z2)

#####
###      Example 2: formulae within the parameter list      ###
#####

## free parameters (above 'var' and 'sc') can be used
## even within the definition of the list of 'param'eters
model <- ~ RMexp(var=var, sc=sc) + RMnugget(var=nugg)
p <- list(var=V, nugg= ~ var * abs(cos(sc)), sc=S) ## ordering does not matter!
z1 <- RFsimulate(model, x, y, params=p)
plot(z1)
RFgetModel(RFsimulate) ## note that V * abs(cos(S) equals 9.553365

## so the above is equivalent to
model <- ~ RMexp(var=var, sc=sc) + RMnugget(var=var * abs(cos(sc)))
z2 <- RFsimulate(model, x, y, params=list(var=V, sc=S))

```

```

plot(z2)

#####
### Example 3: formulae for fitting (i.e. including NAs) ###
#####
## first generate some data
model <- ~ RMexp(var=var, sc=sc) + RMnugget(var=nugg)
p <- list(var=V, nugg= ~ var * abs(cos(sc)), sc=S)
z <- RFsimulate(model, x, y, params=p, n=10)

## estimate the parameters
p.fit <- list(sc = NA, var=NA, nugg=NA)
print(f <- RFfit(model, data=z, params=p.fit))

## estimation with a given boundaries for the scale
p.fit <- list(sc = NA, var=NA, nugg=NA)
lower <- list(sc=0.01)
upper <- list(sc=0.02)
print(f <- RFfit(model, data=z, params=p.fit, lower = lower, upper = upper))

#####
### Example 4 (cont'd Ex 3): formulae with dummy variables ###
#####
V <- 10
S <- 0.3
M <- 3
x <- y <- seq(1, 3, 0.1)

model <- ~ RMexp(sc=sc1, var=var) + RMgauss(var=var2, sc=sc2) +
  RMdeclare(u) ## introduces dummy variable 'u'
p.fit <- list(sc1 = NA, var=NA, var2=~2 * u, sc2 = NA, u=NA)
lower <- list(sc1=20, u=5)
upper <- list(sc2=1.5, sc1=100, u=15)
print(f <- RFfit(model, data=z, params=p.fit, lower = lower, upper = upper
))

```

Description

The function estimates the fractal dimension of a process

Usage

```
RFfractalDIM(x, y = NULL, z = NULL, data, grid,
  bin=NULL,
  vario.n=5,
  sort=TRUE,
  fft.m = c(65, 86), ## in % of range of l.lambd
  fft.max.length=Inf,
  fft.max.regr=150000,
  fft.shift = 50, # in %; 50:WOSA; 100: no overlapping
  method=c("variogram", "fft"),
  mode = if (interactive ()) c("plot", "interactive") else "nographics",
  pch=16, cex=0.2, cex.main=0.85,
  printlevel = Rfoptions()$basic$printlevel,
  height=3.5,
  ...)
```

Arguments

x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced . If x is not given and data is not an sp object, a grid with unit grid length is assumed
y,z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
data	the values measured; it can also be an sp object
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
bin	sequence of bin boundaries for the empirical variogram
vario.n	first vario.n values 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 <code>fft.max.length</code> . For each piece the FFT is calculated and then the average for all pieces is taken. The pieces may overlap, see the argument <code>fft.shift</code> .
fft.max.regr	If the <code>fft.m</code> 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 <code>fft.m</code> is less than <code>fft.max.regr</code> .
fft.shift	This argument is given in percent [of <code>fft.max.length</code>] and defines the overlap of the pieces defined by <code>fft.max.length</code> . If <code>fft.shift=50</code> the WOSA estimator is given; if <code>fft.shift=100</code> no overlap exists.
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 this 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.

<code>pch</code>	vector or scalar; sign by which data are plotted.
<code>cex</code>	vector or scalar; size of <code>pch</code> .
<code>cex.main</code>	The size of the title in the regression plots.
<code>printlevel</code>	integer. If <code>printlevel</code> is 0 nothing is printed. If <code>printlevel=1</code> error messages are printed. If <code>printlevel=2</code> warnings and the regression results are given. If <code>printlevel>2</code> tracing information is given.
<code>height</code>	height of the graphics window
<code>...</code>	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.

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

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, 0.001)
z <- RFsimulate(RMexp(), x)
RFfractaldim(data=z)
```

RFfunction

Evaluation operators (RF commands)

Description

Here, all the RF_name_ commands are listed.

Functionals of RMmodels

The user's [RMmodel](#) is supplemented internally by operators that are tacitly 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 representation as completion to the user's model.

RFcalc	performs some simple calculations based on R.models
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
RFdistr	generic function assigning to a distribution family various values of the distribution
RFfctn	assigns to a model the value of the function at given locations. In case of a covariance model RFfctn i
RFlikelihood	assigns to a model and a dataset the (log)likelihood value.
RFlinearpart	assigns to a model and a set of coordinates the linear part of the model, i.e. the deterministic trend and
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

RFCrossvalidate	cross validation for Gaussian fields
RFvariogram	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 RFunction , 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>

See Also

[RC](#), [RM](#), [RP](#), [RR](#), [R.](#), [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 method string_`). 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 guaranteed 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 guaranteed for further covariance models, e.g. the whittle matern model. In fact, the circulant embedding is called with the cutoff hypermodel, see [RMmodel1](#), and $A = B$ there. cutoff embedding halves 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 guaranteed 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 [RMmodel1](#), and $A = B$ there.
Here, multiplicative models are not allowed (yet).
- **Marked point processes.**
Some methods are based on marked point process $\Pi = \cup [x_i, m_i]$ 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. This method is called automatically if the nugget effect is positive except the method "circulant embedding" or "direct" has been explicitly chosen.
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, a 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 `RFvariogram()` shows a way of checking the precision.

Author(s)

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

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()

RFgetModel	<i>Internally stored model</i>
------------	--------------------------------

Description

The function returns the stored model.

Usage

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

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 or character. 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.
origin	If show.call is a character it behaves as which.submodels . character; one of "original", "MLE conform", "all". This argument determines the parameters that are returned.

Details

Whereas [RFgetModel](#) returns a model that can be re-used by the user, [RFgetModelInfo](#) can return detailed information.

Value

The stored model is returned in list format.

Note

Put `Storing=TRUE`, see [RFoptions](#), if you like to have (more) internal information in case of failure of an initialization of a random field simulation.

Author(s)

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

See Also

[RFgetModelInfo](#), [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)
RFgetModel(show.call=FALSE)
RFgetModel(show.call=TRUE)
```

RFgetModelInfo

Information on RMmodels

Description

The function returns information about an [RMmodel](#), either internal information when used in simulations, for instance, or general information

Usage

```
RFgetModelInfo(...)
```

```
RFgetModelInfo_register(register, level = 1, spConform =
  RFoptions()$general$spConform, which.submodels =
  c("user", "internal", "call+user", "call+internal",
    "user.but.once", "internal.but.once",
    "user.but.once+jump", "internal.but.once+jump", "all"),
  modelname = NULL, origin = "original")
```

```
RFgetModelInfo_model(model, params, dim = 1, Time = FALSE,
  kernel = FALSE, exclude_trend = TRUE, ...)
```

Arguments

...	See the argument of <code>RFgetModelInfo_register</code> and <code>RFgetModelInfo_model</code> ; <code>RFgetModelInfo</code> is an abbreviation for the other two functions.
register	0, ..., 21 or an evaluating function, e.g. RFsimulate . Place where intermediate calculations are stored. See also section Registers in RFoptions .
level	integer [0...5]; level of details, i.e. the higher the number the more details are given.
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. Most models have a leading internal model. The values “call+user” and “call+internal” also return this leading model if existent.

	<p>The values "user.but.once", "internal.but.once" "user.but.once" returns the user path of the internal model following the leading model. "internal.but.once" would return the internal path of the user model following the leading model, but this path should never exist. So as all the other options if a certain direction does not exist, the alternative path is taken.</p> <p>The values "user.but.once+jump", "internal.but.once+jump" same as "user.but.once" and "internal.but.once", except that the first submodel below the leading model is not given.</p> <p>The value "all" returns the whole tree of models (very advanced).</p>
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.
model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula . Here, NAs should be placed where information on the parameters is desired..
dim	positive integer. Spatial dimension.
Time	logical. Should time be considered, too?
kernel	logical. Should the model be considered as a kernel?
exclude_trend	logical. Currently, only TRUE is available.
origin	character; one of "original", "MLE conform", "all". This argument determines the parameters that are returned.

Details

RFgetModelInfo branches either into RFgetModelInfo_register or RFgetModelInfo_model, depending on the type of the *first* argument. The latter two are usually not called by the user.

RFgetModelInfo has three standard usages:

- RFgetModelInfo() returns internal information on the last call of an [RF](#) function.
- RFgetModelInfo(RFfunction) returns internal information on the last call of [RFfunction](#).
- RFgetModelInfo(RMmodel) returns general information on [RMmodel](#)

Whereas RFgetModelInfo() can return detailed internal information, [RFgetModel](#) returns a model that can be re-used by the user.

Value

If RFgetModelInfo(model) is called a list is returned with the following elements:

- trans.inv : logical. Whether the model is translation invariant (stationary)
- isotropic : logical. Whether the model is rotation invariant (stationary)
- NAs : in case of an additive model it gives the number of NAs in each submodel
- minmax : a data frame containing information on all arguments set to NAs

- pmin, pmax : lower and upper endpoint of the parameter values usually found in practice
- type : integer; recognized particularities of a parameter; an explanation of the values is given after the table, if printed.
- NAN : the number of NANs found
- min, max : mathematically valid lower and upper endpoints of the parameter values
- omin, omax : logical. If FALSE the respective mathematical endpoint is included
- col, row : the dimension of the parameter. If the parameter is a scalar then col = row = 1. If it is a vector then col = 1.
- bayes : currently not used (always FALSE)

Else a list of internal structure is returned.

Note

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

Author(s)

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

See Also

command [RFgetModel](#), [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()

model <- RMwhittle(scale=NA, var=NA, nu=NA) + RMnugget(var=NA)
RFgetModelInfo(model)
```

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", "variogram", 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_NAMES, domain = RC_DOMAIN_NAMES,
  isotropy = RC_ISO_NAMES, operator = c(TRUE, FALSE),
  monotone = RC_MONOTONE_NAMES,
  implied_monotonocities = length(monotone) == 1,
  finiterange = c(TRUE, FALSE, NA),
  valid.in.dim = c(1, Inf),
  vdim = c(1, 5),
  group.by,
  exact.match = !missing(group.by),
  simpleArguments = FALSE,
  internal, newnames)
```

Arguments

- `type`, `domain`, `isotropy`, `operator`, `monotone`, `finiterange`, `vdim`
 see [constants](#) for the definition of `RC_TYPE_NAMES`, `RC_DOMAIN_NAMES`, etc. See also [RMmodelgenerator](#).
- `implied_monotonocities`
 logical. If TRUE then all the models with a stronger monotonicity 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 or NULL; must be one of 'type', 'domain', 'isotropy', 'operator', 'monotone', 'finiterange', 'maxdim', 'vdim'. If `group.by` is not given, the result is grouped by 'type' if more than one type is given.
- `exact.match` logical. If not TRUE, then all categories that are subclasses or might match are shown as well.
- `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 RModelgenerator.

The following arguments can be specified. In general, only exact matches are returned. One exception exists: If the length of type equals 1 and if group.by is not given, then types included in type are also returned. E.g. if type="variogram" and group.by is not given then only models are returned that are negative definite. However, also positive definite functions and tail correlaton functions are returned if "type" is included in group.by.

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

stationarity specifies the type of stationarity; for the meaning of the possible values see RModelgenerator

isotropy specifies the type of isotropy; for the meaning of the possible values see RModelgenerator

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

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 RModelgenerator

finiterange indicates whether the covariance of the model has finite range, e.g. RMcircular or RMnugget have covariances with finite range; see RModelgenerator. NA is used if the finiteness depends on the submodel.

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 a 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 RModel object; maxdim=-2 means that the maximal possible dimension is adopted from the called submodels; see also RModelgenerator

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 RModel object; vdim=-2 means that being multivariate in a certain dimension is adopted from the called submodels; see also RModelgenerator

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 RModelgenerator.

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 `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 to 2,
- `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>

See Also

[constants](#), [RMmodelgenerator](#), [RMmodel](#), [RandomFields](#), [RC_DOMAIN_NAMES](#), [RC_ISO_NAMES](#)

Examples

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

# get list of names of all functions
RFgetModelNames()

# any kind of positive definite functions
RFgetModelNames(type="positive definite", exact.match=TRUE)
## Not run: RFgetModelNames(type="positive definite")
```

```

# get a list of names of all stationary models
RFgetModelNames(type="positive definite", domain="single variable",
                 exact.match=TRUE)
## Not run: RFgetModelNames(type="positive definite", domain="single variable")

# get a vector of all model names
RFgetModelNames(group.by=NULL)

```

RFgridDataFrame-class *Class* RFgridDataFrame

Description

Class for attributes in one-dimensional space.

Usage

```

## S4 method for signature 'RFgridDataFrame'
RFspDataFrame2conventional(obj, data.frame=FALSE)

```

Arguments

`obj` an RFgridDataFrame object
`data.frame` logical. If TRUE a data.frame is returned.

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,
- `coordunits` gives the names of the units for the coordinates
- `varunits` 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
- range** signature(x = "RFgridDataFrame"): returns the range
- hist** signature(x = "RFgridDataFrame"): plots histogram
- 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
- as.data.frame** signature(x = "RFgridDataFrame"): converts data-slot and coordinates to a data.frame

Details

Methods summary and dimensions are defined for the “parent”-class [RFsp](#).

Author(s)

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

See Also

[RFspatialGridDataFrame](#), which is for point locations in higher dimensional spaces, [RFpointsDataFrame-class](#) which is for one-dimensional arbitrary locations, [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))
head(coordinates(f))
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, bin = NULL, xcov, ycov,
      sim_only1dim=FALSE, wait = 0, ...)
```

Arguments

<code>data</code>	See RFvariogram . If data is given, the empirical variogram is shown.
<code>x</code>	a sequence of the locations of the simulated process; if not given, <code>x</code> is determined by data and if data is not given by default values
<code>y</code>	a sequence of numbers if a simulation on R^d is performed. Default is <code>y = x</code> ; see <code>x</code> for details.
<code>same.algorithm</code>	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.
<code>ev</code>	instead of the data, the empirical variogram itself might be passed
<code>bin</code>	only considered if data is given. See RFvariogram for details.
<code>xcov</code>	sequence of the locations where the covariance function is plotted
<code>ycov</code>	Only for anisotropic models. sequence of the locations where the covariance function is also plotted
<code>sim_only1dim</code>	Logical. The argument determines whether a process should be simulated on the line or on the plane
<code>wait</code>	integer. See details.
<code>...</code>	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 returns `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 gets slower as well. Reasonable values for `wait` are within `[0, 1000]`.

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

Value

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

If `wait < 0`, a side 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>

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

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 = if (interactive ()) c("plot", "interactive") else "nographics",
  pch = 16, cex = 0.2, cex.main = 0.85,
  printlevel = RFOptions()$basic$printlevel, height = 3.5,
  ...)
```

Arguments

x	vector of x coordinates, or object of class <code>GridTopology</code> or <code>raster</code> ; for more options see RFsimulateAdvanced .
y,z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
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 are 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 <code>fft.max.length</code> then the segments of length <code>fft.max.length</code> are considered, shifted by <code>fft.max.length/2</code> (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 <code>c("plot", "interactive")</code> 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 <code>lm</code> .
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 <code>lm</code> 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>

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)
h <- RFhurst(1:length(x), data=x)
```

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=NULL,
              distances, dim, data, given=NULL, params, err.model, err.params,
              ignore.trend = FALSE, ...)
```

Arguments

<code>model, params</code>	object of class <code>RMmodel</code> , <code>RFformula</code> or <code>formula</code> ; best is to consider the examples below, first. The argument <code>params</code> is a list that specifies free parameters in a formula description, see <code>RMformula</code> .
<code>x</code>	vector of x coordinates, or object of class <code>GridTopology</code> or <code>raster</code> ; for more options see <code>RFsimulateAdvanced</code> .
<code>y, z</code>	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
<code>T</code>	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
<code>grid</code>	logical; the function finds itself the correct value in nearly all cases, so that usually <code>grid</code> need not be given. See also <code>RFsimulateAdvanced</code> .
<code>distances, dim</code>	another alternative for the argument x to pass the (relative) coordinates, see <code>RFsimulateAdvanced</code> .
<code>data</code>	matrix, data.frame or object of class <code>RFsp</code> ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest. If given is not given and data is a matrix or data is a data.frame, RandomFields tries to identify where the data and the coordinates are, e.g. by names in formulae or by fixed names, see <code>Coordinate systems</code> . See also <code>RFsimulateAdvanced</code> . If all fails, the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field. Notes that also lists of data can be passed. If the argument x is missing, data may contain NAs, which are then replaced through imputing.
<code>given</code>	optional, matrix or list. If given matrix then the coordinates can be given separately, namely by given where, in each row, a single location is given. If given is a list, it may consist of x, y, z, T, grid. If given is provided, data must be a matrix or an array containing the data only.
<code>err.model, err.params</code>	For conditional simulation and random imputing only. In case of (assumed) error-free measurements (which is mostly the case in geostatistics) the argument <code>err.model</code> is not given. In case of measurement errors we have <code>err.model=RMnugget(var=var)</code> . <code>err.param</code> plays the same role as <code>params</code> for <code>model</code> .
<code>ignore.trend</code>	logical. If TRUE only the covariance model of the given model is considered, without the trend part.
<code>...</code>	for advanced use: further options and control arguments for the simulation that are passed to and processed by <code>RFoptions</code> . If <code>params</code> is given, then <code>...</code> may include also the variables used in <code>params</code> .

Details

In case of repeated data, they are kriged *separately*; if the argument x is missing, data may contain NAs, which are then replaced by the kriged values (imputing);

In case of intrinsic cokriging (intrinsic kriging for 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>;
 Marco Oesting, <oesting@mathematik.uni-siegen.de>, <https://www.uni-siegen.de/fb6/src/scheffler/mitarbeiter/oesting>

Author(s) of the code:: Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>; Alexander Malinowski; Marco Oesting, <oesting@mathematik.uni-siegen.de>, <https://www.uni-siegen.de/fb6/src/scheffler/mitarbeiter/oesting>

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](#), [RFvariogram](#), [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
dev.new(height=7, width=16)

## 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() + MTrend(mean=1)
dta <- RFsimulate(model, x=points)
plot(dta)
x <- seq(0, 9, 0.25)

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

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

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

## Co-Kriging
```



```
n <- 100
x <- runif(n=n, min=1, max=50)
y <- runif(n=n, min=1, max=50)

rho <- matrix(nc=2, c(1, -0.8, -0.8, 1))
model <- RMparswmX(nudiag=c(0.5, 0.5), rho=rho)

## generation of artificial data
data <- RFsimulate(model = model, x=x, y=y, grid=FALSE)
## introducing some NAs ...
print(data)
len <- length(data)
data@data$variable1[1:(len / 10)] <- NA
data@data$variable2[len - (0:len / 100)] <- NA
print(data)
plot(data)

## co-kriging
x <- y <- seq(0, 50, 1)

k <- RFinterpolate(model, x=x, y=y, data= data)
plot(k, data)

## conditional simulation
z <- RFsimulate(model, x=x, y=y, data= data) ## takes some time
plot(z, data)

close.screen(all = TRUE)
```

RFlinearpart

Linear part of [RMmodel](#)

Description

[RFlinearpart](#) returns the linear part of a model

Usage

```
RFlinearpart(model, x, y = NULL, z = NULL, T = NULL, grid=NULL,
```

data, params, distances, dim, set=0, ...)

Arguments

model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .
data	matrix, data.frame or object of class RFsp ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
set	integer. See section Value for details.
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If params is given, then ... may include also the variables used in params.

Value

[RFlinearpart](#) returns a list of three components, Y, X, vdim returning the deterministic trend, the design matrix, and the multivariability, respectively. If set is positive, Y and X contain the values for the set-th set of coordinates. Else, Y and X are both lists containing the values for all the sets.

Note

In the linear part of the model specification the parameters that are NA must be the first model part. I.e. `NA * sin(R.p(new="isotropic")) + NA + R.p(new="isotropic")` is OK, but not `sin(R.p(new="isotropic")) * NA + NA + R.p(new="isotropic")`

Author(s)

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

See Also

[Bayesian](#), [RMmodel](#), [RFsimulate](#), [RFlikelihood](#).

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, pi, len=10)
trend <- 2 * sin(R.p(new="isotropic")) + 3
model <- RMexp(var=2, scale=1) + trend
print(RFlinearpart(model, x=x)) ## only a deterministic part

trend <- NA * sin(R.p(new="isotropic")) + NA + R.p(new="isotropic") / pi
model <- RMexp(var=NA, scale=NA) + trend
print(RFlinearpart(model, x=x))

```

RFloglikelihood

Likelihood and estimation of linear models

Description

[RFloglikelihood](#) returns the log likelihood for Gaussian random fields. In case NAs are given that refer to linear modeling, the ML of the linear model is returned.

Usage

```

RFlikelihood(model, x, y = NULL, z = NULL, T = NULL, grid = NULL,
             data, params, distances, dim, likelihood,
             estimate_variance =NA, ...)

```

Arguments

model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .

data	matrix, data.frame or object of class RFsp ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
likelihood	Not programmed yet. Character. Choice of kind of likelihood ("full", "composite", etc.), see also likelihood for RFfit in RFoptions .
estimate_variance	logical or NA. See Details.
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If <code>params</code> is given, then ... may include also the variables used in <code>params</code> .

Details

The function calculates the likelihood for data of a Gaussian process with given covariance structure. The covariance structure may not have NA values in the parameters except for a global variance. In this case the variance is returned that maximizes the likelihood. Additional to the covariance structure the model may include a trend. The latter may contain unknown linear parameters. In this case again, the unknown parameters are estimated, and returned.

Value

[RFloglikelihood](#) returns a list containing the likelihood, the log likelihood, and the global variance (if estimated – see details).

Author(s)

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

See Also

[Bayesian](#), [RMmodel](#), [RFfit](#), [RFsimulate](#), [RFlinearpart](#).

Examples

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

requireNamespace("mvtnorm")

pts <- 4
repet <- 3
model <- RMexp()
x <- runif(n=pts, min=-1, max=1)
y <- runif(n=pts, min=-1, max=1)
dta <- as.matrix(RFsimulate(model, x=x, y=y, n=repet, spC = FALSE))
print(cbind(x, y, dta))
print(system.time(likeli <- RFlikelihood(model, x, y, data=dta)))
```

```

str(likeli, digits=8)

L <- 0
C <- RFcovmatrix(model, x, y)
for (i in 1:ncol(dta)) {
  print(system.time(dn <- mvtnorm::dmvnorm(dta[,i], mean=rep(0, nrow(dta)),
sigma=C, log=TRUE)))
  L <- L + dn
}
print(L)
stopifnot(all.equal(likeli$log, L))

pts <- 4
repet <- 1
trend <- 2 * sin(R.p(new="isotropic")) + 3
#trend <- RMtrend(mean=0)
model <- 2 * RMexp() + trend
x <- seq(0, pi, len=pts)
dta <- as.matrix(RFsimulate(model, x=x, n=repet, spC = FALSE))
print(cbind(x, dta))

print(system.time(likeli <- RFlikelihood(model, x, data=dta)))
str(likeli, digits=8)

L <- 0
tr <- RFfctn(trend, x=x, spC = FALSE)
C <- RFcovmatrix(model, x)
for (i in 1:ncol(dta)) {
  print(system.time(dn <- mvtnorm::dmvnorm(dta[,i], mean=tr, sigma=C, log=TRUE)))
  L <- L + dn
}
print(L)

stopifnot(all.equal(likeli$log, L))

pts <- c(3, 4)
repet <- c(2, 3)
trend <- 2 * sin(R.p(new="isotropic")) + 3
model <- 2 * RMexp() + trend
x <- y <- dta <- list()
for (i in 1:length(pts)) {
  x[[i]] <- list(x = runif(n=pts[i], min=-1, max=1),

```

```

        y = runif(n=pts[i], min=-1, max=1))
    dta[[i]] <- as.matrix(RFsimulate(model, x=x[[i]]$x, y=x[[i]]$y,
                                   n=repet[i], spC = FALSE))
}

print(system.time(likeli <- RFlikelihood(model, x, data=dta)))
str(likeli, digits=8)

L <- 0
for (p in 1:length(pts)) {
  tr <- Rffctn(trend, x=x[[p]]$x, y=x[[p]]$y, spC = FALSE)
  C <- RFCovmatrix(model, x=x[[p]]$x, y=x[[p]]$y)
  for (i in 1:ncol(dta[[p]])) {
    print(system.time(dn <- mvtnorm::dmvnorm(dta[[p]][,i], mean=tr, sigma=C,
                                             log=TRUE)))

    L <- L + dn
  }
}
print(L)
stopifnot(all.equal(likeli$log, L))

```

RFmadogram

Empirical (Cross-)Madogram

Description

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

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

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

Usage

```
RFmadogram(model, x, y=NULL, z=NULL, T=NULL, grid, params, distances,
           dim, ..., data, bin=NULL, phi=NULL, theta = NULL,
           deltaT = NULL, vdim=NULL)
```

Arguments

`model, params` object of class `RMmodel`, `RFformula` or `formula`; best is to consider the examples below, first.
The argument `params` is a list that specifies free parameters in a formula description, see `RMformula`.

x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually <code>grid</code> need not be given. See also RFsimulateAdvanced .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If <code>params</code> is given, then ... may include also the variables used in <code>params</code> .
data	matrix, <code>data.frame</code> or object of class RFsp ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
bin	a vector giving the borders of the bins; If not specified an array describing the empirical (pseudo-)(cross-) covariance function in every direction is returned.
phi	an integer defining the number of sectors one half of the X/Y plane shall be divided into. If not specified, either an array is returned (if <code>bin</code> missing) or isotropy is assumed (if <code>bin</code> specified).
theta	an integer defining the number of sectors one half of the X/Z plane shall be divided into. Use only for dimension $d = 3$ if <code>phi</code> is already specified.
deltaT	vector of length 2, specifying the temporal bins. The internal bin vector becomes <code>seq(from=0, to=deltaT[1], by=deltaT[2])</code>
vdim	the number of variables of a multivariate data set. If not given and <code>data</code> is an RFsp object created by RandomFields , the information there is taken from there. Otherwise <code>vdim</code> is assumed to be one. NOTE: still the argument <code>vdim</code> is an experimental stage.

Details

[RFmadogram](#) computes the empirical cross-madogram 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 columns of matrix x. In this case do neither use y, nor z and write the columns 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 [RFmadogram](#) to calculate the pseudomadogram (see [RFoptions](#)).

Value

[RFmadogram](#) returns objects of class [RFempVariog](#).

Author(s)

Jonas Auel; Sebastian Engelke; Johannes Martini; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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](#), [RMmodel](#), [RFsimulate](#), [RFfit](#), [RFCov](#), [RFpseudomadogram](#), [RFvariogram](#).

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, 0.02)
z <- RFsimulate(model, x=x, n=n)
emp.vario <- RFmadogram(data=z)
plot(emp.vario)
```

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

```
## space-time model
model <- RMnsst(phi=RMexp(), psi=RMfbm(alpha=1), delta=2)
x <- seq(0, 10, 0.05)
T <- c(0, 0.1, 100)
z <- RFsimulate(x=x, T=T, model=model, n=n)
emp.vario <- RFmadogram(data=z, deltaT=c(10, 1))
plot(emp.vario, 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, 0.1)
```



```
z <- RFsimulate(model, x=x, y=x, n=n)
emp.vario <- RFmadogram(data=z)
plot(emp.vario)

## 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, 0.1)
dta <- RFsimulate(model, x, x, n=n)
ev <- RFmadogram(data=dta, phi=4)
plot(ev, boundaries=FALSE)
```

RFoldstyle

RFoldstyle

Description

This function 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>

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. It expands the functionality of **RFoptions**.

Usage

```
RFoptions(...)
```

Arguments

... arguments in tag = value form, or a list of tagged values.

Details

The subsections below comment on

0. basic: See **RFoptions**
1. general: **General options**
2. br: **Options for Brown-Resnick Fields**
3. circulant: **Options for circulant embedding methods** [RPsirculant](#)
4. coords: **Options for coordinates and units**, see [coordinate systems](#)
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. solve: **Options for solving linear systems**
- 20. special: **Options for some special methods**
- 21. spectral: **Options for the spectral (turning bands) method**
- 22. tbm: **Options for the turning bands method**
- 23. internal: **Internal**

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. Standard users will never use a value higher than 3.

- 0 : no messages
- 1 : messages and warnings when the user's input looks odd
- 2 : messages (and internal errors) documenting the choice of the simulation method
- 3 : further user relevant informations
- 4 : information on recursive function calls
- 5 : function flow information of central 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 everyth 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. Currently only used when simulating Gaussian random fields.

- TRUE: `RPcoins`, `RPhyperplane`, `RPsequential`, `RPspectral` and `RPtbm` and *approximative* circulant embedding are excluded. If the circulant embedding method is considered as badly behaved, then the matrix decomposition methods are preferred.
- FALSE: all the methods are allowed. 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: Similar to FALSE, but some inexact algorithms get less preference.

Default: NA .

`expected_number_simu` positive integer which is usually 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 lie 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 option being set to `FALSE` after calling `RFoptions` it should be set as first element of a list.

Default: TRUE

`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 high 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. (Only used for kriging – estimation can fully deal with NAs.)

Default: FALSE.

`output` character. one of the values "sp" (if and only if `spConform=TRUE`), "RandomFields" (if and only if `spConform=FALSE`), "geoR".

The output mode `geoR` currently adds some attributes such as the call of the function.

NOTE: `output` is in an experimental stage, whose effects might change in future. Currently, `output` changes the values of `reportcoord`, `returncall` and `spConform`.

`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=` for the list of primitive models.

- FALSE : the practical range adjustment 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 $\text{printlevel} \leq 0$ there is not any output on the screen. The higher the number the more tracing information is given. Standard users will never use a value higher than 3.

- 0 : no messages
- 1 : important (error) messages and warnings
- 2 : less important messages
- 3 : details, but still for the user
- 4 : recursive call tracing (only used within `RFfit`)
- 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

reportcoord character. Current values are "always", "important", "warn", "never",

Both "warn" and "important" have any effect only if the coordinate system is changed internally. In this case "warn" yields a displayed warning message whereas "important" adds an attribute to the result as in the case "always".

If "always" or "important" the reports are added as attribute to the results. Note that in this case the class of the result may change (e.g. from "numeric" to "atomic").

Default: "warn"

returncall logical. If TRUE then the call is returned as an attribute

Default: TRUE

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 i th simulation is started with the seed 'seed+i - 1'.

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

seed_incr, seed_sub_incr (does not work yet) This argument is important iff `RFsimulate` is used within a function from package **parallel**. The value of `seed_incr` should be set only locally, i.e. not by `RFoptions()`.

If `seed_incr != 0` (or the number of simulations n is greater than 1) and `!is.na(seed)` then the seed for each simulation is calculated as

$$\text{seed} + (k - 1) * \text{seed_sub_incr} + \text{seed_incr} * n$$

where k runs from 1 to n .

Default: 0

set integer. Certain models (e.g. `RMfixcov` and `RMcovariate`) allow for lists as arguments. `set` selects a certain list element. If necessary the list is recycled.

`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 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.

See also `output`.

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 occurred. 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=FALSE`.

Default: `FALSE`.

2. Options for Brown-Resnick Fields

`deltaAM` integer; only used for simulation of BR processes via `RPbrmixed` with `optim_mixed=2`.

In this case, `deltaAM` is the number of additionally simulated Gaussian processes used for an update of `areamat` in the optimization procedure.

Default: `300`

`maxtrendmem` integer; the maximal number of real valued variables used for intermediate storage:

- `RPbrshifted`: trends for shifted locations that may be stored at the same time when simulating BR processes.
- `RPbrnormed`: Let n be the number of locations. Then a $n \times n$ (covariance) matrix has to be evaluated at random columns.

if `maxtrendmem` is large (and n small, $n \leq 10^4$), multiple evaluations can be avoided.

Default: `1e7`.

`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 `RPbrmixed` 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_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-divided into 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 `RPtbn` 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

coord_system character. See [coordinate systems](#)
 coordunits See [coordinate systems](#)
 coordnames See [coordinate systems](#)
 new_coord_system See [coordinate systems](#)
 new_coordunits See [coordinate systems](#)
 polar_coord See [coordinate systems](#)
 varnames See [coordinate systems](#)
 varunits See [coordinate systems](#)
 xyz_notation See [coordinate systems](#)
 zenit See [coordinate systems](#)

5. direct: Options for simulating by simple matrix decomposition

max_variab Maximal size of the covariance matrix.
 Default: 12000

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 the starting angle.

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` determines the starting value of the second angle in 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 `Rfit`, `Rfratiotest`, and `RFcrossvalidate`

`algorithm` See `RfitOptimiser`.

Default: NULL

`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

`boxcox_lb` lower bound for the Box-Cox transformation

Default: -10.

`boxcox_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.

estimate_variance see [RFlikelihood](#).

factr, factr_recall See the argument control in [optim](#). `factr_recall` is used for intermediate calculations.

likelihood character – not programmed yet. types of likelihood are "auto", "full", "composite", "tesselation";

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_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 OBSOLETE. 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 variogram 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 variogram 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** See [RFfitOptimiser](#).
Default: "optim".
- pgtol, pgtol_recall** See the argument control in [optim](#). **pgtol_recall** is used for intermediate calculations.
- 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

- `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 $(\text{minimumdistancebetweendifferentpairsofpoints}) / \text{scale_max_relative_factor}$ 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 $(\text{minimumdistancebetweendifferentpairsofpoints}) / \text{lowerbound_scale_ls_factor}$.
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.
- `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` integer. If the number of parameters to be numerically optimised is larger than or equal to `split` 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: 4.
- `cliquesize` integer. `Rffit` tries to split the data set into parts of size `splitn_neighbours[2]` or less, but never more than `splitn_neighbours[3]` and never less than `splitn_neighbours[1]`.
Default: `c(200, 1000, 3000)`.
- `splitfactor_neighbours` The total number of neighbouring boxes in each direction $1+2\text{splitfactor}$, 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 $\text{upperbound_scale_factor} * (\text{maximum distance between all pairs of points})$.
Default: 3.
- `upperbound_var_factor` The upper bound for the variance and the nugget is determined as $\text{upperbound_var_factor} * \text{var}(\text{data})$
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 $\text{cov}(1) \approx 0.05$. `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`.
- note the `use_naturalscaling` only affects simple models, no operators. Also functions that define a parameter of the model are not changed.

Default: FALSE.

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

`boxcox` real vector of one or two components. If the first component is Inf then no transformation is performed. Otherwise the BoxCox transformation is performed. Note that Box Cox only works in a Gaussian framework. Note further that either `boxcox` or `loggauss` may be given.

Default $c(\text{Inf}, 0)$

`direct_bestvar` integer. When searching for an appropriate simulation method the matrix decomposition method (`method="direct"`) is preferred if the number of variables is less than or equal to `direct_bestvariables`.

Default is 1200.

`loggauss` logical. Whether a log-Gauss random fields should be returned. See also `boxcox` for a generalisation.

`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_device` logical. If FALSE the current device is kept as it is; otherwise the current device is closed before the next device is opened. If NA it closes the preceding device if the opened device is pdf or jpeg.
Default: NA.
- `always_open_device` logical. If TRUE a new graphical window is opened for every `plot` if a standard graphical output is used, trying to respect the aspect ratios for the plots. The devices pdf and jpeg are always opened.
If NA then the value is set to `interactive()`.
Default: TRUE.
- `close_screen` logical; only relevant if `split_screen` = TRUE and `always_close_screen` = FALSE. If FALSE the windows opened by `split.screen` are left open.
Default: TRUE.
- `filecharacter`; only relevant if `split_screen` = TRUE. argument file in pdf. If "" then no internal naming is performed.
Default: "".
- `filenumber` integer; only relevant if `split_screen` = TRUE. Starting number of the file if `onefile`=FALSE. It is set to 0 whenever file is changed and `onefile`=FALSE.
Default 0.
- `grDefault` logical. If FALSE the graphic style up to Version 3.2 is used. Otherwise, the changes of th graphical style are reduced to a minimum.
Default: FALSE
- `grPrintlevel` integer values 0, 1, 2; only relevant when simulations are plotted. The higher the more text is shown in the plot.
Default: 1.
- `height` real number; only relevant if a new device is opened, see `alwyas_open_screen`.
- `height`=NA or `height` is not positive: no device is opened.
 - `width` = NA If `height` is greater than zero then it gives the height of a single figure in a plot created by **RandomFields**; See also `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`.
The width is calculated so that the aspect ratio is correct.
 - `width` not NA `height` and `width` give the size of the whole window.
- Default: 6.
- `increase_upto` See `height`.
Default: `c(3,4)`.
- `split_screen` logical. If TRUE `split.screen` is used to split the screen. Otherwise `par(mfcol)`. When using `split_screen` then the figures tend to be fancier.
Default: TRUE.
- `onefile` logical; only relevant if `split_screen` = TRUE. About the behaviour of argument `onefile` in pdf
Default: FALSE.
- `width` real number or NA; only relevant if `always_open_screen`=TRUE. See `height` for details.
Default: NA.

11. gui: Options for cRFgui

`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 corresponding button is pressed.

Default: TRUE.

`simu_method` "RPsirculant", "RPsutoff", "RPintrinsic", "RPsibm", "RPspectral", "RPsirect", "RPsquential", "RPsaverage", "RPsngget", "RPscoins", "RPhyperplane", "RPspecific", "any method".

Default: "RPsirculant".

`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` obsolete

`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 $(2 * \text{locsplitfactor} + 1)$ boxes in each direction.

Default: 2.

`locsplitn` vector of 3 components. A box should contain no more than `locsplitn[3]` points, but never less than `locsplitn[1]`. If a box had originally less than `locsplitn[1]` points, then the box is increased until at least `locsplitn[2]` points are in the box.

Default: `c(200, 1000, 5000)`.

method obsolete

`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 all 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 therefore should not be done too frequently. On the other hand, rare updates increase the computing times for taking the maximum over a single shape function. 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 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 when the constant c in the paper of Oesting, Schlather, Zhou (2018) has to be estimated. E.g. if `eps_zhou=0.01` then the first 2 digits should be correct.

Default: 0.01

`flathull` NA, FALSE, TRUE. Only used in M3 modelling in the algorithm by Oesting, Schlather, Zhou (2018). The argument is considered only if the simulation is performed on a grid. If `flat=TRUE`, then the density is considered to be `flat` in the convex hull of the grid, i.e. the simulation method of Schlather (2002) is used. If `flat=NA` the choice is done automatically.

Default: FALSE.

`max_gauss` The simulation of the max-stable process by the old-fashioned method of Schlather (2002) and by older methods for Brown-Resnick processes 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 (2018) 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 . So, it is advantageous to simulate all fields at once by `RFsimulate(..., n =)`.

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. This option allows the simulation to interrupt after `maxpoints` shape function have been placed.

Default: 2e9 (never).

`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`

`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`

`min_shape_gumbel` To increase speed, the minimum field value is assumed to be `min_shape_gumbel` for calculation of threshold values for simulation short cuts. During a simulation, its value becomes void as soon as the real (current) minimum of the field being simulated exceeds `min_shape_gumbel`

Default: -1e15.

`scatter_method` logical. If

Default: NA;

`xi` Extreme value index. Default: 2e9 . While ξ can be set globally, the shift μ and the scale s can be given only locally within the process definitions, e.g., [RPsmi th](#).

Default: 1.0.

15. mpp: Options for the random coins (shot noise) methods

`about_zero` In certain cases ([Coins, RMtruncsupport](#)), functions are assumed to zero if the value is less than `about_zero`.

Default: 0.001 .

`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 explicitly given.

Default: 50000 .

`scatter_method`

`scatter_size`, `scatter_max` Real valued and integer valued, respectively, or NA.

Used in the internal function `RMscatter` that calculates $\sum_{i=1}^n f(x + h_i)$ for some function f and for some distances h_i .

Let $\varepsilon = \text{about_zero}$, $s = \text{scatter_size}$ and $m = \text{scatter_max}$. We distinguish 4 cases:

- `scatter_size` > 0 and `scatter_max` ≥ 0
Here, n equals $(2m)^d$. and $h_i \in M = \{(ks, \dots, ks), \dots, (ms, \dots, ms)\}$ with $k = -m$.
- `scatter_size` > 0 and `scatter_max` < 0
same as the previous case, but m is chosen such that $f(k_i e_i s_i) \approx \varepsilon$, $-k_i \in N$, $i = 1, \dots, d$ and $f(m_i e_i s_i) \approx \varepsilon$, $m \in N$.

- `scatter_size <= 0` and `scatter_max >= 0`
This option is possible only for grids. Here h_i runs on the given grid $i = 1, \dots, d$, but at most `scatter_max` steps.
- `scatter_size <= 0` and `scatter_max < 0`
this option is possible only for grids. Here, h_i runs over the whole grid.

`shape_power` Shape functions are powered by `shape_power` before used as intensity function for the point process.

Default: 2.0.

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` The nugget tolerance influences two different kind of models

- [RPnugget](#)
- [R.is](#)

See there for more information.

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). The user can use the registers 0..9.

`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 simulation 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. solve: Options for solving linear systems

`det_as_log` See [RFoptions](#)

`eigen2zero` See [RFoptions](#)

`max_chol` integer. Maximum number of rows of a matrix in a Cholesky decomposition

Default: 8192

`max_svd` integer. Maximum number of rows of a matrix in a svd decomposition

Default: 6555

`pivot` Type of pivoting for the Cholesky decomposition. Possible values are

PIVOT_NONE No pivoting.

PIVOT_AUTO If the matrix has a size greater than 3x3 and Cholesky fails without pivoting, pivoting is done. For matrices of size less than 4x4, no pivoting and no checks are performed.

PIVOT_DO Do always pivoting. NOTE: pivoted Cholesky decomposition yields only very approximately an upper triangular matrix L, but still $L^t L = M$ holds true.

PIVOT_IDX uses the same pivoting as in the previous pivoted decomposition. This option becomes relevant only when simulations with different parameters or different models shall be performed with the same seed so that also the pivoting must be coupled.

Default: PIVOT_auto

`pivot_actual_size` See [RFoptions](#)

`pivot_check` logical. Only used in pivoted Cholesky decomposition. If TRUE and a numerically zero diagonal element is detected, it is checked whether the offdiagonal elements are numerically zero as well. (See also `pivot_max_deviation` and `pivot_max_reldeviation`.) if NA then, in [RPdirect](#), the value is equivalent to

FALSE if the model is positive (semi-)definite.

TRUE if the model is genuinely negative definite.

Default: NA

`pivot_idx` See [RFoptions](#)

`pivot_relerror` See [RFoptions](#)

`pivot_max_deviation` See [RFoptions](#)

`pivot_max_reldeviation` See [RFoptions](#)

`solve_method` See [RFoptions](#)

`spam_factor` See [RFoptions](#)

`spam_min_n` See [RFoptions](#)

`spam_min_p` See [RFoptions](#)

`spam_pivot` See [RFoptions](#)

`spam_sample_n` See [RFoptions](#)

`spam_tol` See [RFoptions](#)

`svdtol` See [RFoptions](#)

`use_spam` See [RFoptions](#)

20. 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.

21. 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](#)

22. 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 $k\pi/\text{lines}$ for k in 1:lines, otherwise.

This option is used by both [RPspectral](#) and [RPtbm](#), the latter only when the dimension is 2.

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.

23. 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.

- `do_tests` Internal variable. Do not use it. Default: FALSE.
- `examples_reduced` non-negative integer. If positive, then the design of any simulation in **RandomFields** is internally reduced in size (roughly down to the given value in each direction). Warnings report this behaviour. This option is necessary to run the examples of **RandomFields** under the time constraint of CRAN.
- `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_constant` The definition of `RMconstant` has changed. A warning is displayed if the command is used. `warn_constant` will become obsolete in future versions.
- Default: TRUE.
- `warn_coordinates` internally used logical argument. If TRUE then a transformation from earth coordinates to cartesian coordinates is reported.
- Default: TRUE.
- `allow_duplicated_locations` logical. If FALSE duplicated locations are not allowed. If TRUE then the (standard) nugget effect becomes a non-stationary model in an abstract space that cannot be extended outside the given locations. See also `RMnugget` for the distinction between measurement error and spatial nugget.
- Default: FALSE.
- `warn_missing_zenit` Only for Earth systems: a missing zenit is frequently a cause for errors that are difficult to understand. Therefore, in such cases an additional warning message is displayed.
- Default: TRUE
- `warn_newAniso` obsolete.
- 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_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_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.

`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_var` In some cases, **RandomFields** cannot detect whether the variance is non-negative. If TRUE then a warning is displayed in such a case. Default: TRUE.

Value

NULL if any argument is given, and the full list of arguments, otherwise.

Author(s)

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

References

Basic

- General

- 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.

- rectangular distribution; `eps_zhou`

- Oesting, M., Schlather, M. and Zhou, C. (2013) On the Normalized Spectral Representation of Max-Stable Processes on a compact set. *arXiv*, **1310.1813**

- `shape_power`

- Ballani, F. and Schlather, M. (2015) In preparation.

See Also

[RFsimulate](#), [RFoptionsAdvanced](#), [RFoptions](#), and [RFgetMethodNames](#).


```

# 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

RFoptions(seed=NA)
len <- 10

x <- seq(0, 1, len=len)
y <- seq(0, 1, len=len)
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 <- 5
n <- 100

# using local.dependent=FALSE (which is the default)
c1 <- numeric(m)
time <- system.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, force=TRUE)
    c1[i] <- cov(z[1, dim(z)[2], ], z[dim(z)[1], 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 <- system.time(
  for (i in 1:m) {
    cat("", i)
    z <- RFsimulate(meth(model), x, y, n=2 * n, pch="",
                    dependent=TRUE, spConform=FALSE, trials=5, force=TRUE)
    c2[i] <- cov(z[1, dim(z)[2], ], z[dim(z)[1], 1, ])
  })

print(time) ## 20 times faster
Print(true.cov, mean(c2), sd(c2), empty.lines=1) ## much better results

## the sd is smaller (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                      ##
#####

RFoptions(seed=0)
step <- 1
x1 <- seq(-150,150,step)
y1 <- seq(-15, 15, step)
x2 <- seq(-50, 50, step)
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
```

RFpar

Graphical parameters for plots

Description

This function sets globally graphical parameters for plots of RMmodels, simulations and estimations.

Usage

```
RFpar(...)
```

Arguments

```
... see par
```

Value

If RFpar is called without arguments, the current list is returned.

If RFpar is called with NULL only, the current list is deleted.

Otherwise the arguments are stored for global use in RandomFields.

Author(s)

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

See Also

[plot-method](#)

Examples

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

RFpar(col="red")
plot(RMexp())
```

RFpointsDataFrame-class

Class RFpointsDataFrame

Description

Class for attributes in one-dimensional space that are not on a grid.

Usage

```
## S4 method for signature 'RFpointsDataFrame'
RFspDataFrame2conventional(obj)
```

Arguments

obj an RFspatialPointsDataFrame object

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

range signature(`x = "RFpointsDataFrame"`): returns the range

hist signature(`x = "RFpointsDataFrame"`): plots histogram

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

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

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

as.data.frame signature(`x = "RFpointsDataFrame"`): converts data-slot and coordinates to a `data.frame`

Details

Methods summary and dimensions are defined for the “parent”-class [RFsp](#).

Author(s)

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

See Also

[RFspatialPointsDataFrame](#), which is for point locations in higher dimensional spaces, [RFpointsDataFrame-class](#) which is for one-dimensional locations on a grid, [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))
head(coordinates(f))
str(f[2]) ## selects second column of data-slot
all.equal(f, cbind(f,f)[1:3]) ## TRUE

plot(f, nmax=2)
```

RFpseudomadogram

Empirical Pseudomadogram

Description

Calculates the empirical pseudomadogram. The empirical pseudomadogram of two random fields X and Y is given by

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

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

Usage

```
RFpseudomadogram(model, x, y=NULL, z=NULL, T=NULL, grid, params, distances,
dim, ..., data, bin=NULL, phi=NULL, theta = NULL,
deltaT = NULL, vdim=NULL)
```

Arguments

model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If params is given, then ... may include also the variables used in params.
data	matrix, data.frame or object of class RFsp ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
bin	a vector giving the borders of the bins; If not specified an array describing the empirical (pseudo-)(cross-) covariance function in every direction is returned.
phi	an integer defining the number of sectors one half of the X/Y plane shall be divided 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 divided 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 <code>seq(from=0, to=deltaT[1], by=deltaT[2])</code>
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.

Details

[RFpseudomadogram](#) computes the empirical pseudomadogram 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 columns of matrix x. In this case do neither use y, nor z and write the columns 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 [RFpseudomadogram](#) to calculate the pseudomadogram (see [RFoptions](#)).

Value

RFpseudomadogram returns objects of class RFempVariog.

Author(s)

Jonas Auel; Sebastian Engelke; Johannes Martini; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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, RMmodel, RFsimulate, RFfit, RFCov, RFmadogram, RFvariogram.

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##           RFoptions(seed=NA) to make them all random again
model <- RMbiwm(nudiag=c(1, 2), nured=1, rhored=1, cdiag=c(1, 5),
               s=c(1, 1, 2))
n <- 2
x <- seq(0, 20, 0.1)
z <- RFsimulate(model, x=x, y=x, n=n)
emp.vario <- RFpseudomadogram(data=z)
plot(emp.vario)
```

RFpseudovariogram *Pseudovariogram*

Description

Calculates the theoretical and empirical Pseudovariogram.

Usage

```
RFpseudovariogram(model, x, y=NULL, z = NULL, T=NULL, grid, params, distances,
                 dim, ..., data, bin=NULL, phi=NULL, theta = NULL,
                 deltaT = NULL, vdim=NULL)
```

Arguments

model, params	object of class RMmodel , RFformula or formula ; best is to consider the examples below, first. The argument params is a list that specifies free parameters in a formula description, see RMformula .
x	vector of x coordinates, or object of class GridTopology or raster ; for more options see RFsimulateAdvanced .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually grid need not be given. See also RFsimulateAdvanced .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see RFsimulateAdvanced .
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by RFoptions . If params is given, then ... may include also the variables used in params.
data	matrix, data.frame or object of class RFsp ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
bin	a vector giving the borders of the bins; If not specified an array describing the empirical (pseudo-)(cross-) covariance function in every direction is returned.
phi	an integer defining the number of sectors one half of the X/Y plane shall be divided 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 divided 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 <code>seq(from=0, to=deltaT[1], by=deltaT[2])</code>
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.

Details

[RFpseudovariogram](#) computes the empirical pseudovariogram 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 columns of matrix x. In this case do neither use y, nor z and write the columns 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](#).)

Value

an objects of class `RFempVariog`.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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`, `RMmodel`, `RFsimulate`, `RFfit`, `RFcov`, `RFmadogram`, `RFvariogram`,

Examples

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

```
model <- RMbiwm(nudiag=c(1, 2), nured=1, rhored=1, cdiag=c(1, 5),
               s=c(1, 1, 2))
x <- seq(0, 20, 0.1)
z <- RFsimulate(model, x=x, y=x, n=2)
emp.vario <- RFpseudovariogram(data=z)
plot(emp.vario, model=model)
```

RFratiotest

Likelihood ratio test

Description

The function performs an approximate chi2 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=NULL, data, alpha, n = 5 / alpha, seed = 0,
            lower = NULL, upper = NULL, methods,
            sub.methods, optim.control = NULL, users.guess = NULL,
            distances = NULL, dim, transform = NULL, ...)
```


Arguments

<code>nullmodel</code> , <code>alternative</code>	See Details. The set of parameters to be estimated for <code>nullmodel</code> should be a subset of the parameters to be estimated for <code>alternative</code> if <code>alternative</code> is given.
<code>alpha</code>	value in [0,1] or missing. Significance level.
<code>n</code>	integer. The test is based on <code>n-1</code> simulations.
<code>seed</code>	integer. If not NULL and not NA, the <code>.Random.seed</code> is set to <code>seed</code> . Otherwise, <code>set.seed</code> is set to the value of <code>RFoptions{ }\$basic\$seed</code> if the latter is not NA.
<code>x</code>	vector of x coordinates, or object of class <code>GridTopology</code> or <code>raster</code> ; for more options see <code>RFsimulateAdvanced</code> .
<code>y</code> , <code>z</code>	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
<code>T</code>	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
<code>grid</code>	logical; the function finds itself the correct value in nearly all cases, so that usually <code>grid</code> need not be given. See also <code>RFsimulateAdvanced</code> .
<code>data</code>	matrix, data.frame or object of class <code>RFsp</code> ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
<code>lower</code>	list or vector. Lower bounds for the parameters. If <code>lower</code> is a vector, <code>lower</code> has to be a vector as well and its length must equal the number of parameters to be estimated. The order of <code>lower</code> has to be maintained. A component being NA means that no manual lower bound for the corresponding parameter is set. If <code>lower</code> is a list, <code>lower</code> has to be of (exactly) the same structure of the model.
<code>upper</code>	list or vector. Upper bounds for the parameters. See <code>lower</code> .
<code>methods</code>	Main methods to be used for estimating. If several methods are given, estimation will be performed with each method and the results reported.
<code>sub.methods</code>	variants of the least squares fit of the variogram. variants of the maximum likelihood fit of the covariance function. See <code>RFfit</code> for details.
<code>users.guess</code>	User's guess of the parameters. All the parameters must be given using the same rules as for <code>lower</code> (except that no NA's should be contained).
<code>distances</code> , <code>dim</code>	another alternative for the argument <code>x</code> to pass the (relative) coordinates, see <code>RFsimulateAdvanced</code> .
<code>optim.control</code>	control list for <code>optim</code> , which uses 'L-BFGS-B'. However <code>parscale</code> may not be given.
<code>transform</code>	obsolete for users; use <code>param</code> instead. <code>transform=list()</code> will return structural information to set up the correct function.
<code>...</code>	for advanced use: further options and control arguments for the simulation that are passed to and processed by <code>RFoptions</code> . If <code>params</code> is given, then <code>...</code> may include also the variables used in <code>params</code> .

Details

nullmodel (and the alternative) can be

- a covariance model, see [RMmodel](#) or type `RFgetModelNames(type="variogram")` to get all options.

If [RFoptions](#) `ratiotest_approx` is TRUE 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>

See Also

`RFfit`, `RMmodel`, `RandomFields`, `weather`.

Examples

RFsimulate

Simulation of Random Fields

Description

This function simulates **unconditional** random fields:

- univariate and multivariate, 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 multivariate, spatial and spatio-temporal Gaussian random fields

Here, only the simulation of Gaussian random fields is described. For other kinds of random fields (binary, max-stable, etc.) or more sophisticated approaches see [RFsimulateAdvanced](#).

Usage

```
RFsimulate(model, x, y=NULL, z=NULL, T=NULL, grid=NULL,  
           distances, dim, data, given=NULL, err.model, params,  
           err.params, n=1, ...)
```

Arguments

model, params	object of class <code>RMmodel</code> , <code>RFformula</code> or <code>formula</code> ; best is to consider the examples below, first. The argument <code>params</code> is a list that specifies free parameters in a formula description, see <code>RMformula</code> .
x	vector of x coordinates, or object of class <code>GridTopology</code> or <code>raster</code> ; for more options see <code>RFsimulateAdvanced</code> .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually <code>grid</code> need not be given. See also <code>RFsimulateAdvanced</code> .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see <code>RFsimulateAdvanced</code> .
data	For conditional simulation and random imputing only. If data is missing, unconditional simulation is performed. matrix, data.frame or object of class <code>RFsp</code> ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest. If <code>given</code> is not given and <code>data</code> is a matrix or <code>data</code> is a data.frame, RandomFields tries to identify where the data and the coordinates are, e.g. by names in formulae or by fixed names, see <code>Coordinate systems</code> . See also <code>RFsimulateAdvanced</code> . If all fails, the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field. Notes that also lists of data can be passed. If the argument x is missing, <code>data</code> may contain NAs, which are then replaced by conditionally simulated values (random imputing);
given	optional, matrix or list. If <code>given</code> matrix then the coordinates can be given separately, namely by <code>given</code> where, in each row, a single location is given. If <code>given</code> is a list, it may consist of x, y, z, T, <code>grid</code> . If <code>given</code> is provided, <code>data</code> must be a matrix or an array containing the data only.
err.model, err.params	For conditional simulation and random imputing only. In case of (assumed) error-free measurements (which is mostly the case in geostatistics) the argument <code>err.model</code> is not given. In case of measurement errors we have <code>err.model=RMnugget(var=var)</code> . <code>err.param</code> plays the same role as <code>params</code> for <code>model</code> .
n	number of realizations to generate. For a very advanced feature, see the notes in <code>RFsimulateAdvanced</code> .
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by <code>RFoptions</code> . If <code>params</code> is given, then ... may include also the variables used in <code>params</code> .

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>

References

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

[RFvariogram](#), [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
x.seq <- seq(from, to, length=200)
```

```

y.seq <- seq(from, to, length=200)

simu <- RFsimulate(model, x=x.seq, y=y.seq)
plot(simu)

#####
## ##
## Conditional simulation ##
## ##
#####

# first we simulate some random values at
# 100 random locations:
n <- 100
x <- runif(n=n, min=-1, max=1)
y <- runif(n=n, min=-1, max=1)
dta <- RFsimulate(model = RMexp(), x=x, y=y, grid=FALSE)
plot(dta)

# let simulate a field conditional on the above data
L <- if (interactive()) 100 else 5
x.seq.cond <- y.seq.cond <- seq(-1.5, 1.5, length=L)
model <- RMexp()
cond <- RFsimulate(model, x=x.seq.cond, y=y.seq.cond, data=dta)
plot(cond, dta)

```

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 kinds of random fields (binary, max-stable, etc.) or more sophisticated approaches see [RFsimulateAdvanced](#).

See [RFsimulate.sophisticated.examples](#) for further examples.

Author(s)

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

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 kinds of random fields (binary, max-stable, etc.) or more sophisticated approaches see [RFsimulateAdvanced](#).

Author(s)

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

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 multivariate, 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 multivariate, 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	<p>object of class RMmodel, RFformula or formula; specifies the model to be simulated</p> <ul style="list-style-type: none"> • if of class RMmodel, model specifies <ul style="list-style-type: none"> – the type of random field by using RPfunctions, e.g., <ul style="list-style-type: none"> * RPgauss: Gaussian random field (default if none of the functions in the list is given) * RPsmith: Smith model See RP for an overview. – the covariance or variogram model in case of a Gaussian random field (RPgauss) and for fields based on Gaussian fields (e.g. RPbernoulli); type RFgetModelNames(type="variogram") 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	<p>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_{i,j} 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.</p>
y	<p>optional vector of y coordinates, ignored if x is a matrix</p>
z	<p>optional vector of z coordinates, ignored if x is a matrix</p>
T	<p>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.</p>
grid	<p>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)</p>

<code>data</code>	matrix, <code>data.frame</code> or object of class <code>RFsp</code> ; coordinates and response values of measurements in case that conditional simulation is to be performed; if a matrix or a <code>data.frame</code> , the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field; if <code>x</code> is missing, data may contain NAs, which are then replaced by conditionally simulated values; if <code>data</code> is missing, unconditional simulation is performed; for details on matching of variable names see Details; if of class <code>RFsp</code>
<code>err.model</code>	same as <code>model</code> ; gives the model of the measurement errors for the measured data (which must be given in this case!), see Details. <code>err.model=NULL</code> (default) corresponds to error-free measurements, the most common alternative is <code>err.model=RMnugget()</code> ; ignored if <code>data</code> is missing.
<code>distances</code>	object of class <code>dist</code> representing the upper triangular 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 <code>NULL</code> , <code>dim</code> must be given and <code>x</code> , <code>y</code> , <code>z</code> and <code>T</code> must be missing or <code>NULL</code> . If <code>distances</code> are given, the current value of <code>spConform</code> , see <code>RFoptions</code> , is ignored and instead <code>spConform=FALSE</code> is used. (This fact may change in future.)
<code>dim</code>	integer; space or space-time dimension of the field
<code>n</code>	number of realizations to generate
<code>...</code>	further options and control arguments for the simulation that are passed to and processed by <code>RFoptions</code>

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 the list obtained by `RFgetModelNames(type="variogram", 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 $(x[i], y[i], z[i], T[k])$, $i = 1, \dots, \text{nrow}(x)$, $k = 1, \dots, \text{length}(T)$, 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 notation. 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$len, by=y$stepsize), s`

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 $(\text{dimension of index space} + 1)$ 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) > (\text{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 $[\text{space-time-dimension} > 1]$ and the coordinates are on a grid, result is of class `RFgridDataFrame` if $[\text{space-time-dimension} = 1]$ and the coordinates are on a grid, result is of class `RFspatialPointsDataFrame` if $[\text{space-time-dimension} > 1]$ and the coordinates are not on a grid, result is of class `RFpointsDataFrame` if $[\text{space-time-dimension} = 1]$ 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>

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](#), [RFvariogram](#), [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 and $space - time - dimension = 1$)

The first two 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

RFspDataFrame2dataArray signature(obj = "RFsp"): transforms RFsp objects to array

RFspDataFrame2conventional signature(obj = "RFsp"): transforms RFsp objects to a list with additional information

[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 itself, but the above mentioned classes in this package derived from it.

Author(s)

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

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

## Generating an object of class "RFspatialGridDataFrame"
## and one of class "RFspatialPointsDataFrame".

model <- RMcauchy(gamma=4, var=0.1, scale=2)
x <- seq(0,10,len=100)
r <- cbind(runif(100, min=1, max=9),runif(100, min=1, max=9))

z1 <- RFsimulate(model=model, x=x, y=x, n=4)
z2 <- RFsimulate(model=model, x=r, n=2)

## Applying available functions.

class(z1)
class(z2)
summary(z1)
```

```

summary(z2)
plot(z1)
plot(z2)

z4 <- RFspDataFrame2conventional(z2)
str(z2)
str(z4)

dta <- data.frame(coords=z4$x, data=z2@data[,1])
z3<-RFinterpolate(model=model, x=x, y=x, data=dta)
plot(z3,z2)

## Illustrating the warning.

a1 <- new("RFpointsDataFrame")
str(a1)
try(a2 <- new("RFsp")) ## ERROR

```

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 explicit transformation.

Usage

```

## S4 method for signature 'RFspatialGridDataFrame'
RFspDataFrame2conventional(obj, data.frame=FALSE)

```

Arguments

obj	an RFspatialGridDataFrame object
data.frame	logical. If TRUE a data.frame is returned.

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 [RFspatialGridDataFrame](#).

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 signature(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 signature(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 separate 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

range signature(x = "RFspatialGridDataFrame"): returns the range

hist signature(x = "RFspatialGridDataFrame"): plots histogram

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

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))
head(coordinates(f))
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 <- c(10, 1, 10, 10)
```

```

x2 <- rbind(c(0, 0, 0, 0),
  c(1, 0.2, 2, 5),
  steps)
scale <- 10

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)
f.old <- RFsimulate(model=RMexp(), x=x, n=n, spConform=FALSE)
all.equal(RFspDataframe2conventional(f.sp)$data, f.old, check.attributes=FALSE) ## 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 explicit transformation.

Usage

```

## S4 method for signature 'RFspatialPointsDataFrame'
RFspDataframe2conventional(obj)

```

Arguments

`obj` an [RFspatialPointsDataFrame](#) object

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

range signature(x = "RFspatialPointsDataFrame"): returns the range

hist signature(x = "RFspatialPointsDataFrame"): plots histogram

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

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

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

as.data.frame signature(x = "RFspatialPointsDataFrame"): converts data-slot and coordinates to a data.frame

Details

Note that in the data-slot, each column 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, Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.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))
head(coordinates(f))
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)
f.old <- RFsimulate(model=RMexp(), x=x, n=3, spConform=FALSE)
all.equal(RFspDataframe2conventional(f.sp)$data, f.old, check.attributes=FALSE) ## TRUE
```

RFvariogram

Empirical (Cross-)Variogram

Description

Calculates empirical (cross-)variogram.

Usage

```
RFvariogram(model, x, y=NULL, z = NULL, T=NULL, grid,
            params, distances, dim, ...,
            data, bin=NULL, phi=NULL, theta = NULL,
            deltaT = NULL, vdim=NULL)
```

Arguments

`model, params` object of class [RMmodel](#), [RFformula](#) or [formula](#); best is to consider the examples below, first.
The argument `params` is a list that specifies free parameters in a formula description, see [RMformula](#).

x	vector of x coordinates, or object of class <code>GridTopology</code> or <code>raster</code> ; for more options see <code>RFsimulateAdvanced</code> .
y, z	optional vectors of y (z) coordinates, which should not be given if x is a matrix.
T	optional vector of time coordinates, T must always be an equidistant vector. Instead of <code>T=seq(from=From, by=By, len=Len)</code> , one may also write <code>T=c(From, By, Len)</code> .
grid	logical; the function finds itself the correct value in nearly all cases, so that usually <code>grid</code> need not be given. See also <code>RFsimulateAdvanced</code> .
distances, dim	another alternative for the argument x to pass the (relative) coordinates, see <code>RFsimulateAdvanced</code> .
...	for advanced use: further options and control arguments for the simulation that are passed to and processed by <code>RFoptions</code> . If <code>params</code> is given, then ... may include also the variables used in <code>params</code> .
data	matrix, <code>data.frame</code> or object of class <code>RFsp</code> ; If a matrix is given the ordering of the columns is the following: space, time, multivariate, repetitions, i.e. the index for the space runs the fastest and that for repetitions the slowest.
bin	a vector giving the borders of the bins; If not specified an array describing the empirical (pseudo-)(cross-) covariance function in every direction is returned.
phi	an integer defining the number of sectors one half of the X/Y plane shall be divided into. If not specified, either an array is returned (if <code>bin</code> missing) or isotropy is assumed (if <code>bin</code> specified).
theta	an integer defining the number of sectors one half of the X/Z plane shall be divided into. Use only for dimension $d = 3$ if <code>phi</code> is already specified.
deltaT	vector of length 2, specifying the temporal bins. The internal bin vector becomes <code>seq(from=0, to=deltaT[1], by=deltaT[2])</code>
vdim	the number of variables of a multivariate data set. If not given and <code>data</code> is an <code>RFsp</code> object created by <code>RandomFields</code> , the information there is taken from there. Otherwise <code>vdim</code> is assumed to be one. NOTE: still the argument <code>vdim</code> is an experimental stage.

Details

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

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 distance vector $t_{i,j} = r$.

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

Value

RFvariogram returns objects of class RFempVariog.

Author(s)

Sebastian Engelke; Johannes Martini; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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, RMmodel, RFsimulate, RFfit, RFCov, RFpseudovariogram. RFmadogram.

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, 0.02)
z <- RFsimulate(model, x=x, n=n)
emp.vario <- RFvariogram(data=z)
plot(emp.vario, model=model)

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

## space-time model
model <- RMnsst(phi=RMexp(), psi=RMfbm(alpha=1), delta=2)
x <- seq(0, 10, 0.05)
T <- c(0, 0.1, 100)
z <- RFsimulate(x=x, T=T, model=model, n=n)
emp.vario <- RFvariogram(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, 0.1)
z <- RFsimulate(model, x=x, y=x, n=n)
emp.vario <- RFvariogram(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, 0.1)
dta <- RFsimulate(model, x, x, n=n)
ev <- RFvariogram(data=dta, phi=4)
plot(ev, model=model, boundaries=FALSE)
```

RMangle

Anisotropy matrix given by angle

Description

RMangle delivers an anisotropy matrix for the argument `Aniso` in `RMmodel` in two dimensions. RMangle requires one or two stretching 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 $A = \text{diag}(d1, d2) \%*\% \text{matrix}(\text{ncol}=2, \text{c}(\cos(a), \sin(a), -\sin(a), \cos(a)))$

In three dimensions and with angle equal to a , second angle L and `diag` equal to $(d1, d2, d3)$ the anisotropy matrix A is

$A = \text{diag}(d1, d2, d3) \%*\% \text{matrix}(\text{ncol}=3, \text{c}(\cos(a) * \cos(L), \sin(a) * \cos(L), \sin(L), -\sin(a), \cos(a), 0, -\cos(a) * \sin(L), -\sin(a) * \sin(L), \cos(L)))$ i.e. Ax turns a vector x first in the $x - z$ plane, then in the $x - y$ plane.

Usage

```
RMangle(angle, lat.angle, ratio, diag)
```

Arguments

<code>angle</code>	angle a
<code>lat.angle</code>	second angle; in 3 dimensions only
<code>ratio</code>	equivalent to <code>diag=c(1, 1/ratio)</code> ; in 2 dimensions only
<code>diag</code>	the diagonal components of the matrix

Value

`RMangle` returns an object of class `RMmodel`.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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, 0.05)
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, 0.2)
z <- RFsimulate(x, x, x, model=model)
plot(z, MARGIN.slices=3)

## next model gives an example how to estimate the parameters back
n <- 20
x <- runif(n, 0, 10)
y <- runif(n, 0, 10)
coords <- expand.grid(x, y)
model <- RMexp(Aniso=RMangle(angle=pi/4, diag=c(1/4, 1/12)))
d <- RFsimulate(model, x=coords[, 1], y=coords[, 2], n=10)
estmodel <- RMexp(Aniso=RMangle(angle=NA, diag=c(NA, NA)))
system.time(RFfit(estmodel, data=d, modus_operandi='sloppy'))
```

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 only valid on the real line.

Value

[RMaskey](#) returns an object of class [RMmodel](#).

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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, 0.02)
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>, <http://ms.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 evaluation
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 ) ) )
##

## Not run: stopifnot(abs(z1-z2)<1e-12, abs(z2-z3)<1e-12)
```

RMball

RMball

Description

RMball refers 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>

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

x <- seq(0,10,len=100)
model <- RMball(var=2,scale=1.5)
plot(model)
z <- RFsimulate(RPpoisson(model),x=x,y=x,intensity=0.1)
plot(z)
```

Description

RMbcw is a variogram model that bridges between some intrinsically stationary isotropic processes and some stationary ones. It reunifies the **RMgenfbm** ‘b’, **RMgencauchy** ‘c’ and **RMdewijsian** ‘w’.

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, c, var, scale, Aniso, proj)`

Arguments

<code>alpha</code>	a numerical value; should be in the interval (0,2].
<code>beta</code>	a numerical value; should be in the interval (-infty,2].
<code>c</code>	only for experts. If given, a not necessarily positive definite function $c - \gamma(r)$ is built.
<code>var, scale, Aniso, proj</code>	optional arguments; same meaning for any RMmodel . If not passed, the above variogram remains unmodified.

Details

For $\beta > 0$, $\beta < 0$, $\beta = 0$ we have the generalized fractal Brownian motion **RMgenfbm**, the generalized Cauchy model **RMgencauchy**, and the de Wijsian 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>, <http://ms.math.uni-mannheim.de>

References

- Schlather, M (2014) A parametric variogram model bridging between stationary and intrinsically stationary processes. *arxiv* **1412.1914**.

See Also

[RMl1sfbm](#) is equipped with Matheron's constant c for the fractional brownian motion, [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, 0.02)
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>

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, 0.02)
GaussModel <- RMgneiting()

n <- 1000
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))
z1 <- as.matrix(z)
estim.cov <- apply(z1, 1, function(x) cov(x, z1[1,]))
points(coordinates(z), estim.cov, col="red")
```

R**M**bessel

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

$$C(r) = \sin(r)/r$$

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>, <http://ms.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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMbicauchy

Bivariate Cauchy Model

Description

[RMbicauchy](#) is a bivariate stationary isotropic covariance model whose corresponding covariance function only depends on the distance $r \geq 0$ between two points.

For constraints on the constants see [Details](#).

Usage

```
RMbicauchy(alpha, beta, s, rho, var, scale, Aniso, proj)
```

Arguments

alpha	[to be done]
beta	[to be done]
s	a vector of length 3 of numerical values; each entry positive; the vector (s_{11}, s_{21}, s_{22})
rho	[to be done]
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: [to be done]

Value

[RMbicauchy](#) returns an object of class [RMmodel](#).

Author(s)

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

References

- Moreva, O., Schlather, M. (2016) Modelling and simulation of bivariate Gaussian random fields. *arXiv 1412.1914*

See Also

[RMcauchy](#), [Multivariate RMmodels](#).

Examples

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

## todo
```

 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 generalized Gneiting model with parameters n and δ , see [RMgengneiting](#), i.e.,

$$C_{\kappa=0,\delta}(r) = (1-r)^\beta 1_{[0,1]}(r), \quad \beta = \delta + 2\kappa + 1/2;$$

$$C_{\kappa=1,\delta}(r) = (1+\beta r)(1-r)^\beta 1_{[0,1]}(r), \quad \beta = \delta + 2\kappa + 1/2;$$

$$C_{\kappa=2,\delta}(r) = \left(1 + \beta r + \frac{\beta^2 - 1}{3} r^2\right) (1-r)^\beta 1_{[0,1]}(r), \quad \beta = \delta + 2\kappa + 1/2;$$

$$C_{\kappa=3,\delta}(r) = \left(1 + \beta r + \frac{(2\beta^2 - 3)}{5} r^2 + \frac{(\beta^2 - 4)\beta}{15} r^3\right) (1-r)^\beta 1_{[0,1]}(r), \quad \beta = \delta + 2\kappa + 1/2.$$

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, . . . , 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 random field.
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 $s_{12} = s_{21} = \text{sred12} * \min\{s_{11}, s_{22}\}$.
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

$$c_{12} = \rho_{\text{red}} \cdot m \cdot \left(c_{11}c_{22} \prod_{i,j=1,2} \left(\frac{\Gamma(\gamma_{ij} + \mu + 2\kappa + 5/2)}{b_{ij}^{\nu_{ij} + 2\kappa + 1} \Gamma(1 + \gamma_{ij}) \Gamma(\mu + 2\kappa + 3/2)} \right)^{(-1)^{i+j}} \right)^{1/2}$$

where $\rho_{\text{red}} \in [-1, 1]$.

The constant m in the formula above is obtained as follows:

$$m = \min\{1, m_{-1}, m_{+1}\}$$

Let

$$\begin{aligned} a &= 2\gamma_{12} - \gamma_{11} - \gamma_{22} \\ b &= -2\gamma_{12}(s_{11} + s_{22}) + \gamma_{11}(s_{12} + s_{22}) + \gamma_{22}(s_{12} + s_{11}) \\ e &= 2\gamma_{12}s_{11}s_{22} - \gamma_{11}s_{12}s_{22} - \gamma_{22}s_{12}s_{11} \\ d &= b^2 - 4ae \\ t_j &= \frac{-b + j\sqrt{d}}{2a} \end{aligned}$$

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

`RMbigneiting` returns an object of class `RMmodel`.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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. Technical report.
 RMbigneiting is based on this original work. D.J. Daley, E. Porcu and M. Bevilacqua have published end of 2014 an article intentionally without clarifying the genuine authorship of RMbigneiting, in particular, neither referring to this original work nor to **RandomFields**, which has included RMbigneiting since version 3.0.5 (05 Dec 2013).
- 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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

R**M**bistable

Bivariate stable Model

Description

`RMbistable` is a bivariate stationary isotropic covariance model whose corresponding covariance function only depends on the distance $r \geq 0$ between two points.

For constraints on the constants see Details.

Usage

```
RMbistable(alpha, s, cdiag, rho, rhored, betared, alphadiag, var, scale, Aniso, proj)
```

Arguments

alpha, alphadiag [to be done]

s a vector of length 3 of numerical values; each entry positive; the vector (s_{11}, s_{21}, s_{22})

cdiag [to be done]

rho, rhored [to be done]

betared to do

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: [to be done]

Value

[RMbistable](#) returns an object of class [RMmodel](#).

Author(s)

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

References

- Moreva, O., Schlather, M. (2016) Modelling and simulation of bivariate Gaussian random fields. *arXiv 1412.1914*

See Also

[RMstable](#), [Multivariate RMmodels](#).

Examples

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

## todo
```

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

$$c_{12} = c_{21} = \rho_{red} \sqrt{f m c_{11} c_{22}}$$

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>, <http://ms.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`, `Multivariate RMmodels`.

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 <- 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))
```

 RMblend

Scale model for a few areas of different scales and/or differentiabilitys

Description

Let $Z = (Z_1, \dots, Z_k)$ be an k -variate random field and A_1, \dots, A_k a partition of the space. Then

$$Y(x) = \sum_{i=1}^k Z_i * 1(x \in A_i)$$

i.e. the model blends the components of Z to a new, univariate model Y .

Usage

```
RMblend(multi, blend, thresholds, var, scale, Aniso, proj)
```

Arguments

`multi` a multivariate covariance function

`blend, thresholds`

The threshold is a vector of increasing values. If the value of `blend` is below all thresholds up to the k -th threshold, then the k -th component of the field given by `multi` is taken. If necessary the components are recycled.

Default: `threshold = 0.5`, useful for blending a bivariate field if `blend` takes only the values 0 and 1.

`var, scale, Aniso, proj`

optional arguments; same meaning for any [RMmodel](#). If not passed, the above covariance function remains unmodified.

Value

[RMblend](#) returns an object of class [RMmodel](#).

Author(s)

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

References

- Bonat, W.H. , Ribeiro, P. Jr. and Schlather, M. (2019) Modelling non-stationarity in scale. In preparation.
- Genton, Apanovich *Biometrika*.

See Also

[RMSadvanced](#), [RMBubble](#), [RMScale](#),

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, if (interactive()) 0.01 else 0.5)
len <- length(x)
m <- matrix(1:len, nc=len, nr=len)
m <- m > t(m)
image(m) # two areas separated by the first bisector

biwm <- Rmbiwm(nudiag=c(0.3, 1), nured=1, rhored=1, cdiag=c(1, 1),
              s=c(1, 1, 0.5))
model <- RMblend(multi=biwm, blend=RMcovariate(data = as.double(m), raw=TRUE))
plot(z <- RFsimulate(model, x, x)) ## takes a while ...
```

RMr2bg

Transformation from Brown-Resnick to Bernoulli

Description

This function can be used to model a max-stable process based on 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

$$4(\text{erf}^{-1}(1/2))^2 = 2 * \Phi^{-1}(3/4)^2 = 1.819746/2 = 0.9098728$$

Usage

```
RMr2bg(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

RMr2bg

binary random field `RPbernoulli` simulated with `RMr2bg(RMmodel())` has a uncentered covariance function that equals

1. the tail correlation function of the max-stable process constructed with this binary random field

- 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>

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)
x <- seq(0, 10, 0.05)
z <- RFsimulate(RPschlather(RMbr2eg(model)), x, x)
plot(z)
```

RMbr2eg

Transformation from Brown-Resnick to Gauss

Description

This function can be used to model a max-stable process based on 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

$$4(\operatorname{erf}^{-1}(1/\sqrt{2}))^2 = 2 * [\Phi^{-1}([1 + 1/\sqrt{2}]/2)]^2 = 4.425098/2 = 2.212549$$

Usage

```
RMr2eg(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`RMr2eg`

The extremal Gaussian model `RPschlather` simulated with `RMr2eg(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>

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`, `RMr2bg`, `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(RMr2bg(model))
x <- seq(0, 10, 0.05)

z <- RFsimulate(RPschlather(binary.model), x, x)
plot(z)
```

R**M**brownresnick*Tail correlation function of the Brown-Resnick process***Description**

R**M**brownresnick 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 [R**M**model](#).

var, scale, Aniso, proj

optional arguments; same meaning for any [R**M**model](#). If not passed, the above covariance function remains unmodified.

Details

For a given [R**M**model](#) the function `RMbrownresnick(RMmodel\(\))` 'returns' the tail correlation function of a Brown-Resnick process with variogram [R**M**model](#).

Value

object of class [R**M**model](#)

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 consistent with the rest of the package.

Author(s)

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

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 <- seq(-5, 5, 0.05)
z <- RFsimulate(RMBrownresnick(RMmodel), x=x, y=x)
plot(z)
```

RMBubble

Bubble model for arbitrary areas of scales

Description

A model that allows for arbitrary areas of scale applied to an isotropic model, i.e.

$$C(x, y) = \phi(\|x - y\|/s)$$

as long as $s_x = s_y = s$. Here, s_x is the scaling at location x ,

The cross-correlations between areas of different scales are given through a modified distance d . Let z_s be a finite subset of R^d depending on the scale s . Let w_u be a weight for an auxiliary point $u \in z_s$ with $\sum_{u \in z_s} w_u = 1$. Let $\tau_x = s_x^{-2}$. Then

$$d^2(x, y) = \min\{\tau(x), \tau(y)\} \|x - y\|^2 + \sum_{\xi \in \text{span}(\tau(x), \tau(y))} \sum_{u \in z_{\xi-0.5}} w_u \|x - u\|^2 \Delta\xi$$

Here, $\text{span}(\tau(x), \tau(y))$ is the finite set of values s^{-2} that are realized on the locations of interest and $\Delta\xi$ is the difference of two realized and ordered values of the scaling s .

Usage

```
RMBubble(phi, scaling, z, weight, minscale, barycentre,
          var, scale, Aniso, proj)
```

Arguments

phi	isotropic submodel
scaling	model that gives the non-stationary scaling s_x
z	matrix of the union of all z_s . The number of rows equals the dimension of the field. If not given, the locations with non-vanishing gradient are taken.
weight	vector of weights w whose length equals the number of columns of z. The points given by z might be weighted.
minscale	vector for partitioning z into classes z_s . Its length equals the number of columns of z. The vector values must be descending. See details. If not given then $z_s = z$ for all s . Else see details.
barycentre	logical. If FALSE and z is not given, the reference locations are those with non-vashing gradient. If TRUE then, for each realized value of the scale, the barycentre of the corresponding reference locations is used instead of the reference locations themselves. This leads to higher correlations, but also to highly non-stationary cross-correlation between the areas of different scale. The argument has no effect when z is given. Default: FALSE.
var, scale, Aniso, proj	optional arguments; same meaning for any RMmodel . If not passed, the above covariance function remains unmodified.

Details

minscale gives the minimal scale s value above which the corresponding points z define the set z_s . The validity of the set z_s ends with the next lower value given.

Let minscale = (10, 10, 10, 7, 7, 7, 0.5). Then for some d -dimensional vectors z_1, \dots, z_7 we have

$$z_s = \{z_1, z_2, z_3\}, s \geq 10$$

$$z_s = \{z_4, z_5, z_5\}, 7 \geq s < 10$$

$$z_s = \{z_7\}, s \geq 0.5$$

Note that, in this case, all realized scaling values must be ≥ 0.5 . Note further, that the weights for the subset must sum up to one, i.e.

$$w_1 + w_2 + w_3 = w_4 + w_5 + w_6 = w_7 = 1.$$

Value

[R**M**bubble](#) returns an object of class [RMmodel](#).

Note

This model is defined only for grids.

Author(s)

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

References

- Bonat, W.H. , Ribeiro, P. Jr. and Schlather, M. (2019) Modelling non-stationarity in scale. In preparation.

See Also

[RMSadvanced](#), [RMblend](#), [RMscale](#)

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, if (interactive()) 0.02 else 0.5)
d <- sqrt(rowSums(as.matrix(expand.grid(x-0.5, x-0.5))^2))
d <- matrix(d < 0.25, nc=length(x))
image(d)

scale <- RMcovariate(data=as.double(d) * 2 + 0.5, raw=TRUE)

## two models:
## the first uses the standard approach for determining the
##          reference point z, which is based on gradients
## the second takes the centre of the ball
model1 <- RMbubble(RMexp(), scaling=scale)
model2 <- RMbubble(RMexp(), scaling=scale, z=c(0.5, 0.5))
model3 <- RMbubble(RMexp(), scaling=scale, barycentre=TRUE) # approx. of model2

## model2 has slightly higher correlations than model1:
C1 <- RFcovmatrix(model1, x, x)
C2 <- RFcovmatrix(model2, x, x)
C3 <- RFcovmatrix(model3, x, x)
print(range(C2 - C1))
dev.new(); hist(C2 - C1)
print(range(C3 - C2)) # only small differences to C2
print(mean(C3 - C2))
dev.new(); hist(C3 - C2)

plot(z1 <- RFsimulate(model1, x, x))
plot(z2 <- RFsimulate(model2, x, x))
plot(z3 <- RFsimulate(model3, x, x)) # only tiny differences to z2

## in the following we compare the standard bubble model with
## the models RMblend, RMscale and RMS (so, model2 above
## performs even better)
biwm <- RMBiwm(nudiag=c(0.5, 0.5), nured=1, rhored=1, cdiag=c(1, 1),
```

```

      s=c(0.5, 2.5, 0.5))
blend <- Rblend(multi=biwm, blend=RMcovariate(data = as.double(d), raw=TRUE))
plot(zblend <- RFsimulate(blend, x, x)) ## takes a while ...
Cblend <- RFcovmatrix(blend, x, x)

Mscale <- RMscale(RMexp(), scaling = scale, penalty=RMid() / 2)
plot(zscale <- RFsimulate(Mscale, x, x))
Cscale <- RFcovmatrix(Mscale, x, x)

Mscale2 <- RMscale(RMexp(), scaling = scale, penalty=RMid() / 20000)
plot(zscale2 <- RFsimulate(Mscale2, x, x))
Cscale2 <- RFcovmatrix(Mscale2, x, x)

S <- RMexp(scale = scale)
plot(zS <- RFsimulate(S, x, x))
CS <- RFcovmatrix(S, x, x)

print(range(C1 - CS))
print(range(C1 - Cscale))
print(range(C1 - Cscale2))
print(range(C1 - Cblend))
dev.new(); hist(C1-CS)      ## C1 is better
dev.new(); hist(C1-Cscale) ## C1 is better
dev.new(); hist(C1-Cscale2) ## both are equally good. Maybe C1 slightly better
dev.new(); hist(C1-Cblend) ## C1 is better

```

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 parameter 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 $\text{scale} = \delta$.

Value

[RMcauchy](#) returns an object of class [RMmodel](#).

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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, 0.02)
plot(model, xlim=c(-3, 3))
plot(RFsimulate(model, x=x, n=4))
```

 RMcauchytm

Modifications of the Cauchy Family Covariance Model

Description

`RMcauchytm()` is a shortcut of `RMtbm(RMgencauchy())` and is given here for downwards compatibility.

Usage

```
RMcauchytm(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

`RMcauchytm` returns an object of class [RMmodel](#).

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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 <- RMcauchytm(alpha=1, beta=1, gamma=3)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMchoquet

Schoenberg's representation for the classes $\psi_{d,2}$ and $\psi_{d,\infty}$ in $d=2$ **Description**

[RMchoquet](#) is an isotropic covariance model. The corresponding covariance function only depends on the angle $0 \leq \theta \leq \pi$ between two points on the sphere and is given for $d=2$ by

$$\psi(\theta) = \sum_{n=0}^{\infty} b_{n,2}/(n+1) * P_n(\cos(\theta)),$$

where

$$\sum_{n=0}^{\infty} b_{n,d} = 1$$

and P_n is the Legendre Polynomial of integer order $n \geq 0$.

Usage

`RMchoquet(b)`

Arguments

`b` a numerical vector of weights in $(0, 1)$, such that $\text{sum}(b)=1$.

Details

By the results (cf. Gneiting, T. (2013), p.1333) of Schoenberg and others like Menegatto, Chen, Sun, Oliveira and Peron, the class $\psi_{d,2}$ of all real valued functions on $[0, \pi]$, with $\psi(0) = 1$ and such that the associated isotropic function

$$h(x, y) = \psi(\theta) \text{ with } \cos(\theta) = \langle x, y \rangle$$

$$\text{for } x, y \text{ in } \mathbb{R}^d : \|x\| = 1$$

is (strictly) positive definite is represented by this covariance model. The model can be interpreted as Choquet representation in terms of extremal members, which are non-strictly positive definite.

Special cases are the multiquadric family (see [RMmultiquad](#)) and the model of the sine power function (see [RMsinepower](#)).

Value

[RMchoquet](#) returns an object of class [RMmodel](#).

Author(s)

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

References

- Gneiting, T. (2013) *Strictly and non-strictly positive definite functions on spheres*. Bernoulli, **19**(4), 1327-1349.
- Schoenberg, I.J. (1942) *Positive definite functions on spheres*. Duke Math.J.,**9**, 96-108.
- Menegatto, V.A. (1994) *Strictly positive definite kernels on the Hilbert sphere*. Appl. Anal., **55**, 91-101.
- Chen, D., Menegatto, V.A., and Sun, X. (2003) *A necessary and sufficient condition for strictly positive definite functions on spheres*. Proc. Amer. Math. Soc.,**131**, 2733-2740.
- Menegatto, V.A., Oliveira, C.P. and Peron, A.P. (2006) *Strictly positive definite kernels on subsets of the complex plane*. Comput. Math. Appl., **51**, 1233-1250.

See Also

[RMmodel](#), [RFsimulate](#), [RFfit](#), [spherical models](#), [RMmultiquad](#), [RMsinepower](#)

Examples

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

## to do
```

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

$$C(r) = 1 - 2/\pi(r\sqrt{1-r^2} + \arcsin(r))1_{[0,1]}(r).$$

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>, <http://ms.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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMconstant

Covariance Matrix Constant in Space

Description

`RMconstant` defines a spatially constant covariance function.

Usage

```
RMconstant(M, var)
```

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.
var	variance

Value

`RMconstant` returns an object of class `RMmodel`.

Author(s)

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

See Also

`RMfixcov`, `RMmodel`.

Examples

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

model <- RMconstant(diag(2),var=3)
plot(model)
x <- seq(0,10,length=100)
z <- RFsimulate(model=model,x=x)
```

RMcov

Non-stationary covariance model corresponding to a variogram model

Description

This function generalizes the well-known non-stationary covariance function $2 \min\{x, y\}$ of the Brownian motion with variogram $\gamma(x, y) = |x - y|$, $x, y \geq 0$ to arbitrary variogram models any spatial processes of any dimension and multivariability.

Furthermore, the standard condition for the Brownian motion W is that variance equals 0 at the origin, i.e., $W(x) = Z(x) - Z(0)$ for any zero mean Gaussian process Z with variogram $\gamma(x, y) = |x - y|$ is replaced by $W(x) = Z(x) - \sum_{i=1}^n a_i Z(x_i)$ with $\sum_{i=1}^n a_i = 1$.

For a given variogram γ , a_i and x_i , the model equals $C(x, y) = \sum_{i=1}^n a_i (\gamma(x, x_i) + \gamma(x_i, y)) - \gamma(x, y) - \sum_{i=1}^n \sum_{j=1}^n a_i a_j \gamma(x_i, y_j)$

Usage

```
RMcov(gamma, x, y=NULL, z=NULL, T=NULL, grid, a,
      var, scale, Aniso, proj, raw, norm)
```

Arguments

gamma	a variogram model. Possibly multivariate.
x,y,z,T,grid	The usual arguments as in RFsimulate to define the locations where the covariates are given. Additional x might be set to one of the values "origin", "center", "extremals", or "all". If x is not given, x is set to "origin".
a	vector of weights. The length of a must equal the number of points given by x, y, z and T. The values of a must sum up to 1. If a is not given, equals weights are used.
var,scale,Aniso,proj	optional arguments; same meaning for any RMmodel . If not passed, the above covariance function remains unmodified.
raw	logical. If FALSE then the data are interpolated. This approach is always save, but might be slow. If TRUE then the data may be accessed when covariance matrices are calculated. No rescaling or anisotropy definition is allowed in combination with the model. The use is dangerous, but fast. Default: FALSE.
norm	optional model that gives the norm between locations

Value

[RMcov](#) 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
bm <- RMfbm(alpha=1)
plot(bm)

x <- seq(0, 6, if (interactive()) 0.125 else 3)
plot(RFsimulate(bm, x))

## standardizing with the random variable at the origin
z1 <- RFsimulate(RMcov(bm), x)
plot(z1)
z1 <- as.vector(z1)
zero <- which(abs(x) == 0)
stopifnot(abs(z1[zero]) < 1e-13)

## standardizing with the random variable at the center of the interval

```

```

z2 <- RFsimulate(RMcov(bm, "center"), x)
plot(z2)
z2 <- as.vector(z2)
stopifnot(abs(z2[(length(z2) + 1) / 2]) < 1e-13)

## standardizing with the random variables at the end points of the interval
z3 <- RFsimulate(RMcov(bm, "extremals"), x)
plot(z3)
z3 <- as.vector(z3)
stopifnot(abs(z3[1] + z3[length(z3)]) < 1e-13)

```

RMcovariate

Model for covariates

Description

The model makes covariates available.

Usage

```

RMcovariate(formula=NULL, data, x, y=NULL, z=NULL, T=NULL, grid,
            raw, norm, addNA, factor)

```

Arguments

formula, data	formula and by which the data should be modelled, similar to lm . If formula is not given, the the linear model is given by the data themselves.
x,y,z,T,grid	optional. The usual arguments as in RFsimulate to define the locations where the covariates are given.
raw	logical. If FALSE then the data are interpolated. This approach is always save, but might be slow. If TRUE then the data may be accessed when covariance matrices are calculated. No rescaling or anisotropy definition is allowed in combination with the model. The use is dangerous, but fast. Default: FALSE.
norm	optional model that gives the norm between locations
addNA	If addNA is TRUE, then an additional (linear) factor is estimated in an estimation framework. This parameter must be set in particular when <code>RMcovariate</code> passes several covariates.
factor	real value. From user's point of view very much the same as setting the argument <code>var</code> .

Details

The function interpolates (nearest neighbour) between the values.

Value

`RMcovariate` returns an object of class `RMmodel`.

Note

- `c`, `x` also accept lists of data. However, its use is not in an advanced stage yet.

Author(s)

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

See Also

`RMfixcov`, `RMmodel`, `RMtrend`

Examples

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

z <- 0.2 + (1:10)
Rffctn(RMcovariate(z), 1:10)
Rffctn(RMcovariate(data=z, x=1:10), c(2, 2.1, 2.5, 3))
```

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 <code>RMmodel</code> whose monotone property equals 'normal mixture', see <code>RFgetModelNames(monotone="normal mixture")</code> and whose <code>maxdim</code> 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 <code>RMmodel</code> . 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

$$C(h, t) = E_V C_0(h - Vt)$$

where the random wind speed vector V follows a d -variate normal distribution with expectation mu and covariance matrix $D/2$ (cf. Schlather, M. (2010), Example 9).

Value

`RMcoxisham` returns an object of class `RMmodel`.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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, 0.3)
```

```
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>, <http://ms.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, 0.02)
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 the following components:

- the potential
- the vector field given by

$$C(h) = (-\nabla_h(\nabla_h)^T)C_0(h)$$

- the field of sinks and sources

Usage

```
RMcurlfree(phi, which, var, scale, Aniso, proj)
```

Arguments

`phi` a univariate stationary covariance model (2- or 3-dimensional).

`which` vector of integers. If not given all components are returned; otherwise the selected components are returned.

`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>, <http://ms.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

[RMderiv](#), [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, 0.2)
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, <code>RFgetModelNames(type="positive definite", domain="single variable", isotropy="isotropic")</code> .
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 <code>RMmodel</code> . 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>, <http://ms.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 \mathbb{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

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

model <- RMexp()
plot(model, model.cutoff=RMcutoff(model, diameter=1), xlim=c(0, 4))
```

```

model <- RMstable(alpha = 0.8)
plot(model, model.cutoff=RMcutoff(model, diameter=2), xlim=c(0, 5))
x <- y <- seq(0, 4, 0.05)
plot(RFsimulate(RMcutoff(model), x=x, y = y))

```

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>, <http://ms.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, 0.02)
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 argument 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>, <http://ms.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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMdeclare

Declaration of dummy variables for statistical inference

Description

The only purpose of this function is the declaration of dummy variables for defining more complex relations between parameters that are to be estimated.

Its value as a covariance model is identically zero, independently of the variables declared.

Usage

```
RMdeclare(...)
```

Arguments

... the names of additional parameters, not in inverted commas. No values should be given.

Value

`RMdeclare` returns an object of class `RMmodel`

Note

Only scalars can be defined here, since only scalars can be used within formulae.

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

## The following two examples illustrate the use of RMdeclare and the
## argument 'params'. The purpose is not to give nice statistical models

x <- seq(1, 3, 0.1)

## note that there isn't any harm to declare variables ('u')
## RMdeclare that are of no use in a simulation
model <- ~ RMexp(sc=sc1, var=var1) + RMgauss(var=var2, sc=sc2) + RMdeclare(u)
p <- list(sc1=2, var1=3, sc2=4, var2=5)
z <- RFsimulate(model = model, x=x, y=x, params=p)
plot(z)

## note that the model remains the same, only the values in the
## following list change. Here, sc1, var1, sc2 and u are estimated
## and var2 is given by a formula.
p.fit <- list(sc1 = NA, var1=NA, var2=~2 * u, sc2 = NA, u=NA)
lower <- list(sc1=20, u=5)
upper <- list(sc2=1.5, sc1=100, u=15)
f <- RFfit(model, data=z, params=p.fit, lower = lower, upper = upper)
print(f)

## The second example shows that rather complicated constructions are
## possible, i.e., formulae involving several variables, both known ('abc')
## and unknown ones ('sc', 'var'). Note that there are two different
## 'var's a unknown variable and an argument for RMwhittle
## Not run:

model2 <- ~ RMexp(sc) + RMwhittle(var = g, nu=Nu) +
  RMnugget(var=nugg) + RMexp(var=var, Aniso=matrix(A, nc=2)) +
```

```

RMdeclare(CCC, DD)
p.fit <- list(g=~sc^1.5, nugg=~sc * var * abc, sc=NA, var=~DD, Nu=NA, abc=123,
  A = ~c(1, 2, DD * CCC, CCC), CCC = NA, DD=NA)
lower <- list(sc=1, CCC=1, DD=1)
upper <- list(sc=100, CCC=100, DD=100)
f2 <- Rffit(model2, data=z, params=p.fit, lower = lower, upper = upper)
print(f2)

## End(Not run)

```

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

$$C(h) = (C_0(h - s_i + s_j))_{i,j=0,\dots,m}$$

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 $s = (s_1, \dots, s_m)$ 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 a single univariate random field by shifting it by a fixed value.

Value

[RMdelay](#) returns an object of class [RMmodel](#).

Author(s)

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

References

- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Storkorb, K. (2015) Analysis, simulation and prediction of multivariate random fields with package **RandomFields**. *Journal of Statistical Software*, **63** (8), 1-25, url = 'http://www.jstatsoft.org/v63/i08/'
- Wackernagel, H. (2003) *Multivariate Geostatistics*. Berlin: Springer, 3rd 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")
```

RMderiv

Gradient of a field

Description

[RMderiv](#) is a multivariate covariance model which models a field and its gradient.

For an isotropic covariance model *varphi*, the covariance *C* given by [RMderiv](#) equals

$$C_{11}(x, y) = \varphi(\|x - y\|)$$

$$C_{j1}(x, y) = -C_{1j}(x, y) = \partial\varphi(\|x - y\|)/\partial x$$

$$C_{i,j}(x, y) = \partial^2\varphi(\|x - y\|)/\partial x\partial y$$

for $i, j = 2, \dots, d$ where d is the dimension of the field.

Usage

```
RMderiv(phi, which, var, scale, Aniso, proj)
```

Arguments

phi a univariate stationary covariance model (in 2 or 3 dimensions).
 which vector of integers. If not given all components are returned; otherwise the selected components are returned.
 var, scale, Aniso, proj optional arguments; same meaning for any [RMmodel](#). If not passed, the above covariance function remains unmodified.

Value

[RMderiv](#) returns an object of class [RMmodel](#).

Author(s)

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

References

- Matheron

See Also

[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

model <- RMderiv(RMgauss(), scale=4)
plot(model, dim=2)

x.seq <- y.seq <- seq(-10, 10, 0.4)
simulated <- RFsimulate(model=model, x=x.seq, y=y.seq)

plot(simulated)
```

 RMdewijsian

Modified De Wijsian 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 generalized 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>, <http://ms.math.uni-mannheim.de>

References

- Wackernagel, H. (2003) *Multivariate Geostatistics*. Berlin: Springer, 3rd 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, 0.02)
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 the following components:

- the potential
- the vector field given by

$$C(h) = (-\Delta E + \nabla \nabla^T) C_0(h)$$

- the curl field

Usage

```
RMdivfree(phi, which, var, scale, Aniso, proj)
```

Arguments

`phi` a univariate stationary covariance model (in 2 or 3 dimensions).
`which` vector of integers. If not given all components are returned; otherwise the selected components are returned.
`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>, <http://ms.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](#), [RMderiv](#), [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, 0.2)
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

$$f(h) = h^T A A^T h + \text{diag}(E)$$

and

$$f(h) = h^T A R R A^T h + \text{diag}(E)$$

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>, <http://ms.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

$$C(r) = (\epsilon + r^\alpha)^{-\beta/\alpha}$$

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 parameter 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>, <http://ms.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, 0.02)
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 analogue 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>

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, 0.02)
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>, <http://ms.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)
```

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>, <http://ms.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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMfixcov

Fixed Covariance Matrix

Description

[RMfixcov](#) is a user-defined covariance according to the given covariance matrix.

It extends to the space through a Voronoi tessellation.

Usage

```
RMfixcov(M, x, y=NULL, z=NULL, T=NULL, grid, var, proj, raw, norm)
```

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.
x,y,z,T,grid	optional. The usual arguments as in RFsimulate to define the locations where the covariates are given.
var,proj	optional arguments; same meaning for any RMmodel . If not passed, the above covariance function remains unmodified.
raw	logical. If FALSE then the data are interpolated. This approach is always save, but might be slow. If TRUE then the data may be accessed when covariance matrices are calculated. No rescaling or anisotropy definition is allowed in combination with the model. The use is dangerous, but fast. Default: FALSE.
norm	optional model that gives the norm between locations

Details

The covariances passed are implemented for the given locations. Within any Voronoi cell (around a given location) the correlation is assumed to be one.

In particular, it is used in [RFfit](#) to define neighbour or network structure in the data.

Value

RMfixcov returns an object of class `RMmodel`.

Note

Starting with version 3.0.64, the former argument element is replaced by the general option set in `RFoptions`.

Author(s)

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

`RMcovariate`, `RMmodel`, `RFsimulate`, `RFfit`, `RMuser`.

Examples

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

## Example 1 showing that the covariance structure is correctly implemented
n <- 10
C <- matrix(runif(n^2), nc=n)
(C <- C %*% t(C))
RFcovmatrix(RMfixcov(C), 1:n)

## Example 2 showing that the covariance structure is interpolated
RFcovmatrix(RMfixcov(C, 1:n), c(2, 2.1, 2.5, 3))

## Example 3 showing the use in a separable space-time model
model <- RMfixcov(C, 1:n, proj="space") * RMexp(s=40, proj="time")
(z <- RFsimulate(model, x = seq(0,12, 0.5), T=1:100))
plot(z)
```

 RMfixed

Fixed Effect Model

Description

Expressions of the form $X@RMfixed(beta)$ can be used within a formula of the type

$$response \ fixedeffects + randomeffects + errorterm$$

that specifies the Linear Mixed Model.

Important remark: `RMfixed` is NOT a function although the parentheses 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>, <http://ms.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>, <http://ms.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, 0.1)  
plot(model)  
plot(RFsimulate(model, x=x))
```

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>, <http://ms.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, 0.02)
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 <= 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>, <http://ms.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, 0.02)
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>, <http://ms.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 generalizations;
[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, 0.02)
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 parameter 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>, <http://ms.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 $\alpha \in (0, 1]$)

- Storkorb, 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, 0.02)
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>, <http://ms.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, 0.02)
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 being 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](#),

$$C(r) = (1 - r)^\beta 1_{[0,1]}(r), \quad \beta = \mu + 1/2 = \mu + 2\kappa + 1/2.$$

For $\kappa = 1$ the Gneiting model is given by

$$C(r) = (1 + \beta r) (1 - r)^\beta 1_{[0,1]}(r), \quad \beta = \mu + 2\kappa + 1/2.$$

If $\kappa = 2$

$$C(r) = \left(1 + \beta r + \frac{\beta^2 - 1}{3} r^2\right) (1 - r)^\beta 1_{[0,1]}(r), \quad \beta = \mu + 2\kappa + 1/2.$$

In the case $\kappa = 3$

$$C(r) = \left(1 + \beta r + \frac{(2\beta^2 - 3)}{5} r^2 + \frac{(\beta^2 - 4)\beta}{15} r^3\right) (1 - r)^\beta 1_{[0,1]}(r), \quad \beta = \mu + 2\kappa + 1/2.$$

A special case of this model is [RMgneiting](#).

Usage

```
RMgengneiting(kappa, mu, var, scale, Aniso, proj)
```

Arguments

`kappa` 0, ..., 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>, <http://ms.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, 0.02)
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)
```

RMgennsst

Non-Separable Space-Time model

Description

RMgennsst is a univariate stationary spaceisotropic covariance model on R^d whose corresponding covariance is given by

$$C(h, u) = \phi(\text{sqr}t(h^\top \psi^{-1}(u)h)) / \sqrt{\text{det}(\psi)}$$

Usage

```
RMgennsst(phi, psi, dim_u, var, scale, Aniso, proj)
```

Arguments

phi is a normal mixture **RMmodel**, cf. `RFgetModelNames(monotone="normal mixture")`.

psi is a d -variate variogram model **RMmodel**.

dim_u the dimension of the component u

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

RMgennsst returns an object of class **RMmodel**.

Author(s)

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

References

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

See Also

[RMnsst](#), [RMmodel](#), [RFsimulate](#), [RFfit](#).

Examples

 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 $0 \leq r \leq \frac{1}{s}$ 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 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>, <http://ms.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, 0.2) ## nicer with 0.1 instead of 0.2
z <- RFsimulate(model, x=x, y=x, z=x, T=c(1,1,4), maxGB=3)
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 model `RMwhittle`.

Usage

```
RMgneitingdiff(nu, taper.scale, scale, var, Aniso, proj)
```

Arguments

nu see [RMwhittle](#)

taper.scale is the parameter t in the above formula.

scale is the parameter 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>, <http://ms.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, 0.02)
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 modified 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 the 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>, <http://ms.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, 0.02)
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>, <http://ms.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=1.5, delta=0.5)
plot(model, dim=2)

x <- seq(0, 10, 0.1)
plot(RFsimulate(model, x=x, y=x))
```

RMid

Identical Model

Description

RMid is the identical function $f(x) = x$ where x is a vector of coordinates and $f(x)$ is a model value.

Usage

```
RMid()
```

Value

[RMid](#) returns an object of class [RMmodel](#).

Author(s)

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

See Also

[RMmodel](#). [RMid](#), [RMtrafo](#), [RMprod](#)

Examples

```

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

## C(x,y) = < x, y >
RFcov(RMprod(RMid()), as.matrix(1:10), as.matrix(1:10), grid=FALSE)

```

RMidmodel

Identical Model

Description

RMidmodel is the identical operator on models, i.e. for objects of class [RMmodel](#).

Usage

```
RMidmodel(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

[RMidmodel](#) returns an object of class [RMmodel](#).

Author(s)

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

See Also

[RMmodel](#). [RMid](#), [RMtrafo](#), [RMprod](#)

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 <- RMidmodel(model)
z2 <- RFsimulate(model, x)
sum(abs(as.vector(z)- as.vector(z2))) == 0 # TRUE

```

RMIntern

Internal models

Description

Internal models or model names that may appear in feedbacks from 'RandomFields'. Those endings by 'Intern' should appear only in very rare cases.

Details

The following and many more internal models exist:

- RF__Name__ : internal representation of certain functions [RF__name__](#)
- R0# : model for transforming coordinates within the cartesian system
- R0> : model for transforming earth coordinates to cartesian coordinates
- R0missing : 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, RPplus, 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>

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 'RPplus' appears as internal model
x <- seq(0, 10, 1)
z <- RFsimulate(RPspecific(RMexp() + RMnugget()), x)
RFgetModelInfo(which="internal", level=0)

```

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>, <http://ms.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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

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

$$C(r) = a_0 + a_2 r^2 + \phi(r), 0 \leq r \leq \text{diameter}$$

$$C(r) = b_0 (\text{rawRD} - r)^3 / (r), \text{diameter} \leq r \leq \text{rawR} * \text{diameter}$$

$$C(r) = 0, \text{rawR} * \text{diameter} \leq r$$

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>, <http://ms.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 \mathbb{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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMkolmogorov

Identical Model

Description

RMkolmogorov corresponds to a vector-valued random field 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>, <http://ms.math.uni-mannheim.de>

References

The above formula is from eq. (6.32) of section 6.2 in
 Pope, S.B. (2000) *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=20)
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 positive number

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>, <http://ms.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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMLsfbm	<i>Locally Positive Definite Function Given by the Fractal Brownian Motion</i>
---------	--

Description

RMLsfbm is a positive definite function on the unit ball in R^d centred at the origin,

$$C(r) = c - r^\alpha$$

with $r = \|x - y\| \in [0, 1]$.

Usage

RMLsfbm(alpha, const, var, scale, Aniso, proj)

Arguments

alpha numeric in $(0, 2)$; refers to the fractal dimension of the process.

const the constant c is given by the formula

$$c = 2^{-\alpha} \Gamma(d/2 + \alpha/2) \Gamma(1 - \alpha/2) / \Gamma(d/2)$$

and should not be changed by the user in order to ensure positive definiteness.

var, scale, Aniso, proj

optional arguments; same meaning for any [RMmodel](#). If not passed, the above covariance function remains unmodified.

Value

RMLsfbm returns an object of class [RMmodel](#).

Author(s)

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

References

- Martini, J., Schlather, M., Simianer, H. (In preparation.)

See Also

[Rmbcw](#) generalizes RMLsfbm in case that c is given, [RMfbm](#), [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 <- RMIsfbm(alpha=1, scale=10)
x <- seq(0, 10, 0.02)
plot(model, xlim=c(0,10))
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>, <http://ms.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, 0.02)
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 preceding 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 the left on the above matrix and the first elements of the obtained vector are interpreted 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>, <http://ms.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, 0.1)
plot(RFsimulate(model, x=x, y=x))
```

RMmatrix *Matrix operator*

Description

`RMmatrix` is a multivariate covariance model depending on one multivariate covariance model, or one or several univariate covariance models C_0, \dots . The corresponding covariance function is given by

$$C(h) = M\phi(h)M^t$$

if a multivariate case is given. Otherwise it returns a matrix whose diagonal elements are filled with the univariate model(s) C_0, C_1 , etc, and the offdiagonals are all zero.

Usage

```
RMmatrix(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, M, vdim,
         var, scale, Aniso, proj)
```

Arguments

`C0` a k-variate covariance `RMmodel` or a univariate model or a list of models joined by `c`

`C1, C2, C3, C4, C5, C6, C7, C8, C9`
 optional univariate models

`M` a k times k matrix, which is multiplied from left and right to the given model; M may depend on the location, hence it is then a matrix-valued function and C will be non-stationary with

$$C(x, y) = M(x)\phi(x, y)M(y)^t$$

`vdim` positive integer. This argument should be given if and only if a multivariate model is created from a single univariate model and M is not given. (In fact, if M is given, `vdim` must equal the number of columns of M)

`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`.

Note

- `RMmatrix` also allows variogram models as arguments.

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

## Not run:
## first example: bivariate Linear Model of Coregionalisation
x <- y <- seq(0, 10, 0.2)

model1 <- RMmatrix(M = c(0.9, 0.43), RMwhittle(nu = 0.3)) +
  RMmatrix(M = c(0.6, 0.8), RMwhittle(nu = 2))
plot(model1)
simu1 <- RFsimulate(RPdirect(model1), x, y)
plot(simu1)

## second, equivalent way of defining the above model
model2 <- RMmatrix(M = matrix(ncol=2, c(0.9, 0.43, 0.6, 0.8)),
  c(RMwhittle(nu = 0.3), RMwhittle(nu = 2)))
simu2 <- RFsimulate(RPdirect(model2), x, y)
stopifnot(all.equal(as.array(simu1), as.array(simu2)))

## third, equivalent way of defining the above model
model3 <- RMmatrix(M = matrix(ncol=2, c(0.9, 0.43, 0.6, 0.8)),
  RMwhittle(nu = 0.3), RMwhittle(nu = 2))
simu3 <- RFsimulate(RPdirect(model3), x, y)
stopifnot(all(as.array(simu3) == as.array(simu2)))

## End(Not run)

## second example: bivariate, independent fractional Brownian motion
## on the real axis
x <- seq(0, 10, 0.1)
modelB <- RMmatrix(c(RMfbm(alpha=0.5), RMfbm(alpha=1.5))) ## see the Note above
print(modelB)
simuB <- RFsimulate(modelB, x)
plot(simuB)

## third example: bivariate non-stationary field with exponential correlation
## function. The variance of the two components is given by the
## variogram of fractional Brownian motions.
## Note that the two components have correlation 1.
x <- seq(0, 10, 0.1)
modelC <- RMmatrix(RMexp(), M=c(RMfbm(alpha=0.5), RMfbm(alpha=1.5)))
print(modelC)
simuC <- RFsimulate(modelC, x, x, print=1)
#print(as.vector(simuC))

```

```
plot(simuC)
```

RMmodel	<i>Covariance and Variogram Models in RandomFields (RM commands)</i>
---------	---

Description

Summary of implemented covariance and variogram models

Details

To generate a covariance or variogram model for use within **RandomFields**, calls of the form

$$RM_{name}(\dots, var, scale, Aniso, proj)$$

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.

With ϕ denoting the original model, the transformed model is $C(h) = v * \phi(A * h/s)$. See [RMS](#) for more details.

`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

Others

`RMtrend` trend
`RMangle` defines a 2x2 anisotropy matrix by rotation and stretch arguments.

Author(s)

Alexander Malinowski; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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, 3rd edition.

See Also

`RM` for an overview over more advanced classes of models
`RC`, `RF`, `RP`, `RR`, `R.`, `RFcov`, `RFformula`, `RMmodelsAdvanced`, `RMmodelsAuxiliary`, `trend modelling`

Examples

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

## 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 **RandomFields**' representation of explicit covariance models

Usage

```
RFplotModel(x, y, dim=1,
            n.points=if (dim==1 || is.contour) 200 else 100,
            fct.type=NULL, MARGIN, fixed.MARGIN, maxchar=15, ...,
            plotmethod=if (dim==1) "matplot" else "contour")
```

```
## S4 method for signature 'RMmodel,missing'
```

```
plot(x, y, ...)
```

```
## S4 method for signature 'RMmodel'
```

```
points(x, ..., type="p")
```

```
## S4 method for signature 'RMmodel'
```

```
lines(x, ..., type="l")
```

```
## S4 method for signature 'RMmodel'
```

```
image(x, ..., dim=2)
```

```
## S4 method for signature 'RMmodel'
```

```
persp(x, ..., dim=2, zlab="")
```

Arguments

x	object of class <code>RFsp</code> or <code>RFempVario</code> or <code>RFfit</code> or <code>RMmodel</code> ; 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. whether the field or the covariance model is to be plotted; in all other directions, the first index is taken.
fixed.MARGIN	only for <code>class(x)==CLASS_CLIST</code> and if <code>dim > 2</code> ; a vector of length <code>dim-2</code> with distance values for the coordinates that are not displayed.
maxchar	integer. Maximum number of characters to print the model in the legend.
...	arguments to be passed to methods; mainly graphical arguments, or further models in case of class <code>CLASS_CLIST</code> , see Details.
dim	must equal 1 or 2; only for <code>class(x)==CLASS_CLIST</code> ; the covariance function and the variogram are plotted as a function of \mathbb{R}^{dim} .

<code>n.points</code>	integer; only for <code>class(x)==CLASS_CLIST</code> ; the number of points at which the model is evaluated (in each dimension); defaults to 200.
<code>fct.type</code>	character; only for <code>class(x)==CLASS_CLIST</code> ; must equal <code>NULL</code> , <code>"Cov"</code> or <code>"Variogram"</code> ; controls whether the covariance (<code>fct.type="Cov"</code>) or the variogram (<code>fct.type="Variogram"</code>) is plotted; <code>NULL</code> implies automatic choice, where <code>"Cov"</code> is chosen whenever the model is stationary.
<code>plotmethod</code>	string or function. Internal.
<code>type</code>	character. See points .
<code>zlab</code>	character. See persp .

Value

If `RFOptions()$split_screen=TRUE` and `RFOptions()$close_screen=TRUE` then the plot functions return the screen numbers. Else `NULL`.

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; contains 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 = CLASS_CLIST`): allows to sum up covariance models; internally calls [RMplus](#).
`-` signature(`x = CLASS_CLIST`): allows to subtract covariance models; internally calls [R.minus](#).
`*` signature(`x = CLASS_CLIST`): allows to multiply covariance models; internally calls [R.mult](#).
`/` signature(`x = CLASS_CLIST`): allows to divide covariance models; internally calls [R.div](#).
`c` signature(`x = CLASS_CLIST`): concatenates covariance functions or variogram models.
plot signature(`x = CLASS_CLIST`): gives a plot of the covariance function or of the variogram model; for more details see [plot-method](#).
points signature(`x = CLASS_CLIST`): adds a covariance plot to an existing plot; for more details see [plot-method](#).
lines signature(`x = CLASS_CLIST`): adds a covariance plot to an existing plot; for more details see [plot-method](#).
str signature(`x = CLASS_CLIST`): as the usual `str`-method for S4 objects but only those entries of the `'par.general'`-slot are shown that contain values different from `'RFdefault'`.

show signature(x = CLASS_CLIST): returns the structure of x.

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

[signature(x = CLASS_CLIST): enables accessing the slots via the "["-operator, e.g. x["par.general"].

[<- signature(x = CLASS_CLIST): enables replacing the slots via the "["-operator.

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

Details

All the above arguments apply for all the S3 and S4 functions given here as they call RFplotModel immediately.

Author(s)

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

Description

Extension of class [RMmodel](#) which additionally contains the likelihood of the data w.r.t. the covariance model represented by the CLASS_CLIST 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 meant to be created by the function `RFFit`.

Slots

`model,formel`: See `RMmodel`.
`variab`: vector of estimated variables. Variables are used in the internal representation and can be a subset of the parameters.
`param`: vector of estimated parameters
`covariate`: to do
`globalvariance`: to do
`hessian`: to do
`likelihood`: numeric; the likelihood of the data w.r.t. the covariance model
`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
`residuals`: array or of class `RFsp`; residuals of the data w.r.t. the trend model

Extends

Class `CLASS_CLIST`, directly.

Methods

`[` signature(`x = CLASS_FIT`): enables accessing the slots via the `"["`-operator, e.g. `x["likelihood"]`
`[<-` signature(`x = CLASS_FIT`): 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; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>

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 function that generates an object of class [RMmodel](#)
type: character string; specifies the category 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 field at a single fixed location, equals 1 in most cases, see [Details](#)

Extends

Class [function](#), directly.

Methods

show signature(x = CLASS_CLIST): returns the structure of x
print signature(x = CLASS_CLIST): identical with show-method
[signature(x = CLASS_RM): enables accessing the slots via the "["-operator, e.g. x["maxdim"]
[<- signature(x = CLASS_RM): 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 distribution, 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. a non-stationary covariance function
- 'framework dependent': domain depends on the calling model
- 'mismatch': this option is used only internally 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 $f(x) = f(-x)^T$ 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 internally and should never appear

operator: if TRUE, the model requires at least one submodel

monotone: 'mismatch in monotonicity': used if a statement on the monotonicity 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 categories; either the function is not monotone or properties are unknown
 '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; maxdim=-1 means that the actual maxdim depends on the parameters; maxdim=-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, Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>

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()

```

Description

Various classes of models RMxxx are implemented in RandomFields, that have their own man pages. Here, an overview over these man pages are given.

Man pages

Beginners should start with [RMmodels](#), then go for [RMmodelsAdvanced](#) if more information is needed.

RMmodels	general introduction and a collection of simple models
RMmodelsAdvanced	includes more advanced stationary and isotropic models, variogram models, non-stationary models
Bayesian	hierarchical models
RMmodelsMultivariate	multivariate covariance models and multivariate trend models
RMmodelsNonstationary	non-stationary covariance models
RMmodelsSpaceTime	space-time covariance models
Spherical models	models based on the polar coordinate system, usually used in earth models
Tail correlation functions	models related to max-stable random fields
trend modelling	how to pass trend specifications
Mathematical functions	simple mathematical functions that are typically used to build non-stationary covariance models
RMmodelsAuxiliary	rather specialized models, most of them not having positive definiteness property, but used internally

Author(s)

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

See Also

[RC](#), [RR](#), [RF](#), [R](#).

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) ## RMmodel.Rd
```


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)
RMbcw	bridging model between RMcauchy and RMgenfbm
RMBessel	Bessel family
RMcircular	circular model
RMconstant	spatially constant model
RMcubic	cubic model (see Chiles and 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	generalized hyperbolic model
RMIgd	Gneiting's local-global distinguisher
RMIlsfbm	locally stationary fractal Brownian motion
RMpenta	penta model (see Chiles and Delfiner)
RMPower	Golubov's model
RMwave	cardinal sine

Variogram models (stationary increments/intrinsically stationary)

RMbcw	bridging model between RMcauchy and RMgenfbm
RMDewijsian	generalized 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-arithmetic-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
RMtbn	Turning bands operator

Stationary space-time models

See [RMmodelsSpaceTime](#).

Non-stationary models

See [RMmodelsNonstationary](#).

Negative definite models that are not variograms

RMsum	a non-stationary variogram model
-----------------------	----------------------------------

Models related to max-stable random fields (tail correlation functions)

See [RMmodelsTailCorrelation](#).

Other covariance models

RMcov	covariance structure given by a variogram
RMfixcov	User defined covariance structure
RMuser	User defined model

Trend models

Aniso	for space transformation (not really trend, but similar)
RMcovariate	spatial covariates
RMprod	to model variability of the variance
Rmpolynome	easy modelling of polynomial trends
RMtrend	for explicit trend modelling
R.models	for implicit trend modelling
R.c	for multivariate trend modelling

Auxiliary models

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 a length equal to the number of arguments.
- If an argument equals `NULL` the argument is not set (but must have a valid name).
- `Aniso` can be given also by `RMangle` or any other `RMmodel` instead of 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
  print(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)) ).

```

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; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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.
- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Strokorb, K. (2015) Analysis, simulation and prediction of multivariate random fields with package **RandomFields**. *Journal of Statistical Software*, **63** (8), 1-25, url = '<http://www.jstatsoft.org/v63/i08/>' '[multivariate](#)', the corresponding vignette.
- 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, 3rd edition.

See Also

[RFformula](#), [RM](#), [RMmodels](#), [RMmodelsAuxiliary](#).

Examples

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

## a non-stationary field with a sharp boundary
## of the differentiabilitys
x <- seq(-0.6, 0.6, len=50)
model <- RMwhittle(nu=0.8 + 1.5 * R.is(R.p(new="isotropic"), "<=", 0.5))
z <- RFsimulate(model=model, x, x, n=4)
plot(z)
```

RMmodelsMultivariate *Multivariate models*

Description

Here, multivariate and vector-valued covariance models are presented.

Details**Bivariate covariance models**

Rmbicauchy	a bivariate Cauchy model
Rmbiwm	full bivariate Whittle-Matern model (stationary and isotropic)
Rmbigneiting	bivariate Gneiting model (stationary and isotropic)
Rmbistable	a bivariate stable model

Physically motivated, vector valued covariance and variogram models

RMcurlfree	curlfree (spatial) vector-valued field (stationary and anisotropic)
RMdivfree	divergence free (spatial) vector-valued field (stationary and anisotropic)
RMkolmogorov	Kolmogorov's model of turbulence
RMvector	vector-valued field (combining RMcurlfree and RMdivfree)

Multivariate covariance models

RMdelay	delay effect model (stationary)
RMderiv	field and its gradient
RMmatrix	linear model of coregionalization
RMparswm	multivariate Whittle-Matern model (stationary and isotropic)

Operators

<code>RMcov</code>	covariance structure given by a multivariate variogram
<code>RMexponential</code>	functional returning e^C
<code>RMmatrix</code>	linear model of coregionalization
<code>RMmqam</code>	multivariate quasi-arithmetic mean (stationary)
<code>RMschur</code>	element-wise product with a positive definite matrix
<code>RMtbm</code>	turning bands operator

Trend models

<code>RMtrend</code>	for explicit trend modelling
<code>R.models</code>	for implicit trend modelling
<code>R.c</code>	binding univariate trend models into a vector

Author(s)

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

References

- Chiles, J.-P. and Delfiner, P. (1999) *Geostatistics. Modeling Spatial Uncertainty*. New York: Wiley.
- 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.
- Schlather, M., Malinowski, A., Menck, P.J., Oesting, M. and Storkorb, K. (2015) Analysis, simulation and prediction of multivariate random fields with package **RandomFields**. *Journal of Statistical Software*, **63** (8), 1-25, url = 'http://www.jstatsoft.org/v63/i08/'
- Wackernagel, H. (2003) *Multivariate Geostatistics*. Berlin: Springer, 3rd edition.

See Also

`RFformula`, `RMmodels`, `RM`, `RMmodelsAdvanced`
 ‘`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

n <- 100
x <- runif(n=n, min=1, max=50)
y <- runif(n=n, min=1, max=50)
```

```

rho <- matrix(nc=2, c(1, -0.8, -0.8, 1))
model <- RmparswmX(nudiag=c(0.5, 0.5), rho=rho)

## generation of artificial data
dta <- RFsimulate(model = model, x=x, y=y, grid=FALSE)

## introducing some NAs ...
dta@data$variable1[1:10] <- NA
if (interactive()) dta@data$variable2[90:100] <- NA

plot(dta)

## co-kriging
x <- y <- seq(0, 50, 1)

k <- RFinterpolate(model, x=x, y=y, data= dta)
plot(k, dta)

## conditional simulation
z <- RFsimulate(model, x=x, y=y, data= dta) ## takes a while
plot(z, dta)

```

RMmodelsNonstationary *Non-stationary features of the models*

Description

Here, non-stationary covariance models are presented.

Details

Covariance models

[RMnonstwm](#)

[RMprod](#)

[Aniso](#)

scale, cf. [RMS](#), can be any non-negative function for any scale mixture model, such as the [whittle-matern](#)-classes, the [powered](#)

Trend models See [RMmodelsTrend](#).

Author(s)

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

See Also

[RFformula](#), [RMmodels](#), [RM](#), [RMmodelsAdvanced](#)

`'nonstationary'`, a vignette for non-stationary geostatistics

Examples

```
# to do
```

RMmodelsSpacetime *Space-time Covariance Models*

Description

Here, a collection of implemented space-time models is given.

Details

Stationary space-time models

Here, most of the models are composed models (operators). Note that in space-time modelling the argument `proj` may also take the values "space" for the projection on the space and "time" for the projection onto the time axis.

separable models	are easily constructed using <code>+</code> , <code>*</code> , and <code>proj</code> , see also the example below
<code>RMave</code>	space-time moving average model
<code>RMcoxisham</code>	Cox-Isham model
<code>RMcurlfree</code>	curlfree (spatial) field (stationary and anisotropic)
<code>RMdivfree</code>	divergence free (spatial) vector-valued field (stationary and anisotropic)
<code>RMgennsst</code>	generalization of Gneiting's non-separable space-time model
<code>RMiaco</code>	non-separable space-time model
<code>RMmastein</code>	Ma-Stein model
<code>RMnsst</code>	Gneiting's non-separable space-time model
<code>RMstein</code>	Stein's non-separable space-time model
<code>RMstp</code>	Single temporal process
<code>RMtbn</code>	Turning bands operator

Author(s)

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

References

- 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

[RFformula](#), [RM](#), [RMmodels](#), [RMmodelsAdvanced](#).

Examples

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

## multiplicative separable model with exponential model in space
## and Gaussian in time
model <- RMexp(proj = "space") * RMgauss(proj = "time")
x <- T <- seq(0, 10, 0.1)
z <- RFsimulate(model, x=x, T=T)
plot(z)

## additive separable model with exponential model in space
## and Gaussian in time. The structure is getting rather simple,
## see the function stopifnot below
model <- RMexp(proj = "space") + RMgauss(proj = "time")
x <- T <- seq(0, 10, 0.1)
z <- RFsimulate(model, x=x, T=T)
stopifnot(sum(abs(apply(apply(z, 1, diff), 1, diff))) < 1e-14)
plot(z)
```

RMmppplus

Mixture of shape functions

Description

[RMmppplus](#) is a multivariate covariance model which depends on up to 10 submodels C_0, C_1, \dots, C_9 .

Used together with [RPsmit](#), it allows 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

C_0 an [RMmodel](#)
 $C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9$ 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

`RMppplus` returns an object of class `RMmodel`.

Author(s)

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

`RMmqam` 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 `RMmodels`

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>, <http://ms.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

RFoptions(modus_operandi="sloppy")
model <- RMmqam(phi=RMgauss(),RMgauss(),RMexp(),theta=c(0.4, 0.6), scale=0.5)
x <- seq(0, 10, 0.02)
plot(model)
z <- RFsimulate(model=model, x=x)
plot(z)

RFoptions(modus_operandi="normal")
```

Description

`RMmult` is a multivariate covariance model which depends on up to 10 submodels C_0, C_1, \dots, C_9 . In general, realizations of the created `RMmodel` are pointwise products 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. $C_0 * C_1$.

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>, <http://ms.math.uni-mannheim.de>

See Also

`RMplus`, `RMmodel`, `RMprod`, `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)
```

 RMmultiquad

The Multiquadric Family Covariance Model on the Sphere

Description

[RMmultiquad](#) is an isotropic covariance model. The corresponding covariance function, the multi-quadric family, only depends on the angle $\theta \in [0, \pi]$ between two points on the sphere and is given by

$$\psi(\theta) = (1 - \delta)^{2*\tau} / (1 + \delta^2 - 2 * \delta * \cos(\theta))^\tau$$

where $\delta \in (0, 1)$ and $\tau > 0$.

Usage

```
RMmultiquad(delta, tau, var, scale, Aniso, proj)
```

Arguments

`delta` a numerical value in $(0, 1)$
`tau` a numerical value greater than 0
`var, scale, Aniso, proj` optional arguments; same meaning for any [RMmodel](#). If not passed, the above covariance function remains unmodified.

Details

Special cases (cf. Gneiting, T. (2013), p.1333) are known for fixed parameter $\tau = 0.5$ which leads to the covariance function called 'inverse multiquadric'

$$\psi(\theta) = (1 - \delta) / \sqrt{1 + \delta^2 - 2 * \delta * \cos(\theta)}$$

and for fixed parameter $\tau = 1.5$ which gives the covariance function called 'Poisson spline'

$$\psi(\theta) = (1 - \delta)^3 / (1 + \delta^2 - 2 * \delta * \cos(\theta))^{1.5}$$

For a more general form, see [RMchoquet](#).

Value

`RMmultiquad` returns an object of class `RMmodel`.

Author(s)

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

References

Gneiting, T. (2013) *Strictly and non-strictly positive definite functions on spheres Bernoulli*, **19**(4), 1327-1349.

See Also

`RMmodel`, `RFsimulate`, `RFfit`, `RMchoquet`, `spherical models`

Examples

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

RFoptions(coord_system="sphere")
model <- RMmultiquad(delta=0.5, tau=1)
plot(model, dim=2)

## the following two pictures are the same

x <- seq(0, 0.12, 0.01)
z1 <- RFsimulate(model, x=x, y=x)
plot(z1)

x2 <- x * 180 / pi
z2 <- RFsimulate(model, x=x2, y=x2, coord_system="earth")
plot(z2)

stopifnot(all.equal(as.array(z1), as.array(z2)))

RFoptions(coord_system="auto")
```

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>, <http://ms.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, 0.02)
plot(RMexp(), model=model)
RFcov(model, 1)
```

RMnonstwm

*Non-stationary Whittle-Matern Covariance Model***Description**

The non-stationary Whittle-Matern model 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 ν must be a positive function.

W_μ is the covariance of the [RMwhittle](#) model or the [RMmatern](#) model.

Details

The non-stationary Whittle-Matern models are obtained by the respective stationary model, replacing the real-valued argument for nu by a non-negative function.

Note

It cannot be checked whether nu only takes positive values. So the responsibility is completely left to the user.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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
x <- seq(-1.2, 1.2, len=50)
model <- RMwhittle(nu=RMgauss())

z <- RFsimulate(model=model, x, x, n=4)
plot(z)
```

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 a 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>, <http://ms.math.uni-mannheim.de>

References

- Gneiting, T. (1997) Normal scale mixtures and dual probability densities, *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

[RMgennsst](#), [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, 0.25)
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 $1, \dots, \text{vdim}$ by

$$C_{ij}(r) = \delta_{ij} 1_0(r),$$

where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise.

Usage

```
RMnugget(tol, vdim, var, Aniso, proj)
```

Arguments

tol	Only for advanced users. See RPnugget .
vdim	Must be set only for multivariate models (advanced).
var	optional argument; same meaning for any RMmodel . If not passed, the above covariance function remains unmodified.
Aniso,proj	(zonal modelling and repeated measurements(advanced)); see RPnugget for details.

Details

The nugget effect belongs to Gaussian white noise and is used for modeling measurement errors or to model spatial ‘nuggets’.

Value

[RMnugget](#) returns an object of class [RMmodel](#).

Author(s)

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

See Also

[RMmodel](#), [RFsimulate](#), [RFfit](#), [RPnugget](#) (advanced users).

Examples

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

x <- y <- 1:2
xy <- as.matrix(expand.grid(x, y)) ## we get 4 locations

## Standard use of the nugget effect
model <- RMnugget(var = 100)
RFcovmatrix(model, x=xy)
as.vector(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](#) is 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; the vector $(\nu_{11}, \nu_{22}, \dots)$. 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>, <http://ms.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)
plot(model)
x.seq <- y.seq <- seq(-10, 10, 0.1)
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 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

$$C(r) = \left(1 - \frac{22}{3}r^2 + 33r^4 - \frac{77}{2}r^5 + \frac{33}{2}r^7 - \frac{11}{2}r^9 + \frac{5}{6}r^{11}\right)1_{[0,1]}(r).$$

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 dimensions $d \leq 3$.

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>, <http://ms.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, 0.02)
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, \dots, C_9 . 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

C_0 an **RMmodel**.
 $C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9$ 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. $C_0 + C_1$.

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>, <http://ms.math.uni-mannheim.de>

See Also

[RMmult](#), [RMmodel](#), [RMsum](#), [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, "model1."=model2, type=c("p", "l"), pch=20, xlim=c(0,3)) # the same
```

RMpolygon

RMpolygon

Description

RMpolygon refers 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 hyperplane process creating the random shape function.
The default value is 1.

Author(s)

Felix Ballani, <https://tu-freiberg.de/fakult1/sto/ballani>
Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>

References

Poisson polygons / Poisson hyperplane tessellation

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

Poisson storm process

- Lantuejoul, 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
```

RMpolynome

Creating polynomial models

Description

Polynomials, mainly used in trend models, can be created easily with this function.

Usage

```
RMpolynome(degree, dim, value=NA, coordnames = c("x", "y", "z", "T"),
           proj=1:4)
```

Arguments

degree	degree of the polynome
dim	number of variables in the polynome
value	values of the coefficients. See Details.
coordnames	the names of the variables
proj	the projection to certain dimensions

Details

If the length of `value` is smaller than the number of monomials, the remaining terms are filled with NAs. If the length is larger, the vector is cut.

Value

[RMpolynome](#) returns an object of class [RMmodel](#).

Author(s)

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

See Also

[RMtrend](#), [RFfit](#).

Examples

```
## For examples see the help page of 'RFformula' ##

RMpolynome(1, 1)
RMpolynome(1, 2)
RMpolynome(2, 1)
RMpolynome(2, 2)
RMpolynome(3, 3)
```

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>, <http://ms.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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMprod

Plain scalar product

Description

[RMprod](#) is a non-stationary covariance model given by

$$C(x, y) = \langle \phi(x), \phi(y) \rangle$$

Usage

```
RMprod(phi, var, scale, Aniso, proj)
```

Arguments

`phi` any 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

In general, this model defines a positive definite kernel and hence a covariance model for all functions ϕ with values in an arbitrary Hilbert space. However, as already mentioned above, ϕ should stem from a covariance or variogram model, here.

Value

[RMprod](#) returns an object of class [RMmodel](#).

Note

Do not mix up this model with [RMmult](#).

See also RMS for a simple, alternative method to set an arbitrary, i.e. location dependent, univariate variance.

Author(s)

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

References

Wendland, H. *Scattered Data Approximation* (2005) Cambridge: Cambridge University Press.

See Also

[RMid](#), [RMid](#), [RMsum](#), [RMmodel](#), [RMmult](#).

Examples

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

```
RFcov(RMprod(RMid()), as.matrix(1:10), as.matrix(1:10), grid=FALSE)
```

```
## C(x,y) = exp(-||x|| + ||y||)
RFcov(RMprod(RMexp()), as.matrix(1:10), as.matrix(1:10), grid=FALSE)
```

```
## C(x,y) = exp(-||x / 10|| + ||y / 10 ||)
model <- RMprod(RMexp(scale=10))
x <- seq(0,10,len=100)
z <- RFsimulate(model=model, x=x, y=x)
plot(z)
```

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

$$C(h) = \phi\left(\sqrt{\sum_i \theta_i (\phi^{-1}(C_i(h)))^2}\right)$$

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 [RMmodels](#)

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>, <http://ms.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

[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

model <- RMqam(phi=RMgauss(), RMexp(), RMgauss(),
              theta=c(0.3, 0.7), scale=0.5)
x <- seq(0, 10, 0.02)
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>, <http://ms.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, 0.02)
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

<code>A</code>	a $d \times d$ matrix
<code>a</code>	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>, <http://ms.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-dimensional space.

Value

[RMrotat](#) and [RMrotation](#) return an object of class [RMmodel](#).

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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 Details. However, the following describes the arguments `var`, `scale`, `Aniso`, `proj` that are common to nearly all models. See [RMSadvanced](#) for advanced use of these arguments.

Usage

```
RMS(phi, var, scale, Aniso, proj, anisoT)
```

Arguments

<code>phi</code>	submodel
<code>var</code>	is the optional variance parameter v .
<code>scale</code>	scaling parameter s which is positive.
<code>Aniso</code>	matrix or RMmodel1 . The optional anisotropy matrix A , multiplied from the right by a distance vector x , i.e. Ax .
<code>proj</code>	is the optional projection vector which defines a diagonal matrix of zeros and ones and <code>proj</code> gives the positions of the ones (integer values between 1 and the dimension of x). It also allows for the values 'space' and 'time' in case of space-time modelling.
<code>anisoT</code>	the transpose of the anisotropy matrix B , multiplied from the left by a distance vector x , i.e. $x^T B$.

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 [RMmodel1](#).

Most users will never call **RMS** directly.

Value

RMS returns an object of class [RMmodel1](#).

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>, <http://ms.math.uni-mannheim.de>

See Also

[RMSadvanced](#), [RMmodel](#),

[RMprod](#) for an alternative way to define an arbitrary, location dependent variance. There, the standard deviation is given so that [RMprod](#) might be used even in the multivariate case.

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, 0.02)
print(all(RFcov(model1, x) == RFcov(model2, x))) # TRUE
```

RMSadvanced

Scaling operator – comments for advanced applications

Description

Here advances uses are given for the arguments `var`, `scale`, `Aniso`, `proj` that are available to most of the models

Usage

```
RMS(phi, var, scale, Aniso, proj, anisoT)
```

Arguments

phi submodel

var Instead of a constant it can be also an arbitrary non-negative function, see [R](#). and [RMuser](#) for defining arbitrary functions.

scale instead of a positive constant it can be an arbitrary, positive deterministic function. In case of the latter, the scale should be given by one of the functions [Rmbubble](#) or [RMscale](#). In case none of them are given, [RMscale](#) is assumed with scale penalty $\|s(x) - s(y)\|^2$ for the square of the norm.

The scale can be also a random variable in case of Bayesian modelling.

Aniso matrix or [RMmodel](#). Instead of a matrix, `Aniso` can be an arbitrary, vector-valued function .

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 the dimension of x). It also allows for the values 'space' and 'time' in case of space-time modelling.

anisoT the transpose of the anisotropy matrix B , multiplied from the left by a distance vector x , i.e. $x^\top B$.

Details

See the reference for Gneiting's nsst model used for modelling scales. See also the example below.

Value

`RMSadvanced` returns an object of class `RMmodel`.

Author(s)

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

References

- Bonat, W.H. , Ribeiro, P. Jr. and Schlather, M. (2019) Modelling non-stationarity in scale. In preparation.

See Also

`RMS`, `Rmblend` for a different approach on modelling different scales

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, if (interactive()) 0.01 else 0.5)
d <- sqrt(rowSums(as.matrix(expand.grid(x-0.5, x-0.5))^2))
d <- matrix(d < 0.25, nc=length(x))
image(d)

scale <- RMcovariate(data=as.double(d) * 2 + 0.5, raw=TRUE)

S <- RMexp(scale = scale)
plot(zS <- RFsimulate(S, x, x))
CS <- RFcovmatrix(S, x, x)
```

RMscale

Scale model for arbitrary areas of scales

Description

Let s_x the scaling at location x and p a bijective penalizing function for (different) scales. Then covariance function is given by

$$C(x, y) = \phi(\|x - y\| + |p(s_x) - p(s_y)|)$$

Usage

```
RMscale(phi, scaling, penalty, var, scale, Aniso, proj)
```

Arguments

phi	isotropic submodel
scaling	model that gives the non-stationary scaling s_x
penalty	bijective function p applied to the scaling
var, scale, Aniso, proj	optional arguments; same meaning for any RMmodel . If not passed, the above covariance function remains unmodified.

Value

[RMscale](#) returns an object of class [RMmodel](#).

Author(s)

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

References

- Bonat, W.H. , Ribeiro, P. Jr. and Schlather, M. (2019) Modelling non-stationarity in scale. In preparation.

See Also

[RMSadvanced](#), [RMblend](#), [Rmbubble](#)

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, 0.01)
scale <- RMcovariate(x=c(0,1), y=c(1,0),#2 areas separated by the 1st bisector
                    grid=FALSE, data=c(1, 3))

model <- RMscale(RMexp(), scaling = scale, penalty=RMid() / 2)
plot(z <- RFsimulate(model, x, x))
```

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>

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 example considers an extremal Gaussian random field
## with Gneiting's correlation function.

## first consider the covariance model and its corresponding tail
```

```

## correlation 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
x <- seq(0, 10, 0.1)
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 correlation
## 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 the 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>, <http://ms.math.uni-mannheim.de>

References

- ?

See Also

[RMmodel](#), [RMmatrix](#).

Examples

```
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##                RFoptions(seed=NA) to make them all random again
model <- RMSchur(M=matrix(c(2, 1, 1, 1), ncol=2), RMparswm(nudiag=c(0.5, 2)))
plot(model)
x <- seq(0, 10, 0.02)
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 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>, <http://ms.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, 0.02)
plot(RFsimulate(model, x=x))
```

RMsinepower

The Sinepower Covariance Model on the Sphere

Description

[RMsinepower](#) is an isotropic covariance model. The corresponding covariance function, the sine power function of Soubeyrand, Enjalbert and Sache, only depends on the angle $\theta \in [0, \pi]$ between two points on the sphere and is given by

$$\psi(\theta) = 1 - \left(\sin \frac{\theta}{2}\right)^\alpha$$

where $\alpha \in (0, 2]$.

Usage

```
RMsinepower(alpha, var, scale, Aniso, proj)
```

Arguments

alpha a numerical value in $(0, 2]$
var, scale, Aniso, proj optional arguments; same meaning for any [RMmodel](#). If not passed, the above covariance function remains unmodified.

Details

For the sine power function of Soubeyrand, Enjalbert and Sache, see Gneiting, T. (2013), equation (17). For a more general form see [RMchoquet](#).

Value

`RMsinepower` returns an object of class `RMmodel`.

Author(s)

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

References

Gneiting, T. (2013) *Strictly and non-strictly positive definite functions on spheres Bernoulli*, **19**(4), 1327-1349.

See Also

`RMmodel`, `RFsimulate`, `RFfit`, `spherical models`, `RMchoquet`

Examples

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

RFOptions(coord_system="sphere")
model <- RMsinepower(alpha=1.7)
plot(model, dim=2)

## the following two pictures are the same
x <- seq(0, 0.4, 0.01)
z1 <- RFsimulate(model, x=x, y=x)
plot(z1)

x2 <- x * 180 / pi
z2 <- RFsimulate(model, x=x2, y=x2, coord_system="earth")
plot(z2)

stopifnot(all.equal(as.array(z1), as.array(z2)))

RFOptions(coord_system="auto")
```

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](#).

Note

Although this model is valid on a sphere, do not mix up this model with valid models on a sphere; see [spherical models](#) for a list of the latter.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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](#), [spherical models](#).

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, 0.02)
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](#)) for $\alpha = 1$ 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>, <http://ms.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 $\alpha \in (0, 1]$)

- Storkorb, 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

[RMBistable](#), [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, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMstein

Stein's non-separable 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

$$C(h, t) = W_\nu(y) - (\langle h, z \rangle t) / ((\nu - 1)(2\nu + d)) * W_{\nu-1}(y)$$

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>, <http://ms.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, 0.05)
plot(RFsimulate(model, x=x, y=x))
```

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

$$\begin{aligned} Q(x, y) &= c^2 - m^2 + h^t(S_x + 2(m + c)M)A^{-1}(S_y + 2(m - c)M)h, \\ c &= -z^t h + \xi_2(x) - \xi_2(y), \\ A &= S_x + S_y + 4Mh h^t M \\ m &= h^t M h \\ h &= x - y \end{aligned}$$

Usage

```
RMstp(xi, phi, S, z, M, var, scale, Aniso, proj)
```

Arguments

<code>xi</code>	arbitrary univariate function on R^d
<code>phi</code>	an <code>RMmodel</code> that is a normal mixture model, cf. <code>RFgetModelNames(monotone="normal mixture")</code>
<code>S</code>	functions that returns strictly positive definite $d \times d$
<code>z</code>	arbitrary vector, $z \in R^d$
<code>M</code>	an arbitrary, symmetric $d \times d$ matrix
<code>var, scale, Aniso, proj</code>	optional arguments; same meaning for any <code>RMmodel</code> . 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>, <http://ms.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, 0.7)
plot(RFsimulate(model, x=x, y=x, z=x))
```

RMsum

Plain scalar product

Description

[RMsum](#) is given by

$$C(x, y) = \phi(x) + \phi(y)$$

It is a negative definite function although not a variogram.

Usage

```
RMsum(phi, var, scale, Aniso, proj)
```

Arguments

`phi` any function of class [RMmodel](#)
`var, scale, Aniso, proj` optional arguments; same meaning for any [RMmodel](#). If not passed, the above covariance function remains unmodified.

Value

[RMsum](#) returns an object of class [RMmodel](#).

Note

Do not mix up this model with [RMplus](#).

Author(s)

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

See Also

[RMmodel](#), [RMplus](#), [RMprod](#).

Examples

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

RMtbn

Turning Bands Method

Description

[RMtbn](#) is a univariate or multivariate stationary isotropic covariance model in dimension `reduceddim` which depends on a univariate or multivariate stationary isotropic covariance ϕ in a bigger dimension `fulldim`. For formulas for the covariance function see details.

Usage

```
RMtbn(phi, fulldim, reduceddim, layers, var, scale, Aniso, proj)
```

Arguments

`phi`, `fulldim`, `reduceddim`, `layers`

See [Rptbn](#).

`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 is `reduceddim=1` and `fulldim=3`. If only one of the arguments is given, then the difference of the two arguments equals 2.

For $d = n + 2$, where $n = \text{reduceddim}$ and $d = \text{fulldim}$ the original dimension, we have

$$C(r) = \phi(r) + r\phi'(r)/n$$

which for $n=1$ reduces 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

`RMtbn` returns an object of class `RMmodel`.

Author(s)

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

`RPtbn`, `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, 0.02)
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 coordinate system. 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(phi, new)
RFearth2cartesian(coord, units=NULL, system="cartesian", grid=FALSE)
RFearth2dist(coord, units=NULL, system="cartesian", grid=FALSE, ...)
```

Arguments

new	integer or character. One of the values RC_ISOTROPIC , RC_SPACEISOTROPIC , RC_CARTESIAN_COORD , RC_GNOMONIC_PROJ , RC_ORTHOGRAPHIC_PROJ or the corresponding RC_ISONAMES . Note that RMtrafo only allows for integer values. Default: RC_CARTESIAN_COORD .
phi	optional submodel
coord	matrix or vector of earth coordinates
units	"km" or "miles"; if not given and <code>RFoptions()\$general\$units != ""</code> , the latter is used. Otherwise "km".
system	integer or character. The coordinate system, e.g. "cartesian", "gnomonic" or "orthographic".
grid	logical. Whether the given coordinates are considered to be on a grid given by <code>c(start, step, length)</code> . Default: FALSE.
...	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 correspondence 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`.

Note also that the `zenit` must be given explicitly for projection onto a plane. See the examples below.

Author(s)

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

References

For calculating the earth coordinates as ellipsoid:

- https://en.wikipedia.org/wiki/Geographic_coordinate_system
- <https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html>

See Also

[constants](#), [RMangle](#), [RMid](#), [RMidmodel](#).

Examples

```
data(weather)
(coord <- weather[1:5, 3:4])

(z <- RFfctn(RMtrafo(new=RC_CARTESIAN_COORD), coord))
(z1 <- RFearth2cartesian(coord)) ## equals z
z1 - z ## 0, i.e., z1 and t(z) are the same
dist(z)

(d <- RFearth2dist(coord))
d - dist(z) ## 0, i.e., d and dist(z) are the same

## projection onto planes
RFOptions(zenit=c(-122, 47))
RFearth2cartesian(coord, system="gnomonic")
```

```
RFearth2cartesian(coord, system="orthographic")
```

RMtrend

Trend Model

Description

`RMtrend` is a pure trend model with covariance 0.

Usage

```
RMtrend(mean)
```

Arguments

`mean` numeric or `RMmodel`. If it is numerical, it should be a vector of length p , where p is the number of variables taken into account by the corresponding multivariate random field $(Z_1(\cdot), \dots, Z_p(\cdot))$; the i -th component of `mean` is interpreted as constant mean of $Z_i(\cdot)$.

Details

Note that this function refers to trend surfaces in the geostatistical framework. Fixed effects in the mixed models framework are also being implemented, see `RFformula`.

Value

`RMtrend` returns an object of class `RMmodel`.

Note

Using uncapsulated subtraction to build up a covariance function is ambiguous, see the examples below. Best to define the trend separately, or to use `R.minus`.

Author(s)

Marco Oesting, <oesting@mathematik.uni-siegen.de>, <https://www.uni-siegen.de/fb6/src/scheffler/mitarbeiter/oesting>; Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.math.uni-mannheim.de>

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

## first simulate some data with a sine and a mean as trend
repet <- 100

x <- seq(0, pi, len=10)
trend <- 2 * sin(R.p(new="isotropic")) + 3
model1 <- RMexp(var=2, scale=1) + trend
dta <- RFsimulate(model1, x=x, n=repet)

## now, let us estimate variance, scale, and two parameters of the trend
model2 <- RMexp(var=NA, scale=NA) + NA * sin(R.p(new="isotropic")) + NA

print(RFfit(model2, data=dta))

## model2 can be made explicit by enclosing the trend parts by
## 'RMtrend'
model3 <- RMexp(var=NA, scale=NA) + NA *
          RMtrend(sin(R.p(new="isotropic"))) + RMtrend(NA)
print(RFfit(model2, data=dta))

## IMPORTANT: subtraction is not a way to combine definite models
##           with trends
trend <- -1
(model0 <- RMexp(var=0.4) + trend) ## exponential covariance with mean -1
(model1 <- RMexp(var=0.4) + -1)   ## same as model0
(model2 <- RMexp(var=0.4) + RMtrend(-1)) ## same as model0
(model3 <- RMexp(var=0.4) - 1) ## this is a purely deterministic model
##           ## with exponential trend
plot(RFsimulate(model=model0, x=x, y=x)) ## exponential covariance
##           ## and mean -1
plot(RFsimulate(model=model1, x=x, y=x)) ## dito
plot(RFsimulate(model=model2, x=x, y=x)) ## dito
plot(RFsimulate(model=model3, x=x, y=x)) ## purely deterministic model!

```

RMtruncsupport

Truncating the Support of a Shape Function

Description

`RMtruncsupport` may be used to truncate the support of a shape function when Poisson fields or M3 processes are created.

Usage

```
RMtruncsupport(phi, radius)
```

Arguments

<code>phi</code>	function of class <code>RMmodel</code>
<code>radius</code>	truncation at radius

Value

`RMtruncsupport` returns an object of class `RMmodel`.

Author(s)

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

model <- RMgauss()
model1 <- RMtruncsupport(model, radius=1)
plot(model)
lines(model1, col="red")

## For a real application of 'RMtruncsupport' see example 2 of 'RPpoisson'.
```

 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,
       coordnames = 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: <ul style="list-style-type: none"> • '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.
coordnames	Just the names of the variables. More variable names might be given here than used in the function. See Details for the interpretation of variables.
fctn, fst, snd	a user-defined function and its first, second and third derivative, given as <code>quote(myfunction(x))</code> or as <code>quote(myfunction(x,y))</code> , see Details and 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 coordnames.

A kernel should depend on the first two arguments given by coordnames.

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 user-defined functions, the models of **RandomFields** are not recognized, 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>

See Also

`RMcovariate`, `RMfixcov`, `RFfit`, `RMmodelgenerator`, `RMmodel`, `RFsimulate`, `RC_ISO_NAMES`, `RC_DOMAIN_NAMES`.

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=100)
plot(model)
z <- RFsimulate(model, x=x, y=y)
plot(z)

## the kernel, which is the scalar product (see RMprod)
model <- RMnugget(var=1e-5) +
```

```

      RMuser(type="positive definite", domain="kernel",
             iso="symmetric", fctn=sum(x * y))
x <- y <- seq(1, 10, len=35)
z <- RFsimulate(model, x=x, y=y, n=6, svdtol=1e-9)
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 <i>Dspace</i> 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 spatio-temporal 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>, <http://ms.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, 0.4)
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 dimensions $d \leq 3$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = \sin(r)/r 1_{r>0} + 1_{r=0}.$$

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 only valid for dimensions $d \leq 3$. 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>, <http://ms.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, 0.02)
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

$$C(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu}r)^\nu K_\nu(\sqrt{2\nu}r)$$

The Handcock-Wallis parametrisation is given by

$$C(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} (2\sqrt{\nu}r)^\nu K_\nu(2\sqrt{\nu}r)$$

Usage

```
RMwhittle(nu, notinvnu, var, scale, Aniso, proj)
```

```
RMmatern(nu, notinvnu, var, scale, Aniso, proj)
```

```
RMhandcock(nu, notinvnu, var, scale, Aniso, proj)
```

Arguments

`nu` a numerical value called “smoothness parameter”; should be greater than 0.

`notinvnu` logical. If FALSE then in the definition of the models ν is replaced by $1/\nu$. This parametrization seems to be more natural. Default is, however, TRUE according with the definitions in literature.

`var, scale, Aniso, proj` optional arguments; same meaning for any [RMmodel](#). If not passed, the above covariance function remains unmodified.

Details

The three models are alternative parametrizations of the same covariance function. The Matern model or the Handcock-Wallis parametrisation should be preferred as they separate the effects of the scaling parameter and the shape parameter.

The Whittle-Matern model is the 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](#) $C(r) = -r^2$ appears as limit of the above Handcock-Wallis parametrisation.

For generalizations see section ‘See Also’.

Value

The functions return an object of class [RMmodel](#).

Note

The Whittle-Matern model is a normal scale mixture.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, <http://ms.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 $\nu \in (0, 1/2]$)

- Storkorb, 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=100)
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

<code>phi</code>	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.
<code>stationary_only</code>	optional arguments; same meaning as for RPgauss . It is ignored if the submodel is a process definition.
<code>threshold</code>	real valued. RPbernoulli returns 1 if value of the random field given by <code>phi</code> is equal to or larger than the value of <code>threshold</code> , 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 default value is 0.

Details

[RPbernoulli](#) can be 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>

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, 0.1)
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 Chi2 Random Fields

Description

RPchi2 defines a chi2 field.

Usage

```
RPchi2(phi, boxcox, 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.
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
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>

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, 0.1)
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, boxcox, stationary_only)
```

Arguments

phi	the RMmodel .
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
stationary_only	Logical or NA. Used for the automatic choice of methods. <ul style="list-style-type: none"> • 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 is 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 explicitly 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 models, 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>

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, 0.02)
plot(model)
plot(RFsimulate(model, x=x, seed=0))
plot(RFsimulate(RPgauss(model), x=x, seed=0), col=2) ## the same
```

RPpoisson

Simulation of Poisson 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>

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, 0.02)
model <- Rball()
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, 0.01)
model <- RMtruncsupport(radius=5, RMgauss())
z <- RFsimulate(RPpoisson(model), x, intensity = 100)
plot(z)
```

Description

Here, all 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>

See Also

[RC](#), [RR](#), [RM](#), [RF](#), [R](#).

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 covariance 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$variable1 >=0 ) - b@data$variable1)) == 0 ## TRUE,
## i.e. RPbernoulli is indeed a thresholded Gaussian process
```

Description

RPt defines a t field.

Usage

```
RPt(phi, boxcox, 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.
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
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>

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, 0.1)
z <- RFsimulate(model, x, x, n=4)
plot(z)
```

RRdeterm *Degenerate Distributions*

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>, <http://ms.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 *Definition of Distribution Families*

Description

RRdistr defines a distribution family given by `fct`. It is used to introduce [random parameters](#) based on distributions defined on R.

Usage

```
RRdistr(name, nrow, ncol,
        envir, ...)
```

Arguments

name	an arbitrary family of distributions. E.g. norm() for the family dnorm, pnorm, qnorm, rnorm. See examples below.
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 <code>new.env()</code> .
...	Second possibility to pass the distribution family is to pass a character string as name and to give the argument within ... See examples below.

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

See [Bayesian Modelling](#) for a less technical introduction to hierarchical modelling.

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 mathematically correct.

Further, RRdistr 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>

See Also

[RMmodel](#), [RR](#), [RFsimulate](#), [RFdistr](#).

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 <- 10

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, part 1
## note that the first 'exp' refers to the exponential function,
## the second to the exponential distribution.
(model1 <- RMgauss(var=exp(3), scale=exp(rate=1)))
x <- 1:100/10
plot(z1 <- RFsimulate(model=model, x=x))

## snd, part 2
## exactly the same result as in the previous example
(model2 <- RMgauss(var=exp(3), scale=RRdistr("exp", rate=1)))
plot(z2 <- RFsimulate(model=model, x=x))
all.equal(model1, model2)

```

RRgauss

Vector Of Independent Gaussian Random Variables

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 distribution 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>, <http://ms.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

Location and Scale Modification of A Distribution

Description

RRloc modifies location and scale of a distribution.

Usage

```
RRloc(phi, mu, scale, pow)
```

Arguments

phi	distribution <code>RMmodel</code>
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>, <http://ms.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(ballldim=2)
hist(RFr distr(model, n=1000), 50)

## empirical density of the distribution 'RRspheric', shifted by 3
model <- RRloc(mu=3, RRspheric(ballldim=2))
hist(RFr distr(model, n=1000), 50)
```

RRmcmc

Random Sample From The Modulus Of A Function

Description

RRmcmc draws a random sample from the modulus of any given function (provided the integral is finite).

Usage

```
RRmcmc(phi, mcmc_n, sigma, normed, maxdensity, rand.loc, gibbs)
```

Arguments

phi	an arbitrary integrable function
mcmc_n	positive integer. Every mcmc_nth element of the MCMC chain is returned.
sigma	positive real number. The MCMC update is done by adding a normal variable with standard deviation sigma.
normed	logical. Only used if the value of the density is calculated. If FALSE the un-normed value given by phi is returned. Default: FALSE.
maxdensity	positive real number. The given density is truncated at maxdensity. Default: 1000.
rand.loc	logical. Internal. Do not change the value. Default: FALSE.
gibbs	logical. If TRUE only one component is updated at a time. Default: FALSE.

Value

RRmcmc returns an object of class `RMmodel`.

Note

The use of RRmcmc is completely on the risk of the user. There is no way to check whether the integral of the modulus is finite.

Author(s)

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

See Also

`RMmodel`, `RR`, `RRdistr`, `RMuser`.

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

## not exponential, but the Laplace distribution as symmetry is assumed
z <- RFr distr(RRmcmc(RMexp(), sigma=1), n=10000, cores=1)
hist(z, 100, freq=FALSE)
curve(0.5 * exp(-abs(x)), add=TRUE, col="blue") ## Laplace distribution
```

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 responsibility 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 of the density function, the probability distribution and the quantile function, but not the simulation of random variables.
approx	logical. Default is 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, nor the quantile function will be available if approx=TRUE.
onesided	logical. Only 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 model defines an isotropic density function $f(x)$ with respect to the l_1 norm, i.e. $f(x) = c\phi(\|x\|_{l_1})$ with some function ϕ . Here, c is a norming constant so that the integral of f equals one.

In case that ϕ is monotonically decreasing then rejection sampling is used, else MCMC.

The function ϕ might have a polynomial pole at the origin and asymptotical 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>, <http://ms.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 straightforward way):
distr <- RRrectangular(RMgauss(), approx=FALSE)
z <- RFrndistr(distr, n=1000000)
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 <- RFrtdistr(distr, n=1000000)
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 ball of *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>, <http://ms.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(RFrtdistr(RRspheric(balldim=2), n=1000), 50)

```

Description

The model refers to the d -dimensional uniform 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 indicator 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>, <http://ms.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 [0,2] x [-2, -1]
RRfdistr(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)
```

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>

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
## seeds in the paper.

x <- seq(0, 10, length=128)

# 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)
```

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>

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=256) ## currently does not work
T <- c(0, 0.02, 1275)
col <- c(topo.colors(300)[1:100], cm.colors(300)[c((1:50) * 2,
101:150)])

y <- x <- seq(0, 10, len=5)
T <- c(0, 0.02, 4)

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)
plot(z, MARGIN.movie=3) # add 'file="ci.avi"' to get it stored
```

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>

References

- Strokorb, K., Ballani, F. and Schlather, M. (2014) Systematic co-occurrence of tail correlation functions among max-stable processes. Work in progress.

Examples

Sequential

Methods relying on square roots of the covariance matrix

Description

Sequential method relying on square roots of the covariance matrix

Usage

```
RPsequential(phi, boxcox, back_steps, initial)
```

Arguments

<code>phi</code>	object of class RMmodel ; specifies the covariance model to be simulated.
<code>boxcox</code>	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
<code>back_steps</code>	Number of previous instances on which the algorithm should condition. If less than one then the number of previous instances equals $\max / (\text{number of spatial points})$. Default: 10.
<code>initial</code>	First, $N = (\text{number of spatial points}) * \text{back_steps}$ number of points are simulated. Then, sequentially, all spatial points for the next time instance are simulated at once, based on the previous <code>back_steps</code> 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 <code>initial</code> is non-negative, then <code>initial</code> first steps are performed. If <code>initial</code> is negative, then <code>back_steps - initial</code> initial steps are performed. The latter ensures that none of the very first N variables are returned. Default: -10.

Details

`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>, <http://ms.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

[Gaussian](#), [RP](#), [RPdirect](#).

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=RPsequential(model), 0:10, 0:10, n=4)
plot(z)
```

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 ξ is always a number, i.e. ξ is constant in space. In contrast, μ and s might be constant numerical values or (in future!) be given by an [RMmodel](#), in particular by an [RMtrend](#) model.

For $\xi = 0$, the default values of μ and s are 0 and 1, respectively. For $\xi \neq 0$, the default values of μ and s are 1 and $|\xi|$, respectively, so that it defaults to the standard Frechet case if $\xi > 0$.

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 $\mathbf{R}^d \times (0, \infty)$ 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 from 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>

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 <- seq(0, 1000, 0.2)
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, 0.1)
z <- RFsimulate(RPsmith(model, xi=0), x, x)
plot(z)

## original Smith model
x <- seq(0, 10, 0.05)
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.

Author(s)

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

Source

The data were collected by Wolfgang Falk, Soil Physics Group,
University of Bayreuth, Germany.

References

Falk, W. (2000) *Kleinskalige räumliche Variabilität von Lachgas und bodenchemischen Parametern [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)
)
dta <- soil["moisture"]

## plot the data first
colour <- rainbow(100)
plot(dta, col=colour)

## fit by eye
gui.model <- RFGui(dta)

## fit by ML
model <- ~1 + RMwhittle(scale=NA, var=NA, nu=NA) + RMnugget(var=NA)
(fit <- RFfit(model, data=dta))
plot(fit, method=c("ml", "plain", "sqrt.nr", "sd.inv"),

```

```

model = gui.model, col=1:8)

## Kriging ...
x <- seq(min(dta@coords[, 1]), max(dta@coords[, 1]), l=121)
k <- RFinterpolate(fit, x=x, y=x, data=dta)
plot(x=k, col=colour)
plot(x=k, y=dta, 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@m1, x=x, y=x, data=dta, n=50)
plot(cs, col=colour)
plot(cs, y=dta, col=colour)
Print(mean(apply(as.array(cs) <= 24, 3, 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 explicit transformation is only necessary if several variables and 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>

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 <- RFvariogram(data=z))
(emp.var1 <- RFvariogram(data=z1))
(emp.var2 <- RFvariogram(data=sp2RF(z2, param=list(n=n, vdim=1))))

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, boxcox)
```

Arguments

phi object of class [RMmodel](#); specifies the covariance model to be simulated.

boxcox the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See [RFboxcox](#) for details.

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 trend model **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 field after the field has been simulated. Hence, when `var` is random, then for each realization of the Gaussian field (for `n>1` in **RFsimulate**) a new realization of `var` is used.

Further, new coordinates are created where the old coordinates have been divided 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**.

- **RMppplus**
- **RMtrend**

Note that **RPspecific** applies only to the first model or operator in the argument `phi`.

Value

RPspecific returns an object of class **RMmodel**.

Author(s)

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

Gaussian, **RP**.

Examples

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

## example for implicit 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 explicit use: every simulation has a different variance
```

```

model <- RPspecific(RMS(var=unif(min=0, max=100), RMgauss()))
x <- seq(0,50,0.02)
plot(RFsimulate(model, x=x, n=4), ylim=c(-15,15))

```

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 Bochner's theorem and the corresponding spectral measure Ξ for a given covariance function $C(h)$. For $x \in \mathbf{R}^d$, the field

$$Y(x) = \sqrt{2} \cos(\langle V, x \rangle + 2\pi U)$$

with $V \Xi$ and $U Ufo((0, 1))$ is a random field with covariance function $C(h)$. A scaled superposition of many independent realizations of Y gives a Gaussian field according 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, boxcox, sp_lines, sp_grid, prop_factor, sigma)
```

Arguments

phi	object of class <code>RMmodel</code> ; specifies the covariance model to be simulated.
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See <code>RFboxcox</code> for details.
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 <code>grid=FALSE</code> , and $k\pi/\text{sp_lines}$ for k in $1:\text{sp_lines}$, 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 sampled by MCMC. Let p be the average rejection rate. Then the chain is sampled every n th point where $n = \lceil \log(p) \rceil * \text{prop_factor}$. 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 <code>sigma</code> is not positive then <code>RandomFields</code> tries to find a good choice for <code>sigma</code> itself. Default: 0.

Value

RPspectral returns an object of class [RMmodel](#).

Author(s)

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

[Gaussian](#), [RP](#), [RPtbn](#).

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=400)
z <- RFsimulate(model, x, y, n=2)
plot(z)
```

Spherical models

Covariance models valid on a sphere

Description

This page summarizes the covariance models that can be used for spherical coordinates (and earth coordinates).

Details

The following models are available:

Completely monotone functions allowing for arbitrary scale

RMbcw	Model bridging stationary and intrinsically stationary processes for $\alpha \leq 1$ and $\beta < 0$
RMcubic	cubic model
RMdagum	Dagum model with $\beta < \gamma$ and $\gamma \leq 1$
RMexp	exponential model
RMgencauchy	generalized Cauchy family with $\alpha \leq 1$ (and arbitrary $\beta > 0$)
RMmatern	Whittle-Matern model with $\nu \leq 1/2$
RMstable	symmetric stable family or powered exponential model with $\alpha \leq 1$
RMwhittle	Whittle-Matern model, alternative parametrization with $\nu \leq 1/2$

Other isotropic models with arbitrary scale

<code>RMconstant</code>	spatially constant model
<code>RMnugget</code>	nugget effect model

Compactly supported covariance functions allowing for scales up to π (or 180 degrees)

<code>RMaskey</code>	Askey's model
<code>RMcircular</code>	circular model
<code>RMgengneiting</code>	Wendland-Gneiting model; differentiable models with compact support
<code>RMgneiting</code>	differentiable model with compact support
<code>RMspheric</code>	spherical model

Anisotropic models

None up to now.

Basic Operators

<code>RMmult</code> , *	product of covariance models
<code>RMplus</code> , +	sum of covariance models or variograms

See [RMmodels](#) for cartesian models.

Author(s)

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

See Also

[coordinate systems](#), [RMmodels](#), [RMtrafo](#).

Examples

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

RFgetModelNames(isotropy=c("spherical isotropic"))

## an example of a simple model valid on a sphere
model <- RMexp(var=1.6, scale=0.5) + RMnugget(var=0) #exponential + nugget
plot(model)

## a simple simulation
```

```

l <- seq(0, 85, 1.2)
coord <- cbind(lon=l, lat=l)

z <- RFsimulate(RMwhittle(s=30, nu=0.45), coord, grid=TRUE) # takes 1 min
plot(z)

z <- RFsimulate(RMwhittle(s=500, nu=0.5), coord, grid=TRUE,
               new_coord_sys="orthographic", zenit=c(25, 25))
plot(z)

z <- RFsimulate(RMwhittle(s=500, nu=0.5), coord, grid=TRUE,
               new_coord_sys="gnomonic", zenit=c(25, 25))
plot(z)

## space-time modelling on the sphere
sigma <- 5 * sqrt((R.lat()-30)^2 + (R.lon()-20)^2)
model <- RMprod(sigma) * RMtrafo(RMexp(s=500, proj="space"), "cartesian") *
  RMspheric(proj="time")
z <- RFsimulate(model, 0:10, 10:20, T=seq(0, 1, 0.1),
               coord_system="earth", new_coordunits="km")
plot(z, MARGIN.slices=3)

```

Square root

Methods relying on square roots of the covariance matrix

Description

Methods relying on square roots of the covariance matrix

Usage

```
RPdirect(phi, boxcox)
```

Arguments

phi	object of class RMmodel ; specifies the covariance model to be simulated.
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.

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 12000 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.

Value

RPdirect returns an object of class `RMmodel`.

Author(s)

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

[Gaussian](#), [RP](#), [RPsequential](#).

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)
```

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>

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 correspond to a tail correlation 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>

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, 0.005)
z <- RFsimulate(RPsmith(RMm2r(model), xi=0), x)
plot(z, type="p", pch=20)
```

Tail Correlation Functions

Covariance models valid for max-stable random fields

Description

This page summarizes the models that can be used for tail correlation functions.

Details

The following models are available:

Completely monotone functions allowing for arbitrary scale

RMbcw	Model bridging stationary and intrinsically stationary processes for $\alpha \leq 1$ and $\beta < 0$
RMDagum	Dagum model with $\beta < \gamma$ and $\gamma \leq 1$
RMexp	exponential model
RMgencauchy	generalized Cauchy family with $\alpha \leq 1$ (and arbitrary $\beta > 0$)
RMmatern	Whittle-Matern model with $\nu \leq 1/2$
RMstable	symmetric stable family or powered exponential model with $\alpha \leq 1$
RMwhittle	Whittle-Matern model, alternative parametrization with $\nu \leq 1/2$

Other isotropic models with arbitrary scale

`RMnugget` nugget effect model

Compactly supported covariance functions

`RMaskey` Askey's model
`RMcircular` circular model
`RMconstant` identically constant
`RMcubic` cubic model
`RMgengneiting` Wendland-Gneiting model; differentiable models with compact support
`RMgneiting` differentiable model with compact support
`RMspheric` spherical model

Anisotropic models

None up to now.

Basic Operators

`RMmult`, * product of covariance models
`RMplus`, + sum of covariance models or variograms

Operators related to process constructions

`RMbernoulli` correlation of binary fields
`RMbrownresnick` tcf of a [Brown-Resnick](#) process
`RMschlather` tcf of an extremal Gaussian process / [Schlather](#) process
`RMm2r` M2 process with monotone shape function
`RMm3b` M3 process with balls of random radius
`RMmps` M3 process with hyperplane polygons

See [RMmodels](#) for cartesian models.

Author(s)

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

References

- Strokorb, K., Ballani, F., and Schlather, M. (2015) Tail correlation functions of max-stable processes: Construction principles, recovery and diversity of some mixing max-stable processes with identical TCF. *Extremes*, **18**, 241-271

See Also

[coordinate systems](#), [RM](#), [RMmodels](#), [RMtrafo](#).

Examples

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

## an example of a simple model
model <- RMexp(var=1.6, scale=0.5) + RMnugget(var=0) #exponential + nugget
plot(model)
```

Tbm

*Turning Bands method***Description**

The Turning Bands method is a simulation method for stationary, isotropic (univariate or multivariate) 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 [RMtbn](#).

For details see [RMtbn](#).

Usage

```
RPtbn(phi, boxcox, 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 <code>RFgetModelNames(type="positive definite", domain="single variable", isotropy="isotropic", vdim=1)</code>) which is valid in dimension <code>fulldim</code> .
boxcox	the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See RFboxcox for details.
fulldim	a positive integer. The dimension of the space of the random field to be simulated.
reduceddim	a positive integer; less than <code>fulldim</code> . The dimension of the auxiliary hyperplane (most frequently a line, i.e. <code>reduceddim=1</code>) 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 linessimufactor is ignored. If both arguments are naught then points is used (and must be positive). The grid on the line is linessimufactor-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 linessimufactor. 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 parameters linessimufactor 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 [RMnssst](#), and multiplicative models 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 linessimufactor. For covariance models with larger values of the scale parameter, linessimufactor=2 is too small.

Author(s)

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

[Gaussian, 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, 0.1)
z <- RFsimulate(model, x=x, y=x, z=x, T=c(1,1,5))
plot(z, MARGIN.slices=4, MARGIN.movie=3)
```

Description

The coding of trends, in particular multivariate trends, will be described here.

Details

See [RFcalc](#), [RMtrend](#) and also the examples below for some insight on the possibilities of trend modelling.

Author(s)

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

See Also

[RFcalc](#), [RM](#), [RMmodels](#), [RMtrend](#), [RMmodelsMultivariate](#).

Examples

```
data(ca20) ## data set originally from geoR

head(ca20.df)
RFOptions(coordnames=c("east", "north"), varnames="data")

## covariance model with variance, scale and nugget to be estimated;
## just to abbreviate later on
M <- RMexp(var=NA, scale=NA) + RMnugget(var=NA)

## short definition of a trend using the fact that ca20.df is a
## data.frame
ca20.RFmod02 <- ~ 1 + altitude + M
(ca20.fit02.RF <- RFFit(ca20.RFmod02, data=ca20.df, M=M))

## long definition which also allows for more general constructions
ca20.RFmod02 <- NA + NA*RMcovariate(ca20.df$altitude) + M
(ca20.fit02.RF <- RFFit(ca20.RFmod02, data=ca20.df))

## Note that the following also works.
## Here, the covariance model must be the first summand
ca20.RFmod02 <- M + NA + ca20.df$altitude
print(ca20.fit02.RF <- RFFit(ca20.RFmod02, data=ca20.df))
```

```
### The following does NOT work, as R assumes (NA + ca20.df$altitude) + M
### In particular, the model definition gives a warning, and the
### Rffit call gives an error:
(ca20.RFmod02 <- NA + ca20.df$altitude + M)
try(ca20.fit02.RF <- Rffit(ca20.RFmod02, data=ca20.df)) ### error ...

## factors:
ca20.RFmod03 <- ~ 1 + area + M ###
(ca20.fit03.RF <- Rffit(ca20.RFmod03, data=ca20.df, M=M))
```

weather

Pressure and temperature forecast errors over the Pacific Northwest

Description

Meteorological dataset, which consists of differences between forecasts and observations (forecasts 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; Grit 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.

Author(s)

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

Source

The data were obtained from Cliff Mass and Jeff Baars from 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.
- Gritmit, 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

*Topic **classes**

- RFempVariog-class, 78
- RFfit-class, 85
- RFgridDataFrame-class, 111
- RFpointsDataFrame-class, 154
- RFsp-class, 173
- RFspatialGridDataFrame-class, 175
- RFspatialPointsDataFrame-class, 178
- RMmodel-class, 273
- RMmodelFit-class, 275
- RMmodelgenerator-class, 277

*Topic **datasets**

- ca20, 14
- soil, 362
- weather, 379

*Topic **distribution**

- Distribution Families, 27

*Topic **hplot**

- RFempVariog-class, 78
- RFfit-class, 85
- RFgridDataFrame-class, 111
- RFpointsDataFrame-class, 154
- RFspatialGridDataFrame-class, 175
- RFspatialPointsDataFrame-class, 178
- RMmodel-class, 273
- RMmodelFit-class, 275
- RMmodelgenerator-class, 277

*Topic **htest**

- RFratiotest, 160

*Topic **methods**

- Brown-Resnick-Specific, 10
- Circulant Embedding, 16
- Coins, 20
- Hyperplane, 38
- Independent Variables, 39
- plot-method, 61
- Sequential, 359

- Specific, 365
- Spectral, 367
- Square root, 370
- Tbm, 375

*Topic **models**

- Mathematical C functions, 49
- RFcov, 70
- RFmadogram, 126
- RFpar, 153
- RFpseudomadogram, 156
- RFpseudovariogram, 158
- RFvariogram, 180
- RMangle, 183
- RMaskey, 184
- RMave, 186
- RMbcw, 189
- RMbernoulli, 190
- RMbessel, 191
- RMbicauchy, 193
- RMbigneiting, 194
- RMbistable, 196
- RMbiwm, 198
- RMblend, 200
- RMbubble, 205
- RMcauchy, 208
- RMcauchytbm, 210
- RMchoquet, 211
- RMcircular, 212
- RMconstant, 213
- RMcov, 214
- RMcovariate, 216
- RMcoxisham, 217
- RMcubic, 219
- RMcurlfree, 220
- RMcutoff, 221
- RMdagum, 223
- RMdampedcos, 224
- RMdeclare, 225
- RMdelay, 227

- RMderiv, 228
- RMdewijsian, 229
- RMdivfree, 231
- RMeaxxa, 232
- RMepscauchy, 233
- RMexp, 235
- RMexponential, 236
- RMfbm, 238
- RMfixcov, 239
- RMfixed, 241
- RMflatpower, 241
- RMfractdiff, 243
- RMfractgauss, 244
- RMgauss, 245
- RMgencauchy, 246
- RMgenfbm, 248
- RMgengneiting, 249
- RMgennsst, 251
- RMgneiting, 252
- RMgneitingdiff, 253
- RMhyperbolic, 255
- RMiaco, 256
- RMid, 257
- RMidmodel, 258
- RMintexp, 260
- RMintrinsic, 261
- RMkolmogorov, 262
- RMlgd, 263
- RMlsfbm, 265
- RMma, 266
- RMmastein, 267
- RMmatrix, 269
- RMmodel, 271
- RMmppplus, 288
- RMmqam, 289
- RMmult, 291
- RMmultiquad, 292
- RMnonstwm, 295
- RMnsst, 296
- RMnugget, 297
- RMparswm, 298
- RMpenta, 300
- RMplus, 301
- RMpolynome, 303
- RMpower, 304
- RMprod, 305
- RMqam, 306
- RMqexp, 308
- RMrational, 309
- RMrotat, 310
- RMS, 311
- RMSadvanced, 312
- RMSscale, 313
- RMSchlather, 315
- RMSchur, 316
- RMsign, 317
- RMsinpower, 318
- RMSpheric, 319
- RMstable, 321
- RMstein, 322
- RMstp, 323
- RMsum, 325
- RMtbm, 326
- RMtrafo, 328
- RMtrend, 330
- RMtruncsupport, 332
- RMuser, 333
- RMvector, 335
- RMwave, 336
- RMwhittlematern, 337
- RRdeterm, 347
- RRdistr, 347
- RRgauss, 349
- RRloc, 350
- RRmcmc, 351
- RRrectangular, 352
- RRspheric, 354
- RRunif, 355
- Spherical models, 368
- Tail Correlation Functions, 373
- *Topic **optimize**
 - fitgauss, 32
 - RFfit, 82
 - RFfitoptimiser, 89
- *Topic **print**
 - RFempVariog-class, 78
 - RFfit-class, 85
 - RFgridDataFrame-class, 111
 - RFpointsDataFrame-class, 154
 - RFspatialGridDataFrame-class, 175
 - RFspatialPointsDataFrame-class, 178
 - RMmodel-class, 273
 - RMmodelFit-class, 275
- *Topic **spatial**
 - BrownResnick, 12

Constants, [21](#)
conventional2RFspDataFrame, [22](#)
Coordinate systems, [24](#)
Distribution Families, [27](#)
Extremal t, [29](#)
ExtremalGaussian, [30](#)
fitgauss, [32](#)
GaussianFields, [33](#)
GSPSJ06, [35](#)
Hierarchical Modelling, [36](#)
Internal functions, [41](#)
jss14, [43](#)
Mathematical C functions, [49](#)
Max-stable random fields, [54](#)
Max-stable random fields,
 advanced, [56](#)
Obsolete Functions Version 2, [56](#)
Obsolete Functions Version 3, [58](#)
Others, [59](#)
papers, [60](#)
PrintModelList, [67](#)
RandomFields-package, [6](#)
RFboxcox, [68](#)
RFcov, [70](#)
RFcovmatrix, [73](#)
RFCrossvalidate, [74](#)
RFdistr, [77](#)
RFfctn, [81](#)
RFfit, [82](#)
RFfitoptimiser, [89](#)
RFformula, [91](#)
RFfractaldim, [95](#)
RFfunction, [98](#)
RFgetMethodNames, [100](#)
RFgetModel, [104](#)
RFgetModelInfo, [105](#)
RFgetModelNames, [107](#)
RFgui, [113](#)
RFhurst, [114](#)
RFinterpolate, [117](#)
RFlinearpart, [121](#)
RFloglikelihood, [123](#)
RFmadogram, [126](#)
RFoldstyle, [129](#)
RFoptions, [130](#)
RFpar, [153](#)
RFpseudomadogram, [156](#)
RFpseudovariogram, [158](#)
RFratiotest, [160](#)
RFsimulate, [163](#)
RFsimulateAdvanced, [168](#)
RFvariogram, [180](#)
RMangle, [183](#)
RMaskey, [184](#)
RMave, [186](#)
RMBall, [188](#)
RMbcw, [189](#)
RMbernoulli, [190](#)
RMBessel, [191](#)
RMBicauchy, [193](#)
RMBigneiting, [194](#)
RMBistable, [196](#)
RMBiwm, [198](#)
RMblend, [200](#)
RMbr2bg, [201](#)
RMbr2eg, [202](#)
RMBrownresnick, [204](#)
RMBubble, [205](#)
RMcauchy, [208](#)
RMcauchytbm, [210](#)
RMchoquet, [211](#)
RMcircular, [212](#)
RMconstant, [213](#)
RMcov, [214](#)
RMcovariate, [216](#)
RMcOXisham, [217](#)
RMcubic, [219](#)
RMcurlfree, [220](#)
RMcutoff, [221](#)
RMDagum, [223](#)
RMDampedcos, [224](#)
RMdeclare, [225](#)
RMdelay, [227](#)
RMderiv, [228](#)
RMDewijsian, [229](#)
RMdivfree, [231](#)
RMeaxxa, [232](#)
RMepscauchy, [233](#)
RMexp, [235](#)
RMexponential, [236](#)
RMfbm, [238](#)
RMfixcov, [239](#)
RMflatpower, [241](#)
RMfractdiff, [243](#)
RMfractgauss, [244](#)
RMgauss, [245](#)

- RMgencauchy, [246](#)
- RMgenfbm, [248](#)
- RMgengneiting, [249](#)
- RMgennsst, [251](#)
- RMgneiting, [252](#)
- RMgneitingdiff, [253](#)
- RMhyperbolic, [255](#)
- RMiaco, [256](#)
- RMid, [257](#)
- RMidmodel, [258](#)
- RMintern, [259](#)
- RMintexp, [260](#)
- RMintrinsic, [261](#)
- RMkolmogorov, [262](#)
- RMLgd, [263](#)
- RMLsfbm, [265](#)
- RMma, [266](#)
- RMmastein, [267](#)
- RMmatrix, [269](#)
- RMmodel, [271](#)
- RMmodels Overview, [280](#)
- RMmodelsAdvanced, [281](#)
- RMmodelsMultivariate, [284](#)
- RMmodelsNonstationary, [286](#)
- RMmodelsSpacetime, [287](#)
- RMmppplus, [288](#)
- RMmqam, [289](#)
- RMmult, [291](#)
- RMmultiquad, [292](#)
- RMnonstwm, [295](#)
- RMnsst, [296](#)
- RMnugget, [297](#)
- RMparswm, [298](#)
- RMpenta, [300](#)
- RMplus, [301](#)
- RMpolygon, [302](#)
- RMpolynome, [303](#)
- RMpower, [304](#)
- RMprod, [305](#)
- RMqam, [306](#)
- RMqexp, [308](#)
- RMrational, [309](#)
- RMrotat, [310](#)
- RMS, [311](#)
- RMSadvanced, [312](#)
- RMSscale, [313](#)
- RMSchlather, [315](#)
- RMSchur, [316](#)
- RMsign, [317](#)
- RMsinpower, [318](#)
- RMspheric, [319](#)
- RMstable, [321](#)
- RMstein, [322](#)
- RMstp, [323](#)
- RMsum, [325](#)
- RMtbm, [326](#)
- RMtrafo, [328](#)
- RMtrend, [330](#)
- RMtruncsupport, [332](#)
- RMuser, [333](#)
- RMvector, [335](#)
- RMwave, [336](#)
- RMwhittlematern, [337](#)
- RPbernoulli, [340](#)
- RPchi2, [341](#)
- RPgauss, [342](#)
- RPpoisson, [343](#)
- RPprocess, [344](#)
- RPt, [345](#)
- RRdeterm, [347](#)
- RRdistr, [347](#)
- RRgauss, [349](#)
- RRloc, [350](#)
- RRmcmc, [351](#)
- RRrectangular, [352](#)
- RRspheric, [354](#)
- RRunif, [355](#)
- S02, [356](#)
- S10, [356](#)
- SBS14, [357](#)
- Smith, [360](#)
- sp2RF, [364](#)
- Spherical models, [368](#)
- SS12, [371](#)
- Stokorb's Functions, [372](#)
- Tail Correlation Functions, [373](#)
- Trend Modelling, [378](#)
- *, [287](#)
- *, (RMmult), [291](#)
- *, RMmodel, RMmodel-method (RMmodel-class), [273](#)
- *, RMmodel, character-method (Mathematical C functions), [49](#)
- *, RMmodel, logical-method (RMmodel-class), [273](#)
- *, RMmodel, numeric-method

- (RMmodel-class), 273
- *, character, RMmodel-method
(Mathematical C functions), 49
- *, logical, RMmodel-method
(RMmodel-class), 273
- *, numeric, RMmodel-method
(RMmodel-class), 273
- +, 91, 109, 287
- +, (RMplus), 301
- ++, (RMppplus), 288
- +, RMmodel, RMmodel-method
(RMmodel-class), 273
- +, RMmodel, character-method
(Mathematical C functions), 49
- +, RMmodel, factor-method (Mathematical C
functions), 49
- +, RMmodel, list-method (Mathematical C
functions), 49
- +, RMmodel, logical-method
(RMmodel-class), 273
- +, RMmodel, numeric-method
(RMmodel-class), 273
- +, character, RMmodel-method
(Mathematical C functions), 49
- +, data.frame, RMmodel-method
(Mathematical C functions), 49
- +, factor, RMmodel-method (Mathematical C
functions), 49
- +, logical, RMmodel-method
(RMmodel-class), 273
- +, numeric, RMmodel-method
(RMmodel-class), 273
- , (Mathematical C functions), 49
- , RMmodel, RMmodel-method
(RMmodel-class), 273
- , RMmodel, character-method
(Mathematical C functions), 49
- , RMmodel, logical-method
(RMmodel-class), 273
- , RMmodel, numeric-method
(RMmodel-class), 273
- , character, RMmodel-method
(Mathematical C functions), 49
- , logical, RMmodel-method
(RMmodel-class), 273
- , numeric, RMmodel-method
(RMmodel-class), 273
- .RF_fit (RFfit-class), 85
- .RFfit (RFfit-class), 85
- .Random.seed, 161, 172
- /(Mathematical C functions), 49
- /, RMmodel, RMmodel-method
(RMmodel-class), 273
- /, RMmodel, character-method
(Mathematical C functions), 49
- /, RMmodel, logical-method
(RMmodel-class), 273
- /, RMmodel, numeric-method
(RMmodel-class), 273
- /, character, RMmodel-method
(Mathematical C functions), 49
- /, logical, RMmodel-method
(RMmodel-class), 273
- /, numeric, RMmodel-method
(RMmodel-class), 273
- [, RFfit, ANY, ANY, ANY-method
(RFfit-class), 85
- [, RFfit, ANY, ANY-method (RFfit-class), 85
- [, RFfit-method (RFfit-class), 85
- [, RFgridDataFrame, ANY, ANY, ANY-method
(RFgridDataFrame-class), 111
- [, RFgridDataFrame, ANY, ANY-method
(RFgridDataFrame-class), 111
- [, RFgridDataFrame-method
(RFgridDataFrame-class), 111
- [, RFpointsDataFrame, ANY, ANY, ANY-method
(RFpointsDataFrame-class), 154
- [, RFpointsDataFrame, ANY, ANY-method
(RFpointsDataFrame-class), 154
- [, RFpointsDataFrame-method
(RFpointsDataFrame-class), 154
- [, RFsp, ANY, ANY, ANY-method (RFsp-class),
173
- [, RFsp, ANY, ANY-method (RFsp-class), 173
- [, RFsp-method (RFsp-class), 173
- [, RFspatialGridDataFrame, ANY, ANY, ANY-method
(RFspatialGridDataFrame-class),
175
- [, RFspatialGridDataFrame, ANY, ANY-method
(RFspatialGridDataFrame-class),
175
- [, RFspatialGridDataFrame-method
(RFspatialGridDataFrame-class),
175
- [, RFspatialPointsDataFrame, ANY, ANY, ANY-method
(RFspatialPointsDataFrame-class),

- 178
- [,RFspatialPointsDataFrame,ANY,ANY-method
(RFspatialPointsDataFrame-class),
178
- [,RFspatialPointsDataFrame-method
(RFspatialPointsDataFrame-class),
178
- [,RMmodel,ANY,ANY,ANY-method
(RMmodel-class), 273
- [,RMmodel,ANY,ANY-method (RMmodel), 271
- [,RMmodel-method (RMmodel-class), 273
- [,RMmodelFit,ANY,ANY,ANY-method
(RMmodelFit-class), 275
- [,RMmodelFit,ANY,ANY-method
(RMmodelFit-class), 275
- [,RMmodelFit-method (RMmodelFit-class),
275
- [,RMmodelgenerator,ANY,ANY,ANY-method
(RMmodelgenerator-class), 277
- [,RMmodelgenerator,ANY,ANY-method
(RMmodelgenerator-class), 277
- [,RMmodelgenerator-method
(RMmodelgenerator-class), 277
- [<-,RFgridDataFrame,ANY,ANY,ANY-method
(RFgridDataFrame-class), 111
- [<-,RFgridDataFrame-method
(RFgridDataFrame-class), 111
- [<-,RFpointsDataFrame,ANY,ANY,ANY-method
(RFpointsDataFrame-class), 154
- [<-,RFpointsDataFrame-method
(RFpointsDataFrame-class), 154
- [<-,RFsp,ANY,ANY,ANY-method
(RFsp-class), 173
- [<-,RFsp,ANY,ANY-method (RFsp-class),
173
- [<-,RFsp-method (RFsp-class), 173
- [<-,RFspatialGridDataFrame,ANY,ANY,ANY-method
(RFspatialGridDataFrame-class),
175
- [<-,RFspatialGridDataFrame-method
(RFspatialGridDataFrame-class),
175
- [<-,RFspatialPointsDataFrame,ANY,ANY,ANY-method
(RFspatialPointsDataFrame-class),
178
- [<-,RFspatialPointsDataFrame-method
(RFspatialPointsDataFrame-class),
178
- [<-,RMmodel,ANY,ANY,ANY-method
(RMmodel-class), 273
- [<-,RMmodel,ANY,ANY-method (RMmodel),
271
- [<-,RMmodel-method (RMmodel-class), 273
- [<-,RMmodelFit,ANY,ANY,ANY-method
(RMmodelFit-class), 275
- [<-,RMmodelFit,ANY,ANY-method
(RMmodelFit-class), 275
- [<-,RMmodelFit-method
(RMmodelFit-class), 275
- [<-,RMmodelgenerator,ANY,ANY,ANY-method
(RMmodelgenerator-class), 277
- [<-,RMmodelgenerator,ANY,ANY-method
(RMmodelgenerator-class), 277
- [<-,RMmodelgenerator-method
(RMmodelgenerator-class), 277
- %(Mathematical C functions), 49
- %,ANY,RMmodel-method (Mathematical C
functions), 49
- %,RMmodel,ANY-method (Mathematical C
functions), 49
- ^ (Mathematical C functions), 49
- ^,ANY,RMmodel-method (Mathematical C
functions), 49
- ^,RMmodel,ANY-method (Mathematical C
functions), 49
- ^,RMmodel,RMmodel-method
(RMmodel-class), 273
- ^,RMmodel,character-method
(Mathematical C functions), 49
- ^,RMmodel,logical-method
(RMmodel-class), 273
- ^,RMmodel,numeric-method
(RMmodel-class), 273
- ^,character,RMmodel-method
(Mathematical C functions), 49
- ^,logical,RMmodel-method
(RMmodel-class), 273
- ^,numeric,RMmodel-method
(RMmodel-class), 273
- abs (Mathematical C functions), 49
- abs,RMmodel-method (Mathematical C
functions), 49
- acosh (Mathematical C functions), 49
- acosh,RMmodel-method (Mathematical C
functions), 49
- Advanced RMmodels, 110, 361

- Advanced RMmodels (RMmodelsAdvanced),
281
- AIC, 7
- AIC, RFfit-method (RFfit-class), 85
- AIC.RF_fit (RFfit-class), 85
- AICc, 7
- AICc.RF_fit (RFfit-class), 85
- AICc.RFfit (RFfit-class), 85
- Aniso, 282, 286
- anova, 7, 140
- anova, RFfit-method (RFfit-class), 85
- anova, RMmodelFit-method
(RMmodelFit-class), 275
- anova.RF_fit (RFfit-class), 85
- anova.RM_modelFit (RMmodelFit-class),
275
- approx_step, 34
- areamat, 134, 135
- areamat=1, 135
- as.array.RFgridDataFrame
(RFgridDataFrame-class), 111
- as.array.RFpointsDataFrame
(RFpointsDataFrame-class), 154
- as.array.RFspatialGridDataFrame
(RFspatialGridDataFrame-class),
175
- as.array.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class),
178
- as.data.frame.RFgridDataFrame
(RFgridDataFrame-class), 111
- as.data.frame.RFpointsDataFrame
(RFpointsDataFrame-class), 154
- as.data.frame.RFspatialGridDataFrame
(RFspatialGridDataFrame-class),
175
- as.data.frame.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class),
178
- as.matrix.RFgridDataFrame
(RFgridDataFrame-class), 111
- as.matrix.RFpointsDataFrame
(RFpointsDataFrame-class), 154
- as.matrix.RFspatialGridDataFrame
(RFspatialGridDataFrame-class),
175
- as.matrix.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class),
178
- as.vector.RFgridDataFrame
(RFgridDataFrame-class), 111
- as.vector.RFpointsDataFrame
(RFpointsDataFrame-class), 154
- as.vector.RFspatialGridDataFrame
(RFspatialGridDataFrame-class),
175
- as.vector.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class),
178
- asin (Mathematial C functions), 49
- asin, RMmodel-method (Mathematial C
functions), 49
- asinh (Mathematial C functions), 49
- asinh, RMmodel-method (Mathematial C
functions), 49
- atan (Mathematial C functions), 49
- atan, RMmodel-method (Mathematial C
functions), 49
- atan2 (Mathematial C functions), 49
- atan2, ANY, RMmodel-method (Mathematial
C functions), 49
- atan2, RMmodel, ANY-method (Mathematial
C functions), 49
- atanh (Mathematial C functions), 49
- atanh, RMmodel-method (Mathematial C
functions), 49
- Auxiliary Models (Others), 59
- Auxiliary RMmodels, 282, 340, 341, 346, 361
- Auxiliary RMmodels (Others), 59
- AuxiliaryModels (Others), 59
- Average (Coins), 20
- back_steps, 34
- Bayesian, 48, 69, 122, 124, 280
- Bayesian (Hierarchical Modelling), 36
- bayesian (Hierarchical Modelling), 36
- Bayesian Modelling, 7, 27, 348
- Bayesian Modelling (Hierarchical
Modelling), 36
- BIC, 7
- BIC, RFfit-method (RFfit-class), 85
- BIC.RF_fit (RFfit-class), 85
- Binary field, 345
- Binary fields, 163, 169
- binary processes, 48
- BRmethods (Brown-Resnick-Specific), 10
- Brown-Resnick, 374

- Brown-Resnick (BrownResnick), [12](#)
- Brown-Resnick process (BrownResnick), [12](#)
- Brown-Resnick-Specific, [10](#)
- BrownResnick, [12](#)
- c, [52](#), [269](#)
- c (Mathematical C functions), [49](#)
- c, RMmodel-method (RMmodel-class), [273](#)
- ca20, [14](#)
- cbind.RFgridDataFrame
 - (RFgridDataFrame-class), [111](#)
- cbind.RFpointsDataFrame
 - (RFpointsDataFrame-class), [154](#)
- cbind.RFspatialGridDataFrame
 - (RFspatialGridDataFrame-class), [175](#)
- cbind.RFspatialPointsDataFrame
 - (RFspatialPointsDataFrame-class), [178](#)
- cbrt (Mathematical C functions), [49](#)
- ceiling (Mathematical C functions), [49](#)
- ceiling, RMmodel-method (Mathematical C functions), [49](#)
- Changings, [8](#), [15](#)
- changings (Changings), [15](#)
- checkExamples (Internal functions), [41](#)
- chi2 field, [345](#)
- Chi2 fields, [163](#), [168](#)
- chi2 processes, [48](#)
- Choquet's representation (RMchoquet), [211](#)
- Circulant (Circulant Embedding), [16](#)
- Circulant Embedding, [16](#)
- class, [75](#)
- coerce, RFempVariog, list-method
 - (RFempVariog-class), [78](#)
- coerce, RFfit, RFempVariog-method
 - (RFfit-class), [85](#)
- coerce, RFgridDataFrame, RFpointsDataFrame, ANY-method
 - (RFgridDataFrame-class), [111](#)
- coerce, RFgridDataFrame, RFpointsDataFrame-method
 - (RFgridDataFrame-class), [111](#)
- coerce, RFpointsDataFrame, RFgridDataFrame, ANY-method
 - (RFpointsDataFrame-class), [154](#)
- coerce, RFpointsDataFrame, RFgridDataFrame-method
 - (RFpointsDataFrame-class), [154](#)
- coerce, RFspatialGridDataFrame, data.frame-method
 - (RFspatialGridDataFrame-class), [175](#)
- coerce, RFspatialGridDataFrame, RFspatialPointsDataFrame, ANY-method
 - (RFspatialGridDataFrame-class), [175](#)
- coerce, RFspatialGridDataFrame, RFspatialPointsDataFrame-method
 - (RFspatialGridDataFrame-class), [175](#)
- coerce, RFspatialPointsDataFrame, data.frame-method
 - (RFspatialPointsDataFrame-class), [178](#)
- coerce, RFspatialPointsDataFrame, RFspatialGridDataFrame, ANY-method
 - (RFspatialPointsDataFrame-class), [178](#)
- coerce, RFspatialPointsDataFrame, RFspatialGridDataFrame-method
 - (RFspatialPointsDataFrame-class), [178](#)
- coerce, SpatialGridDataFrame, RFspatialGridDataFrame-method
 - (RFspatialGridDataFrame-class), [175](#)
- coerce, SpatialPointsDataFrame, RFspatialPointsDataFrame-method
 - (RFspatialPointsDataFrame-class), [178](#)
- Coins, [20](#), [145](#)
- composed Poisson, [345](#)
- compound Poisson processes, [48](#)
- CondSimu (Obsolete Functions Version 2), [56](#)
- Constants, [21](#)
- constants, [108](#), [110](#), [329](#)
- constants (Constants), [21](#)
- contour, [176](#)
- contour.RFempVariog (RFfit-class), [85](#)
- contour.RFfit (RFfit-class), [85](#)
- contour.RFspatialGridDataFrame
 - (plot-method), [61](#)
- conventional2RFspDataFrame, [22](#), [111](#), [155](#), [175](#), [178](#)
- coord_units (Changings), [15](#)
- coordinate system, [52](#)
- coordinate system (Coordinate systems), [24](#)
- Coordinate systems, [24](#), [118](#), [164](#)
- coordinate systems, [7](#), [130](#), [136](#), [369](#), [375](#)
- coordinate systems (Coordinate systems), [24](#)
- coordinates (Coordinate systems), [24](#)
- coregionalisation (RMmatrix), [269](#)
- coregionalization (RMmatrix), [269](#)
- cos (Mathematical C functions), [49](#)

- cos, RMmodel-method (Mathematical C functions), [49](#)
- cosh (Mathematical C functions), [49](#)
- cosh, RMmodel-method (Mathematical C functions), [49](#)
- Covariance (Obsolete Functions Version 2), [56](#)
- CovarianceFct (Obsolete Functions Version 2), [56](#)
- CovMatrix (Obsolete Functions Version 2), [56](#)
- CRS, [176](#), [178](#)
- Cutoff (Circulant Embedding), [16](#)
- data.frame, [111](#), [155](#), [176](#), [178](#)
- DeleteAllRegisters (Obsolete Functions Version 2), [56](#)
- DeleteRegister (Obsolete Functions Version 2), [56](#)
- Dependencies (Internal functions), [41](#)
- derivative (RMderiv), [228](#)
- dimensions, RFdataFrame-method (RFsp-class), [173](#)
- dimensions, RFgridDataFrame-method (RFgridDataFrame-class), [111](#)
- dimensions, RFpointsDataFrame-method (RFpointsDataFrame-class), [154](#)
- dimensions, RFsp-method (RFsp-class), [173](#)
- dimensions, RFspatialDataFrame-method (RFsp-class), [173](#)
- dimensions, RFspatialGridDataFrame-method (RFspatialGridDataFrame-class), [175](#)
- dimensions, RFspatialPointsDataFrame-method (RFspatialPointsDataFrame-class), [178](#)
- Direct (Square root), [370](#)
- dist, [170](#), [328](#), [329](#)
- Distribution Families, [27](#)
- Distributions, [48](#)
- DoSimulateRF (Obsolete Functions Version 2), [56](#)
- Earth models (Spherical models), [368](#)
- earth models (Spherical models), [368](#)
- EmpiricalVariogram (Obsolete Functions Version 2), [56](#)
- erf (Mathematical C functions), [49](#)
- erfc (Mathematical C functions), [49](#)
- error function model (RMBrownresnick), [204](#)
- exp (Mathematical C functions), [49](#)
- exp, RMmodel-method (Mathematical C functions), [49](#)
- exp2 (Mathematical C functions), [49](#)
- expm1 (Mathematical C functions), [49](#)
- expm1, RMmodel-method (Mathematical C functions), [49](#)
- extremal Gaussian (ExtremalGaussian), [30](#)
- extremal Gaussian fields, [144](#)
- extremal Gaussian process (ExtremalGaussian), [30](#)
- Extremal t, [29](#)
- extremal t (Extremal t), [29](#)
- extremal t fields, [144](#)
- extremal t process (Extremal t), [29](#)
- ExtremalGaussian, [30](#)
- FinalizeExample (Internal functions), [41](#)
- fitgauss, [32](#), [84](#), [160](#)
- fitvario (Obsolete Functions Version 2), [56](#)
- floor (Mathematical C functions), [49](#)
- floor, RMmodel-method (Mathematical C functions), [49](#)
- formula, [52](#), [70](#), [73](#), [75](#), [79](#), [81](#), [83](#), [106](#), [118](#), [122](#), [123](#), [126](#), [157](#), [159](#), [164](#), [169](#), [180](#)
- formula notation, [48](#)
- fractal.dim (Obsolete Functions Version 2), [56](#)
- function, [277](#)
- gamma (Mathematical C functions), [49](#)
- Gaussian, [19](#), [21](#), [39](#), [40](#), [343](#), [345](#), [360](#), [366](#), [368](#), [371](#), [377](#)
- Gaussian (GaussianFields), [33](#)
- Gaussian random fields, [74](#), [82](#), [84](#), [163](#)
- GaussianFields, [33](#)
- GaussRF (Obsolete Functions Version 2), [56](#)
- GetModelList (PrintModelList), [67](#)
- GetModelNames (PrintModelList), [67](#)
- GKS11, [61](#)
- GKS11 (weather), [379](#)
- gradient (RMderiv), [228](#)
- GridTopology, [23](#), [70](#), [73](#), [75](#), [81](#), [83](#), [96](#), [111](#), [115](#), [118](#), [122](#), [123](#), [127](#), [157](#), [159](#),

- [161, 164, 169, 176, 181](#)
- GridTopology2gridVectors, GridTopology-method (RFspatialGridDataFrame-class), [175](#)
- GridTopology2gridVectors, matrix-method (RFspatialGridDataFrame-class), [175](#)
- GSPSJ06, [35, 61](#)
- Hierarchical (Hierarchical Modelling), [36](#)
- Hierarchical Modelling, [36](#)
- hist.RFgridDataFrame (RFgridDataFrame-class), [111](#)
- hist.RFpointsDataFrame (RFpointsDataFrame-class), [154](#)
- hist.RFspatialGridDataFrame (RFspatialGridDataFrame-class), [175](#)
- hist.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), [178](#)
- hurst (Obsolete Functions Version 2), [56](#)
- Hyperplane, [38](#)
- Hyperplanes (Hyperplane), [38](#)
- hypot (Mathematical C functions), [49](#)
- image, [80](#)
- image, RMmodel-method (RMmodel-class), [273](#)
- Independent Variables, [39](#)
- InitGaussRF (Obsolete Functions Version 2), [56](#)
- InitMaxStableRF (Obsolete Functions Version 2), [56](#)
- InitSimulateRF (Obsolete Functions Version 2), [56](#)
- interactive, [142](#)
- Internal functions, [41](#)
- Intrinsic (Circulant Embedding), [16](#)
- Inverse multiquadric (RMmultiquad), [292](#)
- jss14, [43, 61, 379, 380](#)
- kind of isotropy, [52](#)
- Kriging (Obsolete Functions Version 2), [56](#)
- kriging (RFinterpolate), [117](#)
- lambda, [135](#)
- lgamma (Mathematical C functions), [49](#)
- lgamma, RMmodel-method (Mathematical C functions), [49](#)
- linear models, [74, 82, 84](#)
- lines, RMmodel-method (RMmodel-class), [273](#)
- lines.RMmodel (RMmodel-class), [273](#)
- list, [80](#)
- lm, [97, 116, 216](#)
- log (Mathematical C functions), [49](#)
- log, RMmodel-method (Mathematical C functions), [49](#)
- log1p (Mathematical C functions), [49](#)
- log1p, RMmodel-method (Mathematical C functions), [49](#)
- log2 (Mathematical C functions), [49](#)
- log2, RMmodel-method (Mathematical C functions), [49](#)
- logLik, [7](#)
- logLik.RF_fit (RFfit-class), [85](#)
- logLik.RFfit (RFfit-class), [85](#)
- M2 (Smith), [360](#)
- M3 (Smith), [360](#)
- maintainers.machine (Internal functions), [41](#)
- Major Revisions, [47](#)
- MajorRevisions, [8, 16](#)
- MajorRevisions (Major Revisions), [47](#)
- math.c (Mathematical C functions), [49](#)
- Mathematical C functions, [49](#)
- Mathematical functions, [280](#)
- matrix_methods, [34](#)
- max (Mathematical C functions), [49](#)
- Max-stable random fields, [54](#)
- Max-stable random fields, advanced, [56](#)
- Maxstable, [345](#)
- Maxstable (Max-stable random fields), [54](#)
- maxstable, [12, 14, 30, 31, 361](#)
- maxstable (Max-stable random fields), [54](#)
- maxstable processes, [48](#)
- maxstableAdvanced, [12, 14, 30, 31, 54, 55, 202, 203, 361](#)
- maxstableAdvanced (Max-stable random fields, advanced), [56](#)
- MaxStableRF (Obsolete Functions Version 2), [56](#)
- method specification, [170](#)

- methods for simulating Gaussian random fields, [173](#)
- min (Mathematical C functions), [49](#)
- mixed model, [259](#), [278](#)
- mixed moving maxima (Smith), [360](#)
- models valid on a sphere, [26](#)
- moving maxima (Smith), [360](#)
- multiquadric family (Rmmultiquad), [292](#)
- Multivariate and vector-valued random fields, [48](#)
- Multivariate RMmodels, [15](#), [194](#), [197](#), [199](#)
- Multivariate RMmodels (RMmodelsMultivariate), [284](#)

- new.env, [348](#)
- new_coord_units (Changings), [15](#)
- non-stationary (RMmodelsNonstationary), [286](#)
- non-stationary RMmodels (RMmodelsNonstationary), [286](#)
- Nonstationary RMmodels (RMmodelsNonstationary), [286](#)
- norm, [349](#), [350](#)
- Normal, [349](#)
- Nugget (Independent Variables), [39](#)

- Obsolete Functions Version 2, [56](#)
- Obsolete Functions Version 3, [58](#)
- optim, [75](#), [83](#), [84](#), [89](#), [138–140](#), [161](#)
- options, [48](#)
- Other models, [28](#), [35](#)
- Other models (Others), [59](#)
- Others, [59](#)

- papers, [7](#), [60](#)
- par, [154](#)
- pdf, [142](#)
- persp, [64](#), [80](#), [88](#), [176](#), [274](#)
- persp, RFempVariog-method (RFempVariog-class), [78](#)
- persp, RFfit-method (RFfit-class), [85](#)
- persp, RFspatialGridDataFrame-method (plot-method), [61](#)
- persp, RMmodel-method (RMmodel-class), [273](#)
- plot, [7](#), [8](#), [142](#)
- plot, RFempVariog, missing-method (RFempVariog-class), [78](#)
- plot, RFfit, missing-method (RFfit-class), [85](#)
- plot, RFgridDataFrame, data.frame-method (plot-method), [61](#)
- plot, RFgridDataFrame, matrix-method (plot-method), [61](#)
- plot, RFgridDataFrame, missing-method (plot-method), [61](#)
- plot, RFgridDataFrame, RFgridDataFrame-method (plot-method), [61](#)
- plot, RFgridDataFrame, RFpointsDataFrame-method (plot-method), [61](#)
- plot, RFpointsDataFrame, data.frame-method (plot-method), [61](#)
- plot, RFpointsDataFrame, matrix-method (plot-method), [61](#)
- plot, RFpointsDataFrame, missing-method (plot-method), [61](#)
- plot, RFpointsDataFrame, RFgridDataFrame-method (plot-method), [61](#)
- plot, RFpointsDataFrame, RFpointsDataFrame-method (plot-method), [61](#)
- plot, RFspatialGridDataFrame, data.frame-method (plot-method), [61](#)
- plot, RFspatialGridDataFrame, matrix-method (plot-method), [61](#)
- plot, RFspatialGridDataFrame, missing-method (plot-method), [61](#)
- plot, RFspatialGridDataFrame, RFspatialGridDataFrame-method (plot-method), [61](#)
- plot, RFspatialGridDataFrame, RFspatialPointsDataFrame-method (plot-method), [61](#)
- plot, RFspatialPointsDataFrame, data.frame-method (plot-method), [61](#)
- plot, RFspatialPointsDataFrame, matrix-method (plot-method), [61](#)
- plot, RFspatialPointsDataFrame, missing-method (plot-method), [61](#)
- plot, RFspatialPointsDataFrame, RFspatialGridDataFrame-method (plot-method), [61](#)
- plot, RFspatialPointsDataFrame, RFspatialPointsDataFrame-method (plot-method), [61](#)
- plot, RMmodel, missing-method (RMmodel-class), [273](#)
- plot-method, [61](#), [154](#)
- plotWithCircles (Internal functions), [41](#)
- points, [274](#)
- points, RMmodel-method (RMmodel-class),

- 273
- points.RMmodel (RMmodel-class), 273
- Poisson spline (Rmmultiquad), 292
- powered error function
 - (Rmbrownresnick), 204
- powered exponential, 286
- powered exponential (RMstable), 321
- print, 8
- print,RFempVariog-method
 - (RFempVariog-class), 78
- print,RFfit-method (RFfit-class), 85
- print,RMmodelFit-method
 - (RMmodelFit-class), 275
- print.crossvalidate (RFCrossvalidate), 74
- print.RF_empVariog (RFempVariog-class), 78
- print.RF_fit (RFfit-class), 85
- print.RFfit (RFfit-class), 85
- print.RFgridDataFrame
 - (RFgridDataFrame-class), 111
- print.RFpointsDataFrame
 - (RFpointsDataFrame-class), 154
- print.RFratiotest (RFratiotest), 160
- print.RFspatialGridDataFrame
 - (RFspatialGridDataFrame-class), 175
- print.RFspatialPointsDataFrame
 - (RFspatialPointsDataFrame-class), 178
- print.RM_modelFit (RMmodelFit-class), 275
- print.RMmodel (RMmodel-class), 273
- print.RMmodelFit (RMmodelFit-class), 275
- print.RMmodelgenerator
 - (RMmodelgenerator-class), 277
- print.summary.crossvalidate
 - (RFCrossvalidate), 74
- PrintModelList, 67
- processes, 259
- proj, 287
- R., 10, 22, 28, 59, 99, 272, 280, 312, 345
- R. (Mathematial C functions), 49
- R.acos (Mathematial C functions), 49
- R.acosh (Mathematial C functions), 49
- R.asin (Mathematial C functions), 49
- R.asinh (Mathematial C functions), 49
- R.atan (Mathematial C functions), 49
- R.atan2 (Mathematial C functions), 49
- R.atanh (Mathematial C functions), 49
- R.c, 282, 285
- R.c (Mathematial C functions), 49
- R.cbrt (Mathematial C functions), 49
- R.ceil (Mathematial C functions), 49
- R.const (Mathematial C functions), 49
- R.cos (Mathematial C functions), 49
- R.cosh (Mathematial C functions), 49
- R.div, 274
- R.div (Mathematial C functions), 49
- R.erf (Mathematial C functions), 49
- R.erfc (Mathematial C functions), 49
- R.exp (Mathematial C functions), 49
- R.exp2 (Mathematial C functions), 49
- R.expm1 (Mathematial C functions), 49
- R.fabs (Mathematial C functions), 49
- R.fdim (Mathematial C functions), 49
- R.floor (Mathematial C functions), 49
- R.fmax (Mathematial C functions), 49
- R.fmin (Mathematial C functions), 49
- R.fmod (Mathematial C functions), 49
- R.gamma (Mathematial C functions), 49
- R.hypot (Mathematial C functions), 49
- R.ilogb (Mathematial C functions), 49
- R.is, 146
- R.is (Mathematial C functions), 49
- R.lat (Mathematial C functions), 49
- R.lgamma (Mathematial C functions), 49
- R.log (Mathematial C functions), 49
- R.log1p (Mathematial C functions), 49
- R.log2 (Mathematial C functions), 49
- R.ln (Mathematial C functions), 49
- R.minus, 274, 330
- R.minus (Mathematial C functions), 49
- R.models, 98, 282, 285, 286
- R.models (Mathematial C functions), 49
- R.mult, 274
- R.mult (Mathematial C functions), 49
- R.nextafter (Mathematial C functions), 49
- R.nexttoward (Mathematial C functions), 49
- R.p (Mathematial C functions), 49
- R.plus (Mathematial C functions), 49
- R.pow (Mathematial C functions), 49
- R.remainder (Mathematial C functions), 49

- R.round (Mathematical C functions), 49
- R.sin (Mathematical C functions), 49
- R.sinh (Mathematical C functions), 49
- R.sqrt (Mathematical C functions), 49
- R.tan (Mathematical C functions), 49
- R.tanh (Mathematical C functions), 49
- R.trunc (Mathematical C functions), 49
- random parameters, 347
- RandomFields, 16, 76, 84, 92, 103, 110, 120, 163
- RandomFields (RandomFields-package), 6
- RandomFields-package, 6
- range.RFgridDataFrame (RFgridDataFrame-class), 111
- range.RFpointsDataFrame (RFpointsDataFrame-class), 154
- range.RFspatialGridDataFrame (RFspatialGridDataFrame-class), 175
- range.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 178
- raster, 70, 73, 75, 81, 83, 96, 115, 118, 122, 123, 127, 157, 159, 161, 164, 169, 181
- RC, 10, 28, 99, 272, 280, 345
- RC (Constants), 21
- RC_CARTESIAN_COORD, 328
- RC_CARTESIAN_COORD (Constants), 21
- RC_DOMAIN_NAMES, 110, 334
- RC_DOMAIN_NAMES (Constants), 21
- RC_DOUBLEISOTROPIC (Constants), 21
- RC_EARTH_COORDS (Constants), 21
- RC_EARTH_ISOTROPIC (Constants), 21
- RC_GNOMONIC_PROJ, 328
- RC_GNOMONIC_PROJ (Constants), 21
- RC_ISO_NAMES, 110, 334
- RC_ISO_NAMES (Constants), 21
- RC_ISONAMES, 328
- RC_ISOTROPIC, 328
- RC_ISOTROPIC (Constants), 21
- RC_LIKELIHOOD_NAMES (Constants), 21
- RC_MONOTONE_NAMES (Constants), 21
- RC_NLOPTR_NAMES (Constants), 21
- RC_OPTIMISER_NAMES (Constants), 21
- RC_ORTHOGRAPHIC_PROJ, 328
- RC_ORTHOGRAPHIC_PROJ (Constants), 21
- RC_SPACEISOTROPIC, 328
- RC_SPACEISOTROPIC (Constants), 21
- RC_SPHERICAL_COORDS (Constants), 21
- RC_TYPE_NAMES (Constants), 21
- RC_UNREDUCED (Constants), 21
- residuals, RFfit-method (RFfit-class), 85
- RF, 10, 22, 28, 48, 59, 106, 272, 278, 280, 345
- RF (RFfunction), 98
- RF__name__, 259
- RF_fit-class (RFfit-class), 85
- RFboxcox, 7, 17, 20, 38, 39, 68, 68, 69, 341, 342, 346, 359, 365, 367, 370, 375
- RFcalc, 7, 74, 82, 98, 378
- RFcalc (Mathematical C functions), 49
- RFcov, 7, 15, 58, 59, 70, 71, 74, 82, 98, 128, 158, 160, 182, 272, 278
- RFcovmatrix, 7, 58, 59, 73, 73, 81, 82, 98
- RFcrossvalidate, 7, 74, 99, 130, 131, 137
- RFdataFrame (RFsp-class), 173
- RFdataFrame-class (RFsp-class), 173
- RFddistr (RFdistr), 77
- RFdistr, 28, 77, 98, 133, 348
- RFearth2cartesian, 26, 99
- RFearth2cartesian (RMtrafo), 328
- RFearth2dist, 99
- RFearth2dist (RMtrafo), 328
- RFempiricalcovariance (Obsolete Functions Version 3), 58
- RFempiricalmadogram (Obsolete Functions Version 3), 58
- RFempiricalvariogram (Obsolete Functions Version 3), 58
- RFempVario, 273
- RFempVariog, 61, 71, 79, 86, 88, 127, 158, 160, 182
- RFempVariog-class, 78
- RFfctn, 7, 53, 74, 81, 81, 82, 98
- RFfit, 7, 32, 33, 47, 48, 58, 61, 69, 71, 74–76, 82, 82, 84, 86–89, 91, 92, 99, 124, 128, 130–133, 137, 140, 141, 150, 158, 160–163, 166, 173, 182, 185, 187, 190, 192, 196, 199, 209, 210, 212, 213, 215, 218, 219, 221, 222, 224–226, 228, 230, 232, 234, 236, 237, 239–243, 245, 246, 248–251, 253, 254, 256, 260, 262, 264, 265, 267–269, 273, 276, 289–291, 293–295, 297–300, 302, 303, 305, 307, 319, 320, 322, 323, 325, 334,

- [336, 337, 339](#)
- RFfit-class, [85](#)
- RFfitOptimiser, [84, 137, 139](#)
- RFfitOptimiser (RFfitoptimiser), [89](#)
- RFfitoptimiser, [89](#)
- RFformula, [7, 15, 52, 70, 73, 75, 79, 81, 83, 91, 106, 118, 122, 123, 126, 157, 159, 164, 165, 169, 171, 180, 241, 272, 284–286, 288, 330, 331](#)
- RFformulaAdvanced, [91, 93](#)
- RFfractalDim, [7, 58, 95, 99, 117, 338](#)
- RFfunction, [98, 99](#)
- RFfunctions (RFfunction), [98](#)
- rfGenerateConstants (Internal functions), [41](#)
- rfGenerateMaths (Internal functions), [41](#)
- rfGenerateModels (Internal functions), [41](#)
- rfGenerateTest (Internal functions), [41](#)
- RFgetMethodNames, [35, 99, 100, 100, 150, 343](#)
- RFgetModel, [99, 104, 104, 106, 107](#)
- RFgetModelInfo, [99, 104, 105, 166](#)
- RFgetModelInfo_model (RFgetModelInfo), [105](#)
- RFgetModelInfo_register (RFgetModelInfo), [105](#)
- RFgetModelNames, [21, 22, 68, 83, 99, 107, 109, 110, 132, 162, 169, 170, 186, 218, 271, 279, 281, 283, 289, 307, 343](#)
- RFgridDataFrame, [23, 65, 111, 112, 155, 165, 171, 173, 174](#)
- RFgridDataFrame (RFgridDataFrame-class), [111](#)
- RFgridDataFrame-class, [111](#)
- RFgui, [7, 48, 99, 113, 114, 130, 143, 166, 173](#)
- RFhessian (RFfit-class), [85](#)
- RFhurst, [7, 58, 98, 99, 114](#)
- RFinterpolate, [8, 58, 69, 99, 117, 131](#)
- RFlikelihood, [7, 69, 84, 98, 122, 138](#)
- RFlikelihood (RFloglikelihood), [123](#)
- RFlinearpart, [7, 98, 121, 121, 122, 124](#)
- RFloglikelihood, [123, 123, 124](#)
- RFmadogram, [15, 58, 59, 71, 126, 127, 158, 160, 182](#)
- RFmodel (RFfunction), [98](#)
- RFmodels (RFfunction), [98](#)
- RFoldstyle, [129](#)
- RFoptions (RFoptions), [130](#)
- RFoptions, [8, 11–13, 22, 26, 31, 40, 52, 53, 58, 69, 71, 73, 75–77, 80, 81, 83, 84, 86, 87, 89, 92, 99–102, 104, 105, 107, 113, 118, 119, 122, 124, 127, 130, 130, 134, 141, 146, 147, 150, 151, 157, 159, 161–166, 170–173, 181, 240, 329, 343, 353, 361, 364](#)
- RFoptionsAdvanced, [18, 150, 151](#)
- RFpar, [7, 65, 153](#)
- RFparameters (Obsolete Functions Version 2), [56](#)
- RFpdistr (RFdistr), [77](#)
- RFplotEmpVariogram, [86](#)
- RFplotEmpVariogram (RFempVariog-class), [78](#)
- RFplotModel (RMmodel-class), [273](#)
- RFplotSimulation (plot-method), [61](#)
- RFplotSimulation1D (plot-method), [61](#)
- RFpointsDataFrame, [23, 112, 155, 165, 171, 173, 174](#)
- RFpointsDataFrame (RFpointsDataFrame-class), [154](#)
- RFpointsDataFrame-class, [154](#)
- RFpseudomadogram, [59, 128, 156, 157, 158](#)
- RFpseudovariogram, [59, 74, 82, 98, 158, 159, 182](#)
- RFqdistr (RFdistr), [77](#)
- RFratiotest, [7, 48, 76, 84, 99, 130, 133, 137, 140, 160](#)
- RFrdistr (RFdistr), [77](#)
- RFsimulate, [8, 28, 58, 69, 71, 74, 82, 91, 92, 98, 100, 103–105, 107, 122, 124, 128, 132–134, 146, 149, 150, 158, 160, 163, 165, 167–171, 182, 185, 187, 188, 190–192, 196, 199, 205, 209, 210, 212, 213, 215, 216, 218, 219, 221, 222, 224–226, 228, 230, 232, 234, 236, 237, 239–243, 245, 246, 248–251, 253, 254, 256, 260, 262, 264, 265, 267–269, 278, 289–291, 293–295, 297–300, 302, 303, 305, 307, 315, 319, 320, 322, 323, 325, 327, 331, 334, 336, 337, 339, 348, 366, 373](#)
- RFsimulate.more.examples, [166, 167, 169, 173](#)
- RFsimulate.sophisticated.examples, [167,](#)

- 168, 169, 173
- RFsimulateAdvanced, 35, 70, 73, 75, 81, 83, 96, 115, 118, 122, 123, 127, 157, 159, 161, 163–168, 168, 181
- RFsp, 22, 61, 63, 70, 75, 83, 86, 112, 118, 122, 124, 127, 155–157, 159, 161, 164, 165, 170, 171, 177, 179–181, 273, 276, 364, 365
- RFsp-class, 173
- RFspatialDataFrame (RFsp-class), 173
- RFspatialDataFrame-class (RFsp-class), 173
- RFspatialGridDataFrame, 23, 64, 112, 165, 171, 173–176, 179
- RFspatialGridDataFrame (RFspatialGridDataFrame-class), 175
- RFspatialGridDataFrame-class, 175
- RFspatialPointsDataFrame, 23, 156, 165, 171, 173, 174, 176, 178, 179
- RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 178
- RFspatialPointsDataFrame-class, 178
- RFspDataFrame2conventional (RFsp-class), 173
- RFspDataFrame2conventional, RFgridDataFrame-method (RFgridDataFrame-class), 111
- RFspDataFrame2conventional, RFpointsDataFrame-method (RFpointsDataFrame-class), 154
- RFspDataFrame2conventional, RFsp-method (RFsp-class), 173
- RFspDataFrame2conventional, RFspatialGridDataFrame-method (RFspatialGridDataFrame-class), 175
- RFspDataFrame2conventional, RFspatialPointsDataFrame-method (RFspatialPointsDataFrame-class), 178
- RFspDataFrame2dataArray (RFsp-class), 173
- RFspDataFrame2dataArray, RFgridDataFrame-method (RFgridDataFrame-class), 111
- RFspDataFrame2dataArray, RFsp-method (RFsp-class), 173
- RFspDataFrame2dataArray, RFspatialGridDataFrame-method (RFspatialGridDataFrame-class), 175
- RFvariogram, 7, 15, 48, 58, 59, 69, 71, 74, 81, 82, 88, 98, 99, 102, 113, 120, 128, 158, 160, 166, 173, 180, 181, 182
- RM, 7, 8, 10, 22, 28, 48, 60, 99, 272, 284–286, 288, 345, 375, 378
- RM (RMmodels Overview), 280
- RM_COVARIATE (RMcovariate), 216
- RM_DECLARE (RMdeclare), 225
- RM_DEFAULT (Internal functions), 41
- RM_DISTR (RRdistr), 347
- RM_modelFit-class (RMmodelFit-class), 275
- RM_MULT (RMmult), 291
- RM_NUGGET (RMnugget), 297
- RM_PLUS (RMplus), 301
- RM_TREND (RMtrend), 330
- RM_USER (RMuser), 333
- RMangle, 59, 183, 183, 271, 272, 283, 329
- RMaskey, 184, 185, 196, 249, 250, 281, 369, 374
- RMave, 186, 186, 187, 287
- RMball, 28, 59, 188, 303, 354
- RMbcw, 189, 189, 248, 249, 265, 281, 368, 373
- RMbernoulli, 190, 191, 281, 340, 374
- RMbessel, 53, 191, 191, 192, 224, 225, 281, 336, 337
- RMbicauchy, 193, 193, 284
- RMbigneiting, 185, 194, 194, 195, 196, 250, 253, 254, 284
- RMbignoble, 196, 196, 197, 284, 322
- RMbiwendland (RMbigneiting), 194
- RMbiwm, 196, 198, 198, 199, 284, 299, 339
- RMblend, 200, 200, 207, 313, 314
- RMbmg, 201, 203
- RMbr2eg, 202, 202, 203
- RMbrownresnick, 204, 204, 372, 374
- RMbrbl, 200, 205, 206, 312, 314
- RMcardinalsine (RMwave), 336
- RMcauchy, 109, 194, 208, 208, 209, 210, 233, 234, 246–248, 255, 256, 271, 281
- RMcauchytbm, 209, 210, 210, 234, 248
- RMchoquet, 211, 211, 292, 293, 318, 319
- RMcircular, 109, 212, 212, 213, 281, 369, 374
- RMconstant, 149, 213, 213, 214, 281, 369, 374
- RMcov, 214, 215, 282, 285
- RMcov (RMcov), 214
- RMcovariate, 133, 216, 217, 240, 282, 334
- RMcoxisham, 217, 217, 218, 287
- RMcubic, 219, 219, 281, 368, 374

- RMcurlfree, [220](#), [220](#), [229](#), [231](#), [232](#), [263](#),
[284](#), [287](#), [336](#)
 RMcutoff, [16](#), [18](#), [221](#), [221](#), [222](#), [282](#)
 RMdagum, [223](#), [223](#), [281](#), [368](#), [373](#)
 RMdampedcos, [192](#), [224](#), [224](#), [225](#), [281](#)
 RMdeclare, [15](#), [225](#), [226](#)
 RMdelay, [109](#), [227](#), [227](#), [284](#)
 RMderiv, [221](#), [228](#), [228](#), [229](#), [232](#), [284](#)
 RMdewijsian, [189](#), [190](#), [229](#), [230](#), [281](#)
 RMdivfree, [220](#), [221](#), [229](#), [231](#), [231](#), [263](#), [284](#),
[287](#), [336](#)
 RMeaxxa, [60](#), [232](#), [233](#)
 RMepscauchy, [233](#), [233](#), [234](#)
 RMetaxxa, [60](#), [233](#)
 RMetaxxa (RMeaxxa), [232](#)
 RMexp, [109](#), [235](#), [235](#), [271](#), [321](#), [322](#), [338](#), [339](#),
[368](#), [373](#)
 RMexponential, [236](#), [236](#), [237](#), [281](#), [285](#), [290](#),
[307](#)
 RMfbm, [16](#), [238](#), [238](#), [248](#), [249](#), [265](#), [272](#), [372](#)
 RMfixcov, [47](#), [133](#), [214](#), [217](#), [239](#), [239](#), [240](#),
[282](#), [334](#)
 RMfixed, [91](#), [241](#)
 RMflatpower, [241](#), [241](#), [242](#), [249](#), [281](#)
 RMformula, [52](#), [70](#), [73](#), [75](#), [79](#), [81](#), [83](#), [106](#),
[118](#), [122](#), [123](#), [126](#), [157](#), [159](#), [164](#),
[180](#)
 RMformula (RFormula), [91](#)
 RMfractdiff, [243](#), [243](#), [281](#)
 RMfractgauss, [244](#), [244](#), [281](#)
 RMgauss, [245](#), [245](#), [252–254](#), [271](#), [290](#), [307](#),
[321](#), [322](#), [338](#), [339](#), [343](#), [350](#)
 RMgencauchy, [189](#), [190](#), [208–210](#), [222](#), [246](#),
[246](#), [247](#), [264](#), [271](#), [286](#), [368](#), [373](#)
 RMgenfbm, [189](#), [190](#), [238](#), [239](#), [241](#), [242](#), [248](#),
[248](#), [281](#)
 RMgengneiting, [185](#), [194](#), [196](#), [249](#), [249](#), [250](#),
[252–254](#), [281](#), [369](#), [374](#)
 RMgennsst, [251](#), [251](#), [287](#), [297](#)
 RMgneiting, [185](#), [196](#), [246](#), [249](#), [250](#), [252](#),
[252](#), [253](#), [254](#), [271](#), [369](#), [374](#)
 RMgneitingdiff, [253](#), [253](#), [254](#), [281](#)
 RMhandcock (RMwhittlematern), [337](#)
 RMhyperbolic, [209](#), [255](#), [255](#), [281](#), [339](#)
 RMIaco, [256](#), [256](#), [287](#)
 RMid, [60](#), [257](#), [257](#), [306](#), [329](#)
 RMidmodel, [60](#), [258](#), [258](#), [329](#)
 RMintern, [259](#)
 RMintexp, [260](#), [260](#), [282](#)
 RMintrinsic, [17](#), [18](#), [261](#), [261](#), [282](#)
 RMjbessel (RMbessel), [191](#)
 RMkbessel (RMwhittlematern), [337](#)
 RMkolmogorov, [262](#), [262](#), [284](#)
 RMLgd, [234](#), [247](#), [263](#), [263](#), [264](#), [281](#)
 RMLsfbm, [190](#), [265](#), [265](#), [281](#)
 RMM2r, [58](#), [59](#), [202](#), [203](#), [205](#), [360](#), [374](#)
 RMM2r (Strokorb's Functions), [372](#)
 RMM3b, [58](#), [59](#), [205](#), [374](#)
 RMM3b (Strokorb's Functions), [372](#)
 RMMa, [266](#), [266](#)
 RMMastein, [267](#), [267](#), [268](#), [287](#)
 RMMatern, [36](#), [198](#), [246](#), [271](#), [295](#), [337](#), [368](#),
[373](#)
 RMMatern (RMwhittlematern), [337](#)
 RMMatrix, [269](#), [269](#), [284](#), [285](#), [317](#), [332](#)
 RMMixed (RMIntern), [259](#)
 RMmodel, [11–14](#), [19–21](#), [29–31](#), [35](#), [38–40](#), [52](#),
[53](#), [55](#), [61](#), [63](#), [67–71](#), [73–77](#), [79](#),
[81–84](#), [86](#), [91](#), [92](#), [98](#), [100](#), [101](#), [103](#),
[105–107](#), [109](#), [110](#), [114](#), [117](#), [118](#),
[120–124](#), [126](#), [128](#), [132](#), [141](#),
[157–160](#), [162–166](#), [169](#), [173](#), [180](#),
[182–193](#), [195–206](#), [209–215](#),
[217–258](#), [260–269](#), [271](#), [271](#), [273](#),
[275–277](#), [279](#), [281](#), [283](#), [288–327](#),
[330–344](#), [346–348](#), [350–355](#),
[359–361](#), [365–368](#), [370–373](#), [375](#),
[376](#)
 RMmodel-class, [273](#)
 RMmodelFit, [162](#)
 RMmodelFit-class, [275](#)
 RMmodelgenerator, [28](#), [99](#), [108–110](#), [271](#),
[274](#), [275](#), [333](#), [334](#)
 RMmodelgenerator-class, [22](#), [277](#)
 RMmodels, [7](#), [49](#), [53](#), [98](#), [110](#), [280](#), [284–286](#),
[288](#), [369](#), [374](#), [375](#), [378](#)
 RMmodels (RMmodel), [271](#)
 RMmodels Overview, [280](#)
 RMmodelsAdvanced, [13](#), [37](#), [67](#), [272](#), [280](#), [281](#),
[285](#), [286](#), [288](#)
 RMmodelsAuxiliary, [272](#), [280](#), [284](#)
 RMmodelsAuxiliary (Others), [59](#)
 RMmodelsMultivariate, [280](#), [284](#), [378](#)
 RMmodelsNonstationary, [280](#), [282](#), [286](#)
 RMmodelsSpaceTime, [280](#), [282](#)
 RMmodelsSpaceTime (RMmodelsSpacetime),

- 287
- RMmodelsSpacetime, 287
- RMmodelsSphere (Spherical models), 368
- RMmodelsTailCorrelation, 282
- RMmodelsTailCorrelation (Tail Correlation Functions), 373
- RMmodelsTrend, 286
- RMmodelsTrend (Trend Modelling), 378
- RMmplusplus, 60, 288, 288, 289, 366
- RMmps, 58, 60, 205, 374
- RMmps (Strokorb's Functions), 372
- RMmqam, 285, 289, 289, 290, 307
- RMmult, 34, 147, 272, 291, 291, 302, 306, 366, 369, 374
- RMmult_inverse (RMintern), 259
- RMmultiquad, 211, 212, 292, 292, 293
- RMnatasc, 104, 282, 293, 294
- RMnonstwm, 286, 295, 339
- RMnsst, 251, 287, 296, 296, 376
- RMnugget, 39, 40, 92, 109, 149, 170, 271, 297, 297, 369, 374
- RMnull (RMintern), 259
- RMparswm, 199, 284, 298, 298, 299
- RMparswmX (RMparswm), 298
- RMpenta, 281, 300, 300
- RMplus, 34, 259, 272, 274, 289, 291, 301, 301, 325, 326, 331, 366, 369, 374
- RMpolygon, 60, 188, 302
- RMpolynome, 282, 303, 303
- RMpower, 281, 282, 304, 304
- RMpoweredexp (RMstable), 321
- RMpoweredexponential (RMstable), 321
- RMprod, 282, 286, 291, 305, 305, 312, 326
- RMptsGivenShape (RMintern), 259
- RMqam, 282, 290, 306, 306, 307
- RMqexp, 281, 308, 308
- RMr3biner (RMintern), 259
- RMrational, 60, 309, 309
- RMrotat, 60, 310, 310
- RMrotation, 60, 310
- RMrotation (RMrotat), 310
- RMS, 34, 104, 259, 271, 279, 282, 283, 286, 311, 311, 313, 366
- RMSadvanced, 200, 207, 311, 312, 312, 313, 314
- RMScale, 200, 207, 312, 313, 314
- RMSchlather, 315, 315, 374
- RMSchur, 285, 316, 316
- RMselect (RMintern), 259
- RMsetparam (RMintern), 259
- RMshape.ave (RMintern), 259
- RMshape.stp (RMintern), 259
- RMsign, 28, 60, 317, 317
- RMsinpower, 211, 212, 318, 318, 319
- RMspheric, 28, 188, 271, 303, 319, 319, 320, 369, 374
- RMSpower (RMintern), 259
- RMstable, 71, 128, 158, 160, 182, 197, 222, 235, 236, 245, 246, 271, 290, 307, 321, 321, 368, 373
- RMstandardShape (RMintern), 259
- RMstatShape (RMintern), 259
- RMstein, 282, 287, 322, 322, 323, 339
- RMstp, 187, 287, 323, 324
- RMstrokorb (Strokorb's Functions), 372
- RMstrokorbBall (Obsolete Functions Version 3), 58
- RMstrokorbMono (Obsolete Functions Version 3), 58
- RMstrokorbPoly (Obsolete Functions Version 3), 58
- RMsum, 282, 302, 306, 325, 325
- RMtbm, 210, 282, 285, 287, 326, 326, 327, 375
- RMtent (RMaskey), 184
- RMtrafo, 22, 25, 26, 60, 184, 328, 369, 375
- RMtrend, 11, 13, 29, 31, 53, 91, 165, 217, 272, 278, 282, 285, 303, 330, 330, 343, 361, 366, 378
- RMtruncsupport, 60, 145, 332, 332
- RMuser, 25, 60, 240, 282, 312, 333, 334, 352
- RMvector, 220, 221, 229, 231, 232, 263, 284, 335, 335
- RMwave, 192, 281, 336, 336, 337
- RMwendland (RMgengneiting), 249
- RMwhittle, 16, 198, 199, 222, 235, 236, 253–256, 268, 271, 295, 298, 299, 322, 368, 373
- RMwhittle (RMwhittlematern), 337
- RMwhittlematern, 337
- RO> (RMintern), 259
- RO# (RMintern), 259
- ROmissing (RMintern), 259
- round (Mathematical C functions), 49
- round, RMmodel, missing-method (Mathematical C functions), 49
- RP, 10, 19, 21, 22, 28, 35, 39, 40, 48, 55, 59,

- 99, 163, 169, 272, 340, 341, 343,
 344, 346, 360, 366, 368, 371, 377
 RP (RPprocess), 344
 RPaverage (Coins), 20
 RPbernoulli, 55, 169, 190, 191, 201–203,
 340, 340, 361
 RPbrmixed, 13, 14, 54, 135
 RPbrmixed (Brown-Resnick-Specific), 10
 RPbrorig, 13, 14, 54
 RPbrorig (Brown-Resnick-Specific), 10
 RPbrownresnick, 12, 54, 202, 203, 343
 RPbrownresnick (BrownResnick), 12
 RPbrshifted, 13, 14, 54
 RPbrshifted (Brown-Resnick-Specific), 10
 RPchi2, 341, 343
 RPcirculant, 16, 33, 34, 130, 135
 RPcirculant (Circulant Embedding), 16
 RPcoins, 21, 33, 34, 40, 131, 344
 RPcoins (Coins), 20
 RPcutoff, 33, 34
 RPcutoff (Circulant Embedding), 16
 RPdirect, 26, 33, 34, 147, 360, 371
 RPdirect (Square root), 370
 RPgauss, 12, 14, 17, 30, 31, 33, 34, 55, 98,
 141, 169, 173, 246, 340, 341, 342,
 343, 346, 350, 361
 RPhyperplane, 21, 33, 34, 40, 131
 RPhyperplane (Hyperplane), 38
 RPintrinsic, 33, 34, 261, 262
 RPintrinsic (Circulant Embedding), 16
 RPloggaussnormed
 (Brown-Resnick-Specific), 10
 RPmaxstable, 56
 RPmaxstable (Max-stable random fields),
 54
 RPmaxstableAdvanced (Max-stable random
 fields, advanced), 56
 RPmodel (RPprocess), 344
 RPmodels (RPprocess), 344
 RPmppplus (RMintern), 259
 RPmult (RMintern), 259
 RPnugget, 33, 34, 146, 297, 298
 RPnugget (Independent Variables), 39
 RPopitz, 54, 343
 RPopitz (Extremal t), 29
 RPplus (RMintern), 259
 RPpoisson, 37, 332, 343
 RPprocess, 344
 RPprocesses (RPprocess), 344
 RPS (RMintern), 259
 RPschlather, 54, 202, 203, 315, 343
 RPschlather (ExtremalGaussian), 30
 RPsequential, 33, 34, 131, 146, 359, 371
 RPsequential (Sequential), 359
 RPsmith, 54, 145, 169, 278, 288, 289, 372
 RPsmith (Smith), 360
 RPspecific, 33, 34, 366
 RPspecific (Specific), 365
 RPspectral, 21, 33, 34, 40, 131, 148, 377
 RPspectral (Spectral), 367
 RPt, 343, 345
 RPtbm, 21, 33, 34, 40, 131, 135, 148, 326, 327,
 366, 368
 RPtbm (Tbm), 375
 RPtrend (RMintern), 259
 RR, 10, 22, 37, 48, 59, 78, 99, 272, 278, 280,
 318, 345, 348, 352
 RR (Distribution Families), 27
 RRarcsqrt (RMintern), 259
 RRdeterm, 28, 347, 347
 RRdistr, 28, 37, 77, 347, 347, 348–353, 355
 RRgauss, 28, 78, 246, 343, 347, 349, 350, 351,
 353, 355
 RRloc, 28, 350, 350
 RRmcmc, 351, 352
 RRmodel, 77
 RRmodel (Distribution Families), 27
 RRmodels, 37, 279
 RRmodels (Distribution Families), 27
 RRrectangular, 130, 136, 137, 352, 353, 361
 RRsetDistr (RMintern), 259
 RRsign (RMsign), 317
 RRspheric, 28, 354, 354, 355
 RRunif, 28, 350, 355, 355

 S02, 61, 356
 S10, 44, 61, 233, 309, 310, 356
 SBS14, 61, 357
 scale, 34
 Schlather, 374
 Schoenberg's representation
 (RMchoquet), 211
 ScreenDevice (Internal functions), 41
 seq, 113
 Sequential, 359
 set.seed, 133, 161

- show, RFempVariog-method (RFempVariog-class), 78
- show, RFfit-method (RFfit-class), 85
- show, RFgridDataFrame-method (RFgridDataFrame-class), 111
- show, RFpointsDataFrame-method (RFpointsDataFrame-class), 154
- show, RFspatialGridDataFrame-method (RFspatialGridDataFrame-class), 175
- show, RFspatialPointsDataFrame-method (RFspatialPointsDataFrame-class), 178
- show, RMmodel-method (RMmodel-class), 273
- show, RMmodelFit-method (RMmodelFit-class), 275
- show, RMmodelgenerator-method (RMmodelgenerator-class), 277
- showManpages (Internal functions), 41
- sin (Mathematial C functions), 49
- sin, RMmodel-method (Mathematial C functions), 49
- sine power function (RMsinepower), 318
- sinh (Mathematial C functions), 49
- sinh, RMmodel-method (Mathematial C functions), 49
- Smith, 360
- Sobolev (RMwhittlematern), 337
- soil, 7, 114, 362
- sp2RF, 8, 174, 175, 177, 178, 180, 364, 364
- space-time (RMmodelsSpacetime), 287
- spam, 34
- Spatial, 174, 179
- SpatialGridDataFrame, 112, 175, 176
- SpatialPoints, 178, 179
- SpatialPointsDataFrame, 155, 178, 179
- spConform, 84
- spConform=FALSE, 8
- spConform=TRUE, 8
- Specific, 365
- Spectral, 367
- sphere (Spherical models), 368
- Spherical models, 280, 368
- spherical models, 7, 25, 212, 293, 319, 320
- spherical models (Spherical models), 368
- split.screen, 142
- sqrt (Mathematial C functions), 49
- sqrt, RMmodel-method (Mathematial C functions), 49
- Square root, 370
- SS12, 44, 61, 371
- StartExample (Internal functions), 41
- stationary max-stable random fields, 163, 169
- stationary Poisson fields, 163, 168
- str, 8, 274
- str.RMmodel (RMmodel-class), 273
- Stokorb's Functions, 372
- summary, 8
- summary, RFempVariog-method (RFempVariog-class), 78
- summary, RFfit-method (RFfit-class), 85
- summary, RFfit-methodt (RFfit-class), 85
- summary, RFsp-method (RFsp-class), 173
- summary, RMmodelFit-method (RMmodelFit-class), 275
- summary.crossvalidate (RFCrossvalidate), 74
- summary.RF_empVariog (RFempVariog-class), 78
- summary.RF_fit (RFfit-class), 85
- summary.RM_modelFit (RMmodelFit-class), 275
- Sweave, 132
- t field, 345
- t fields, 168
- Tail Correlation Functions, 373
- Tail correlation functions, 280
- Tail correlation functions (Tail Correlation Functions), 373
- tail correlation functions, 48
- tail correlation functions (Tail Correlation Functions), 373
- tan (Mathematial C functions), 49
- tan, RMmodel-method (Mathematial C functions), 49
- tanh (Mathematial C functions), 49
- tanh, RMmodel-method (Mathematial C functions), 49
- Tbm, 375
- tbmdim (Changings), 15
- tcf (Tail Correlation Functions), 373
- trend, 48
- trend (RMtrend), 330
- Trend Modelling, 378
- trend modelling, 272, 280

trend modelling (Trend Modelling), [378](#)
trunc (Mathematical C functions), [49](#)
trunc, RMmodel-method (Mathematical C functions), [49](#)
truncated power function (RMaskey), [184](#)

unif, [355](#)
user, [48](#)

variab_units (Changings), [15](#)
variance, RFgridDataFrame-method (RFgridDataFrame-class), [111](#)
variance, RFpointsDataFrame-method (RFpointsDataFrame-class), [154](#)
variance, RFsp-method (RFsp-class), [173](#)
variance, RFspatialGridDataFrame-method (RFspatialGridDataFrame-class), [175](#)
variance, RFspatialPointsDataFrame-method (RFspatialPointsDataFrame-class), [178](#)

Variogram (Obsolete Functions Version 2), [56](#)

weather, [7](#), [44](#), [61](#), [76](#), [84](#), [163](#), [379](#)
which.submodels, [104](#)
whittle-matern, [286](#)
whittle-matern (RMwhittlematern), [337](#)

zenit (Coordinate systems), [24](#)