Package ‘RandomFields’

March 6, 2019

Version 3.3.6

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 Martin [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 (>= 3.0), sp, RandomFieldsUtils (>= 0.5.1)

Imports graphics, methods, grDevices, stats, utils

Suggests colorspace, RColorBrewer, mvtnorm, raster, tcltk2, tcltk, tkrplot, spam, tools, geoR, 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.


License GPL (>= 3)

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-03-06 11:40:06 UTC
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFgetModelInfo</td>
</tr>
<tr>
<td>RFgetModelNames</td>
</tr>
<tr>
<td>RFgridDataFrame-class</td>
</tr>
<tr>
<td>RFgui</td>
</tr>
<tr>
<td>RFHurst</td>
</tr>
<tr>
<td>RFinterpolate</td>
</tr>
<tr>
<td>RFlinearpart</td>
</tr>
<tr>
<td>RFloglikelihood</td>
</tr>
<tr>
<td>RFmadogram</td>
</tr>
<tr>
<td>RFFoldstyle</td>
</tr>
<tr>
<td>RFoptions</td>
</tr>
<tr>
<td>RFoptionsAdvanced</td>
</tr>
<tr>
<td>RFpar</td>
</tr>
<tr>
<td>RFpointsDataFrame-class</td>
</tr>
<tr>
<td>RFPseudomadogram</td>
</tr>
<tr>
<td>RFPseudovariogram</td>
</tr>
<tr>
<td>RFRatiotest</td>
</tr>
<tr>
<td>RFSimulate</td>
</tr>
<tr>
<td>RFSimulate.more_examples</td>
</tr>
<tr>
<td>RFSimulate.sophisticated.examples</td>
</tr>
<tr>
<td>RFSimulate.advanced</td>
</tr>
<tr>
<td>RFsp-class</td>
</tr>
<tr>
<td>RFspatialDataFrame-class</td>
</tr>
<tr>
<td>RFspatialPointsDataFrame-class</td>
</tr>
<tr>
<td>RFvariogram</td>
</tr>
<tr>
<td>RMangle</td>
</tr>
<tr>
<td>RMaskey</td>
</tr>
<tr>
<td>RMave</td>
</tr>
<tr>
<td>RMBall</td>
</tr>
<tr>
<td>RMBcw</td>
</tr>
<tr>
<td>RMBernoulli</td>
</tr>
<tr>
<td>RMBessel</td>
</tr>
<tr>
<td>RMBicauchy</td>
</tr>
<tr>
<td>RMBigneiting</td>
</tr>
<tr>
<td>RMBistable</td>
</tr>
<tr>
<td>RMBiwsm</td>
</tr>
<tr>
<td>RMBblend</td>
</tr>
<tr>
<td>RMBr2bg</td>
</tr>
<tr>
<td>RMBr2eg</td>
</tr>
<tr>
<td>RMBrownresnick</td>
</tr>
<tr>
<td>RMBubble</td>
</tr>
<tr>
<td>RMCauchy</td>
</tr>
<tr>
<td>RMCauchy2bm</td>
</tr>
<tr>
<td>RMCchoquet</td>
</tr>
<tr>
<td>RMCircular</td>
</tr>
<tr>
<td>RMCconstant</td>
</tr>
<tr>
<td>RMCcov</td>
</tr>
<tr>
<td>RMCovariate</td>
</tr>
<tr>
<td>R topic</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>RMcoxisham</td>
</tr>
<tr>
<td>RMcubic</td>
</tr>
<tr>
<td>RMcurlfree</td>
</tr>
<tr>
<td>RMcutoff</td>
</tr>
<tr>
<td>RMdagum</td>
</tr>
<tr>
<td>RMdampedcos</td>
</tr>
<tr>
<td>RMdeclare</td>
</tr>
<tr>
<td>RMDelay</td>
</tr>
<tr>
<td>RMDeriv</td>
</tr>
<tr>
<td>RMdewijsian</td>
</tr>
<tr>
<td>RMDivfree</td>
</tr>
<tr>
<td>RMEaxxa</td>
</tr>
<tr>
<td>RMepscauchy</td>
</tr>
<tr>
<td>RMExp</td>
</tr>
<tr>
<td>RMExponential</td>
</tr>
<tr>
<td>RMFbm</td>
</tr>
<tr>
<td>RMfixcov</td>
</tr>
<tr>
<td>RMfixed</td>
</tr>
<tr>
<td>RMFlatpower</td>
</tr>
<tr>
<td>RMFractdiff</td>
</tr>
<tr>
<td>RMFractgauss</td>
</tr>
<tr>
<td>RMGauss</td>
</tr>
<tr>
<td>RMGencauchy</td>
</tr>
<tr>
<td>RMGenfbm</td>
</tr>
<tr>
<td>RMGengneiting</td>
</tr>
<tr>
<td>RMGennsst</td>
</tr>
<tr>
<td>RMGneiting</td>
</tr>
<tr>
<td>RMGneitingdiff</td>
</tr>
<tr>
<td>RMHyperbolic</td>
</tr>
<tr>
<td>RMIacog</td>
</tr>
<tr>
<td>RMId</td>
</tr>
<tr>
<td>RMidmodel</td>
</tr>
<tr>
<td>RMItern</td>
</tr>
<tr>
<td>RMIintexp</td>
</tr>
<tr>
<td>RMItrinsic</td>
</tr>
<tr>
<td>RMKolmogorov</td>
</tr>
<tr>
<td>RMLgd</td>
</tr>
<tr>
<td>RMlsfbm</td>
</tr>
<tr>
<td>RMma</td>
</tr>
<tr>
<td>RMMaststein</td>
</tr>
<tr>
<td>RMmatrix</td>
</tr>
<tr>
<td>RMModel</td>
</tr>
<tr>
<td>RMModel-class</td>
</tr>
<tr>
<td>RMModelFit-class</td>
</tr>
<tr>
<td>RMModelgenerator-class</td>
</tr>
<tr>
<td>RMModels Overview</td>
</tr>
<tr>
<td>RMModelsAdvanced</td>
</tr>
<tr>
<td>RMModelsMultivariate</td>
</tr>
</tbody>
</table>
### R topics documented:

- RmodelNonstationary .......................................................... 285
- RmodelsSpacetime ............................................................. 286
- Rnppplus ................................................................. 287
- Rmqam ................................................................. 288
- Rmult ................................................................. 290
- Rmultiquad ............................................................ 291
- Rnatse ................................................................. 292
- Rnonstwm ............................................................. 294
- Rnsst ................................................................. 295
- Rnugget ............................................................... 296
- Rnparswm ............................................................. 297
- Rmpenta ............................................................... 299
- Rmplus ............................................................... 300
- Rmpolygon ............................................................ 301
- Rpolynome ............................................................. 302
- Rmpower ............................................................... 303
- Rmprod ............................................................... 304
- Rmqam ............................................................... 305
- Rmqexp ............................................................... 307
- RMrational ............................................................. 308
- Rmrotat ............................................................... 309
- RMS ................................................................. 310
- RMSAdvanced .......................................................... 311
- RMSscale .............................................................. 312
- RMschlather ............................................................ 314
- RMschur .............................................................. 315
- RMsign ............................................................... 316
- RMsinepower .......................................................... 317
- RMSpheric ............................................................. 318
- RMSstable .............................................................. 320
- RMSstein .............................................................. 321
- RMSstp ............................................................... 322
- RMSum ............................................................... 324
- RMtbm ............................................................... 325
- RMrtrafo ............................................................... 327
- RMtrend ............................................................... 329
- RMrtruncsupport ........................................................ 331
- RMuser ............................................................... 332
- RMvector .............................................................. 334
- RMwave ............................................................... 335
- RMwhittlematern ........................................................ 336
- RPbernoulli ............................................................ 339
- RPchi2 ................................................................. 340
- RPhauss ............................................................... 341
- RPoisson ............................................................... 342
- RProcess ............................................................... 343
- RPt ................................................................. 344
- RRdeterm ............................................................. 346
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).

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](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; tilda 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 Strokorb (Univ. Mannheim; help pages, 2011-13)

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

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

Thanks
Patrick Brown : comments on Version 3
Paulo 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 —

• V3.0 has been financially supported by Volkswagen Stiftung within the project ‘WEX-MOP’

• 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

• V1.0 has been financially supported by the German Federal Ministry of Research and Tech-
nology (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

Note

The following packages enable further choices for the optimizer (instead of optim) in Random-
Fields: 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 Pro-
cessing Committee and IEEE Acoustics, Speech, and Signal Processing Committee (1979)
IEEE press.
Brown-Resnick-Specific

Simulation methods for Brown-Resnick processes

Description

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


• see also the corresponding vignette.

See Also

See also RF, RM, RP, RR, RC, R.

Examples

```r
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)
```
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

phi, variogram object of class Rmmodel; specifies the covariance model to be simulated.
tcf the extremal correlation function; either phi or tcf must be given.
xi, mu, s the shape parameter, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.
lambda positive constant factor in the intensity of the Poisson point process used in the M3 representation, cf. Thm. 6 and Remark 7 in Oesting et. al (2012); can be estimated by setting optim_mixed if unknown. Default value is 1.
areamat vector of values in [0, 1]. The value of the kth component represents the portion of processes whose maximum is located at a distance d with k − 1 ≤ d < k from the origin taken into account for the simulation of the shape function in the M3 representation. areamat can be used for isotropic models only; can be optimized by setting optim_mixed if unknown. Default value is 1.

meshsize, vertnumber, optim_mixed, optim_mixed_tol, variobound further arguments for simulation via the mixed moving maxima (M3) representation; see Rfoptions.
prob to do
optimize_p to do
nth to do
burn.in to do
rejection 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 \( \gamma \) 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 \texttt{RMmodel}.

Note
Advanced options for \texttt{RPbroriginal} and \texttt{RPbrshifted} are \texttt{maxpoints} and \texttt{max\_gauss}, see \texttt{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

See Also
\texttt{rpbrownresnick}, \texttt{RMmodel}, \texttt{RPgauss}, \texttt{maxstable}, \texttt{maxstableAdvanced}.

Examples

```r
# 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
```

---

\texttt{BrownResnick} \hfill \textit{Brown-Resnick process}

\textbf{Description}

\texttt{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 `xi` is always a number, i.e. $\xi$ is constant in space. In contrast, $\mu$ 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 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 $x^{-\beta}dx$, $W_i \sim W$ are iid centered Gaussian processes with stationary increments and variogram $\gamma$ 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 `ROptions`.

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 `ROptions`.

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


See Also

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

Examples

```r
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'
```

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Version 3.3</td>
</tr>
<tr>
<td>- RFempiricalvariogram, RFempiricalcovariance and RFempiricalmadogram became obsolete. Use RFvariogram, RFcov, RFmadogram instead.</td>
</tr>
<tr>
<td>- RFoptions(grDefault=FALSE) returns to the old style of graphical device handling. Otherwise there is no handling.</td>
</tr>
<tr>
<td>- C code is started to be parallelized.</td>
</tr>
<tr>
<td>- Some new Multivariate RMmodels</td>
</tr>
<tr>
<td>- 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.</td>
</tr>
<tr>
<td>• Options getting obsolete (Version 3 and older)</td>
</tr>
<tr>
<td>- oldstyle is becoming warn_oldstyle</td>
</tr>
<tr>
<td>- newstyle is becoming warn_newstyle</td>
</tr>
<tr>
<td>- newAniso is becoming warn_newAniso</td>
</tr>
<tr>
<td>- ambiguous is becoming warn_ambiguous</td>
</tr>
<tr>
<td>- normal_mode is becoming warn_normal_mode</td>
</tr>
<tr>
<td>- colour_palette is becoming warn_colour_palette</td>
</tr>
<tr>
<td>• Changings in option names</td>
</tr>
<tr>
<td>- several changes in RFoptions()$graphics in version 3.1.11</td>
</tr>
<tr>
<td>- pdfnumber became in version 3.0.42 filenumber</td>
</tr>
</tbody>
</table>

Documentation of some further changings
Circulant Embedding

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 occurring at different borders of the grid which would be close on the torus, the simulation area is multiplied by a natural number. There is also a multivariate version of circulant embedding.

Cut-off embedding is a fast simulation method for stationary, isotropic Gaussian random fields on square lattices based on the standard RPCirculant method, so that exact simulation is 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 RPCirculant method, for further variogram models, e.g. RMfbm.

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

Here, multiplicative models are not allowed (yet).

For details see RMintrinsic.

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

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 (force=TRUE) after trials number of trials. Default: FALSE.
mmin                      Scalar or vector, integer if positive. CE.mmin determines the initial size of the circulant matrix. If CE.mmin=0 the minimal starting size is determined automatically according to the dimensions of the grid. If CE.mmin>0 then the absolute starting size is given. If CE.mmin<0 then the automatically determined matrix size is multiplied by |CE.mmin|; here, CE.mmin must be smaller than -1; the value -1 takes over the minimal starting size. Note: in any cases, the initial size might be increased according to CE.useprimes. Default: 0.
strategy                  Logical. 0: If the circulant matrix has negative eigenvalues then the size in each direction is doubled; 1: The size is enhanced only in one direction, namely that one where the covariance function has the largest value at the end point of the grid — note that the default value of trials is probably too small in that case. In some cases strategy=0 works better, in other cases strategy=1. Just try. Clearly, if the field is isotropic and a square grid should be simulated, then strategy=0 is the better choice. Default: 0.
maxGB                     Maximal memory used for the circulant matrix in units of GB. If this argument is set then maxmem is set to MAXINT. Default: 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
Circulant Embedding

32 (= 4 * sizeof(double)) time as large (factor 2 is needed as complex numbers must be considered for calculating the fft of the covariance matrix; another factor 2 is needed for storing the simulated result).

The value of maxmem must be at least $2^d$ times as large as the number of points to be simulated. Here, $d$ is the space dimension. In some cases even much larger.

Note that maxmem can be used to control the automatic choice of the simulation algorithm. Namely, in case of huge circulant matrices, other simulation methods (TBM) might be faster and might be preferred by the user.

If this argument is set then maxGB is set to Inf.

Default: MAXINT.

tolIm If the modulus of the imaginary part is less than tolim then the eigenvalue is always considered as real (independently of the value of force).

Default: 1E-3.

tolRe Eigenvalues between tolRe and 0 are always considered as 0 and set 0 (independently of the value of force).

Default: -1E-7.

trials Integer. A larger circulant matrix is likely to make more eigenvalues non-negative. If at least one of the thresholds tolRe and tolim are missed then the matrix size is doubled according to strategy, and the matrix is checked again. This procedure is repeated up to trials - 1 times. If there are still negative eigenvalues, the simulation method fails if force=FALSE.

Default: 3.

useprimes Logical. If FALSE the columns of the circulant matrix have length $2^k$ for some $k$. Otherwise the algorithm tries to find a nicely factorizable number close to the size of the given matrix.

Default: TRUE.

dependent Logical. If FALSE then independent random fields are created. If TRUE then at least 4 non-overlapping rectangles are taken out of the the expanded grid defined by the circulant matrix. These simulations are dependent. See RFoptionsAdvanced for an example. See trials for some more information on the circulant matrix.

Default: FALSE.

approx_step Real value. It gives the grid size of the approximating grid in case circulant embedding is used although the points do not 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 \texttt{RMmodel}.

Author(s)

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

References

Circulant Embedding


Cutoff and Intrinsic


See Also

Gaussian, \texttt{RP}

Examples

```r
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)
```
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 $\mathbb{R}^d$ 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

```r
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.
Constants

Value

RPcoins returns an object of class \texttt{RMmodel}. 

Author(s)

Martin Schlather, \texttt{<schlather@math.uni-mannheim.de>}, \url{http://ms.math.uni-mannheim.de}

References


See Also

\texttt{Gaussian, RP, RPhyperplane, RPspectral, RPtbm}.

Examples

\begin{verbatim}
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##    RFoptions(seed=NA) to make them all random again
\end{verbatim}

Description

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

Value

\begin{verbatim}
RC_TYPE_NAMES = c("tail correlation", "positive definite", "variogram", "negative definite", "point process correlation")
RC_DOMAIN_NAMES = c("single variable", "kernel", "framework dependent", "submodel dependent", "parameter dependent")
RC_ISO_NAMES = c("isotropic", "space-isotropic", "vector-isotropic", "symmetric", "cartesian system")
RC_MONOTONE_NAMES = c("not set", "mismatch in monotonicity", "submodel dependent monotonicity")

RC_ISOTROPIC gives the numerical code for option "isotropic"
RC_DOUBLEISOTROPIC gives the numerical code for option "space-isotropic"
RC_CARTESIANCOORD gives the numerical code for option "cartesian system"
RC_GNOMONICPROJ gives the numerical code for the gnomonic projection, see also zenith in RFoptions.
\end{verbatim}
RC_ORTHOGONAL_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", "tessellation") gives the names of the ML variants: (i) internal choice according to the number of data, (ii) full likelihood, (iii) (pairwise) 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)

description = "Coercion to class 'RFsp' objects"

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)
Coordinate systems

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 GridTopology; specifies the grid vectors; either coords or gridTopology must be NULL.
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 as.array if the spatial locations are randomly scattered.
vdim_close_together: logical. Currently, only vdim_close_together=FALSE is coded. In this case the dimensions of the data follow the order “locations, multivariate, repeated”. Otherwise “multivariate, locations, repeated”.

Value

Object of class RFspatialGridDataFrame, RFspatialPointsDataFrame, RFgridDataFrame or RFpointsDataFrame.

Author(s)

Alexander Malinowski, 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)

Coordinate systems

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 zenith.
- **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 "latitude" (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 \(\pi\) 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

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

Covariance models on a sphere


Tail correlation function


See Also

`RMtrafo, RFearth2cartesian, RPdirect, models valid on a sphere, RFoptions`.

Examples

```r
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]
```
Distribution Families

### 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

- **RRdeterm**: no scattering
- **RRdistr**: families of distributions transferred from R

---

```r
(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**

- **Rrgauss**: a (multivariate) Gaussian random variable
- **RRloc**: modification of location and scale
- **RRspheric**: random scale for the *RMball* to simulate *RMspheric*, etc.
- **RRunif**: a (multivariate) uniform random variable

**Note**

The allowance of random parameters is a very recent, developing feature of RandomFields. Future changings of the behaviour are not unlikely.

**Note**

A further random element is **RMsing**, 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](http://ms.math.uni-mannheim.de)

**See Also**

**RC, RF, RM, RP, Other models, RFdistr, RMmodelgenerator, R.**

**Examples**

```r
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 **RModel**; 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 \( \alpha \)-Frechet marginal distribution, see the original literature for details.

Details

The argument \( xi \) is always a number, i.e. \( \xi \) is constant in space. In contrast, \( \mu \) and \( s \) might be constant numerical values or (in future!) be given by an **RModel**, 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 \( \xi > 0 \).

Author(s)

Martin Schlather, &lt;schlather@math.uni-mannheim.de&gt;, [http://ms.math.uni-mannheim.de](http://ms.math.uni-mannheim.de)

References

See Also

RMmodel, RPGauss, maxstable, maxstableAdvanced.

Examples

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

---

**ExtremalGaussian**

**Extremal Gaussian process**

**Description**

`RPschlather` defines an extremal Gaussian process.

**Usage**

```r
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. $\xi$ is constant in space. In contrast, $\mu$ 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)
Description

Here, some details of \texttt{RFit} 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 \( \gamma \). 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 \texttt{sub.methods} in \texttt{RFit} 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, \texttt{<schlather@math.uni-mannheim.de>}, \url{http://ms.math.uni-mannheim.de}

See Also

\texttt{RFit}, \texttt{RFit-class}.

Examples

```r
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
## See 'RFit'.
```
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
- **RPTbm**: 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>grid</th>
<th>( v )</th>
<th>( d )</th>
<th>time</th>
<th>memory</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPcirculant</td>
<td>yes</td>
<td>any</td>
<td>( \leq 13 )</td>
<td>( O(v^3 f(n, d)) )</td>
<td>( O(v^2 f(n, d)) )</td>
<td>( k \sim \text{approx}_\text{step}^{-d} )</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>any</td>
<td>( \leq 13 )</td>
<td>( O(v^3 f(k, d)) )</td>
<td>( O(v^2 f(k, d)) )</td>
<td>( k \sim (\text{latticespacing})^{-d} )</td>
</tr>
<tr>
<td>RPcutoff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>see RPcirculant above</td>
</tr>
<tr>
<td>RPcoins</td>
<td>yes</td>
<td>1</td>
<td>( \leq 4 )</td>
<td>( O(kn) )</td>
<td>( O(n) )</td>
<td>( k \sim \text{scale}^{-d} )</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>1</td>
<td>( \leq 4 )</td>
<td>( O(kn) )</td>
<td>( O(n) )</td>
<td>depends on the geometry</td>
</tr>
<tr>
<td>RPdirect</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>( O(1) \ldots O(v^2 n^2) )</td>
<td>( O(v^2 n^2) )</td>
<td>effort to investigate the covariance matrix, if not diagonal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( O(vn) )</td>
<td>( O(vn) )</td>
<td>( \text{spam} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( O(v^2 n^2) )</td>
<td>( O(v^2 n^2) )</td>
<td>covariance matrix is sparse matrix with ( z ) non-zeros</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( O(v^3 n^2) )</td>
<td>( O(v^2 n^2) )</td>
<td>arbitrary covariance matrix (preparation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( O(v^2 n^3) )</td>
<td>( O(v^2 n^2) )</td>
<td>arbitrary covariance matrix (simulation)</td>
</tr>
<tr>
<td>RPgauss</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>( O(1) \ldots O(v^3 n^3) )</td>
<td>( O(1) \ldots O(n^2) )</td>
<td>only the selection process; ( O(1) ) if first method succeeds</td>
</tr>
<tr>
<td>RPhyperplane</td>
<td>any</td>
<td>1</td>
<td>2</td>
<td>( O(n/sd) )</td>
<td>( O(n/sd) )</td>
<td>( s = \text{scale} )</td>
</tr>
</tbody>
</table>
Computing demand for interpolation

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

<table>
<thead>
<tr>
<th>grid</th>
<th>( v )</th>
<th>( d )</th>
<th>time</th>
<th>memory</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>any</td>
<td>any</td>
<td>( O(1) )</td>
<td>( O(v^2n^2) )</td>
<td>effort to investigate the covariance matrix, if ( \text{matrix_methods} ) is not specified (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( O(v^2nk) )</td>
<td>( O(v(n + k)) )</td>
<td>covariance matrix is diagonal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>see ( \text{spam} + O(v^2nk) )</td>
<td>( O(z + v(n + k)) )</td>
<td>covariance matrix is sparse matrix with ( z ) non-zeros</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( O(v^3n^3 + v^2nk) )</td>
<td>( O(v^3n^2 + v^2k) )</td>
<td>arbitrary covariance matrix</td>
</tr>
</tbody>
</table>

Computing demand for conditional simulation

Assume \( v \)-variate data are given at \( n \) locations \( x_1, \ldots, x_n \) in \( d \) dimensions. To conditionally simulate at \( k \) locations \( y_1, \ldots, y_k \), 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 \( y_1, \ldots, y_k \) are defined by a grid and the \( n \) locations \( x_1, \ldots, x_n \) are a subset of \( y_1, \ldots, y_k \), a situation typical in image analysis.)

Author(s)

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

References


See Also

RP, Other models, RMmodel, RFgetMethodNames, RFsimulateAdvanced.

Examples

```r
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)
```

---

**GSPSJ06**

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

**Description**

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

**Author(s)**

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

**References**


**Examples**

```r
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)
stablestest <- 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))
}
```
Hierarchical Modelling

Bayesian Spatial Modelling

Description

**RandomFields** provides Bayesian modelling to some extend: (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 \( \nu \) 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. \( \text{norm}() \). This function call may contain arguments that must be named, e.g. \( \text{norm}(mean=3, sd=5) \).

Usage:

- \( \exp() \) denotes the exponential distribution family with rate 1.
- \( \exp(3) \) is just the scalar \( e^3 \) and
- \( \exp(\text{rate}=3) \) is the exponential distribution family with rate 3.

The family can be passed in three ways:

- implicitly, e.g. **RMwhittle**\((\nu=\exp())\) or

```r
RFcov(dist=0, model=model, dim=2, seed=0)
```

```
100x712
Hierarchical Modelling

Bayesian Spatial Modelling

Description

**RandomFields** provides Bayesian modelling to some extend: (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 \( \nu \) 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. \( \text{norm}() \). This function call may contain arguments that must be named, e.g. \( \text{norm}(mean=3, sd=5) \).

Usage:

- \( \exp() \) denotes the exponential distribution family with rate 1.
- \( \exp(3) \) is just the scalar \( e^3 \) and
- \( \exp(\text{rate}=3) \) is the exponential distribution family with rate 3.

The family can be passed in three ways:

- implicitly, e.g. **RMwhittle**\((\nu=\exp())\) or

```r
RFcov(dist=0, model=model, dim=2, seed=0)
```

```
100x712
Hierarchical Modelling

Bayesian Spatial Modelling

Description

**RandomFields** provides Bayesian modelling to some extend: (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 \( \nu \) 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. \( \text{norm}() \). This function call may contain arguments that must be named, e.g. \( \text{norm}(mean=3, sd=5) \).

Usage:

- \( \exp() \) denotes the exponential distribution family with rate 1.
- \( \exp(3) \) is just the scalar \( e^3 \) and
- \( \exp(\text{rate}=3) \) is the exponential distribution family with rate 3.

The family can be passed in three ways:

- implicitly, e.g. **RMwhittle**\((\nu=\exp())\) or

```r
RFcov(dist=0, model=model, dim=2, seed=0)
```
Hierarchical Modelling

- explicitly through \texttt{RRdistr}, e.g. \texttt{RMwhittle(nu=RRdistr(exp()))}.
- by use of \texttt{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 \texttt{RPoisson}, 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, \texttt{<schlather@math.uni-mannheim.de>}, \url{http://ms.math.uni-mannheim.de}

See Also

\texttt{RMmodelsAdvanced}. For hierarchical modelling see \texttt{RR}.

Examples

\begin{verbatim}
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)
\end{verbatim}
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

\texttt{RPhyperplane(phi, boxcox, superpos, maxlines, mar\_distr, mar\_param, additive)}

Arguments

\textbf{phi} \hspace{1cm} object of class \texttt{RMmodel}; specifies the covariance function to be simulated; only exponential covariance functions and scale mixtures of it are allowed.

\textbf{boxcox} \hspace{1cm} the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See \texttt{RFboxcox} for details.

\textbf{superpos} \hspace{1cm} integer. number of superposed hyperplane tessellations. Default: 300.

\textbf{maxlines} \hspace{1cm} integer. Maximum number of allowed lines. Default: 1000.

\textbf{mar\_distr} \hspace{1cm} integer. code for the marginal distribution used in the simulation:

- 0 uniform distribution
- 1 Frechet distribution with form parameter \texttt{mar\_param}
- 2 Bernoulli distribution (Binomial with \(n = 1\)) with argument \texttt{mar\_param}

This argument should not be changed yet.

Default: 0.

\textbf{mar\_param} \hspace{1cm} Argument used for the marginal distribution. The argument should not be changed yet.

Default: NA.

\textbf{additive} \hspace{1cm} logical. If TRUE independent realizations are added, else the maximum is taken.

Default: TRUE.

Value

\texttt{RPhyperplane} returns an object of class \texttt{RMmodel}.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, \texttt{http://ms.math.uni-mannheim.de}
References


See Also

Gaussian, RP.

Examples

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

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

**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`. However, if the anisotropy matrix does not have full rank and `nugget.tol=0`, then the simulations are likely to be odd. The value of `nugget.tol` should be of order $10^{-10}$.

**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 measurements 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


See Also

`Gaussian, RP, RPcoins, RPhyperplane, RPspectral, RPtbm`.

Examples

```r
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**

**Description**

These functions are internal and should not be used.

**Usage**

```r
rfGenerateModels(package = "RandomFields", assigning,
RFpath="/svn/RandomFields/RandomFields",
RMmodels.file=paste(RFpath, "R/RMmodels.R", sep="/"),
PL = RFoptions()$basic$printlevel)
```
Internal functions

rfGenerateConstants(package="RandomFields", aux.package = "RandomFieldsUtils",
RFpath = paste0("~/svn",package,"/",package),
RCall.to.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, R.models.file, PL
    internal
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
See Also
weather, SS12, S10.

Examples

```r
## Not run:

library(rf)

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 <- RMatrix(M = M1, RMwhittle(nu = 0.3)) +
          RMatrix(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(4, 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 <- RMatrixGencauchy(alpha = 1.5, beta = 3)
simu <- RFsimulate(model, x, y, z = c(0, 1))
plot(simu, MARGIN.slices = 3, n.slices = 2)
```
```r
## 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 = pi, 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 forth component to the
## field of sinks and sources
model <- RMcurlfree(RMmatern(nu = 5), scale = 4)
simu <- RFsimulate(model, x, y)
plot(simu, select:variables = list(1, 2 : 3, 4))
plot(model, dim = 2, xlim = c(-3, 3), main = "", cex = 2.3, col="brown")

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

```
# Section 5: Data Analysis

## get the data
data("weather")
PT <- weather[, 1 : 2]  # full data set takes more than
# half an hour to be analysed
# transformation of earth coordinates to Euclidean coordinates
Dist.mat <- as.vector(RFeart2dist(weather[, 3 : 4]))
```
```r
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]))))
```
```r
x <- RFearth2cartesian(weather[All, 3:4])
plane <- t(x) %*% x[ , 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)
```

---

Major Revisions

---

**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](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

Mathematial 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, k, 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)
Mathematical C functions

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'
Mathematical C functions

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

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

Details

R.plus adds two values
R.minus substracts two values
R.mult multiplies two values
R.div devides two values
R.const defines a constant
R.c builds a vector
**Mathematical C functions**

- **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. please use the functions given in `RMmodels` for definite functions (for `cos` see `Rmbessel`)
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](http://ms.math.uni-mannheim.de)

**See Also**

`RMmodel`, `RFfctn`, `RMtrend`

**Examples**

```r
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))
```
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

## Calculation performed on a field

```r
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

```r
print(RFFctn(R.p(new="iso"), 1:10, 1:10))
```

## Simulation of a non-stationary field where

anisotropy by a transform the coordinates $(x_{1}^2, x_{2}^1.5)$

```r
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:

```r
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

Author(s)

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

References


See Also

RP, RMmodel, RPgauass, RPherduilli, 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
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


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 = NULL, 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)
Arguments

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

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
Strokorb’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 $H_n$
operator to define mixed moving maxima (M3) processes
spectral functions belonging to a tail correlation function of the Gneiting class \( H_n \)
Indicator of a typical Poisson polygon
shape function used in the Bernoulli paper given in the references
shape function used in the Bernoulli paper given in the references
random sign
truncates the support of a shape in a Poisson based model

Special transformations

\( \text{RMeaxxa} \)  
shape function used in the Bernoulli paper given in the references
\( \text{RMetaxxa} \)  
shape function used in the Bernoulli paper given in the references
\( \text{RMidmodel} \)  
model identity
\( \text{RMid} \)  
identity but interpretation turns from a coordinate to a model value
\( \text{RMtrafo} \)  
allows to model the identity within the set of coordinates
\( \text{RMrotation} \)  
shape function used in the Bernoulli paper given in the references

Other models

\( \text{RMuser} \)  
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

\begin{verbatim}
RFoptions(seed=0) ## ANY simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
RFgetModelNames()
\end{verbatim}

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

  See GKS11 for the code.
  See GSPSJ06 for the code.
  See SS12 for the code.
  See S02 for the code.
  See S10 for the code.
  See `multivariate_jss` for the vignette.
  See SBS14 for the code.

See Also

weather, GSPSJ06, SS12, S02, S10, jss14.

Examples

```R
# For examples, see the specific papers.
```

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

```r
defplotSimulation(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")
```

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

```r
## 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, ...)
```

```r
## 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, ...)
```

```r
## 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, ...)
```

```r
## 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, ...)
• $l = 1$
  the component is shown in the usual way
• $l = 2$
  the two components are interpreted as vector and arrows are plotted
• $l = 3$
  the first component is shown as single component; the remaining two component are interpreted as a vector and plotted into the picture of the first component

```
legend        logical, whether a legend should be plotted
zlim          vector of length 2 with the usual meaning. In case of multivariate random fields, zlim can also be a character 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 $\geq 2$. If space-time-dimension $\geq 3$, 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 = "RFspatialDataFrame", 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")
    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))
plot-method

```r
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=10, 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)
```
\begin{verbatim}
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")
\end{verbatim}

\section*{PrintModelList \quad Information about the implemented covariance models}

\section*{Description}
\texttt{PrintModelList} prints the list of currently implemented models including the corresponding simulation methods.

\section*{Usage}
\texttt{PrintModelList(operators=FALSE, internal=FALSE, newstyle=TRUE)}

\section*{Arguments}
\begin{description}
  \item[operators] logical. Flag whether operators should also be considered.
  \item[internal] logical. Flag whether internal models should also be considered. In case of \texttt{PrintModelList} and \texttt{internal=2}, variants of internal models are also printed.
  \item[newstyle] logical. If \texttt{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 \texttt{RMmodelsAdvanced}. If \texttt{TRUE} then the standard names will also be shown.
\end{description}

\section*{Details}
See \texttt{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

data matrix or list of matrices.

boxcox the one or two parameters \((\lambda, \mu)\) of the box cox transformation, in the univariate case; if \(\mu\) is not given, then \(\mu\) is set to 0. If not given, the globally defined parameters are used, see Details. In the \(m\)-variate case boxcox should be a \(2 \times m\) matrix. If \(\lambda = \infty\) then no transformation is performed.

vdim the multivariate dimensionality of the field;

inverse logical. Whether the inverse transformation should be performed.

ignore.na logical. If FALSE an error message is returned if any value of boxcox 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


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")],
  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**

```r
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 `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`: 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 `seq(from = 0, to = deltaT[1], by = deltaT[2])`
- `distances, dim`: another alternative for the argument `x` to pass the (relative) coordinates, see RFsimulateAdvanced.
the number of variables of a multivariate data set. If not given and data is
an \texttt{RFsp} object created by \texttt{RandomFields}, the information there is taken from
there. Otherwise \texttt{vdim} is assumed to be one. NOTE: still the argument \texttt{vdim} is
an experimental stage.

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

Details

\texttt{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
\texttt{RFoptions}).

It is also possible to use \texttt{RFcov} to calculate the pseudocovariance function (see \texttt{RFoptions}).

Value

\texttt{RFcov} returns objects of class \texttt{RFempVariog}.

Author(s)

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

References

Boca Raton: Chapman & Hall/CRL.


See Also

\texttt{RFvariogram}, \texttt{RFmadogram}, \texttt{RMstable}, \texttt{RMmodel}, \texttt{RFsimulate}, \texttt{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, cdig=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

Covariance matrix

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 RMmodel, RFormula 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 RFormula.
  
  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.
  
  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 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
Rfcrossvalidate

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  

**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

```
Rfcrossvalidate(model, x=NULL, 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 T=seq(from=From, by=By, len=Len), one may also write T=seq(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  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. transform=list() 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.
krije.var the cross-validation prediction variance.
error the differences data - predicted value.
std.error the errors divided by the square root of the prediction variances.
p In contrast to geoR the p-value is returned, i.e. the probability that a difference with absolute value larger than the absolute value of the actual difference is observed.
A method for summary returns summary statistics for the errors and standard errors similar to geoR.
If cross_refit = TRUE and detailed_output = TRUE the returned object also contains a fitted which is a list of fitted models.

Methods

print prints the summary
summary gives a summary

Note

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

Note

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

Author(s)

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

References

• Ribeiro, P.J., Jr. and Diggle, P.J (2014) R package geoR.

See Also

RFratietest Rfit 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 *RFDistr* distribution families can be passed to *RandomFields* to create distributions available in the *RMmodel* definitions.

**Usage**

```r
RFddistr(model, x, params, dim=1, ...)
RFpdfistr(model, q, params, dim=1, ...)
RFqdfistr(model, p, params, dim=1, ...)
RFnistr(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.
**RGeomVariog-class**

**Value**

as described in the arguments

**Author(s)**

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

**See Also**

[Rgeom, Rg](#)

**Examples**

```r
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())
ν <- 0.5
Print(RFdistr(model=model, x=ν), dnorm(x=ν))
Print(RFdistr(model=model, q=ν), pnorm(q=ν))
Print(RFdistr(model=model, p=ν), qnorm(p=ν))

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)))
}
```

**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. This 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 class(x)="RFempVariog": the number of bins of angle phi that are to be plotted.

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

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

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

plot.sd logical; only for class(x)="RFempVariog"; indicates whether the calculated standard deviation (x@sd) is to be plotted (in form of arrows of length +-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 class(x)="RFempVariog" and the anisotropic case where model is given. As the empirical variogram is calculated on a sector of angles, no exact variogram curve corresponds to the mean values in this sector. If boundaries=TRUE the values of the variogram on the sector boundaries are plotted. If FALSE some kind of mean model values are plotted. Neither the boundaries may contain the values of 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](http://ms.math.uni-mannheim.de)
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 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.
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.

See Also

RFvariogram, plot-method

Examples

# see 'RFvariogram'
**Value**

`RFfitn` returns a vector.

**Author(s)**

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

**See Also**

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

**Examples**

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

model <- RMexp() - 1
RFfitn(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**

```r
RFfit(model, x = NULL, 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**: object of class `RFmodel`, `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 `RFformula`.
  All parameters that are set to NA will be estimated; see the examples below.
  Type `RFgetModelNames(type="variogram")` to get all options for `model`.

- **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**: 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`.

- **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

See Also
RFfitOptimiser, RFlikelihood, RFratios, 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")
### RFfit-class

#### Description

Class for RandomFields’ representation of model estimation results

#### Usage

```r
## 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'

```r
contour(x, ...)
```

### S3 method for class 'RFempVariog'

```r
contour(x, ...)
```

RFhessian(model)

## Arguments

- `object` see the generic function;
- `...` • `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 class(x)="Rffit"; a vector of slot names for which the fitted covariance or variogram model is to be plotted; should be a subset of `slotNames(x)` for which the corresponding slots are of class `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` class(x)="RF_fit" or class(x)="Rffit", 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 fist gives the position where the parameters are set to zero. The second gives the position where the parameters are set to one.
internal1: RMmodelFit; analog to slot 'autostart'
internal2: RMmodelFit; analog to slot 'autostart'
internal3: RMmodelFit; analog to slot 'autostart'
lowerbounds: RMmodel; covariance model in which each parameter value gives the lower bound for the respective parameter
ml: RMmodelFit; analog to slot 'autostart'
modelinfo: table with information on the parameters: name, boundaries, type of parameter
n.covariates: number of covariates
n.param: number of parameters (given by the user)
n.variab: number of variables (used internally); n.variab is always less than or equal to n.param
number.of.data: the number of data values passed to RFFit that are not NA or NaN
number.of.parameters: total number of parameters of the model that had to be estimated including variances, scales, co-variables, etc.
p.proj: vector of integers. The original position of those parameters that are used in the submodel
plain: RMmodelFit; analog to slot 'autostart'
report: If not empty, it indicates that this model should be reported and gives a standard name of the model.
Various functions, e.g. print.RMmodelFit, use this information if their argument full equals TRUE.
sel: RMmodelFit; analog to slot 'autostart'
sd.inv: RMmodelFit; analog to slot 'autostart'
sqrt.nr: RMmodelFit; analog to slot 'autostart'
submodels: list. Sequence (in some cases even nested sequence) of models that is used to determine an initial value in
table: matrix; summary of estimation results of different methods
transform: function;
true.tsdim: time space dimension of the (original!) data, even for submodels that consider parts of separable models.
true.vdim: multivariability of the (original!) data, even for submodels that consider independent models for the multivariate components.
upperbounds: RMmodel; see slot 'lowerbounds'
users.guess: RMmodelFit; analog to slot 'autostart'
ml: RMmodelFit; analog to slot 'autostart'; with maximum likelihood method
v.proj: vector of integers. The components selected in one of the submodels
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
- **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:


**See Also**

Rffit, RFvariogram, RMmodel-class, RMmodelfit-class, plot-method.

**Examples**

```r
# see Rffit
```
RFfitoptimiser

Optimisers for fitting model parameters to spatial data

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_LN_CRS2_LM", "NLOPT_LN_COBYLA", "NLOPT_LN_NELDERMEAD", "NLOPT_LN_SBPLX", "NLOPT_LN_BOBYQA", "NLOPT_LN_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")],
  RFparams=list(vdim=6, n=1))
data <- soil["moisture"]
\dontshow{if (RFoptions$internal$examples_red) {


RFfitoptimiser

```
warning("data have been reduced ")
All <- 1:7
rm(soil)
data(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 + 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)
```

RFformula

Description

It is described how to create a formula, which, for example, can be used as an argument of `RFFsimulate` and `RFFfit` 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 `beta` 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 `beta` is estimated provided that the formula is used in `RFFfit`).
• 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. ~ 1 + ...) 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


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")

RMmodel, RFsimulate, RFfit, RandomFields.
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

---

**Advanced RFformula**

**Description**

Here examples for much more advanced formula are given
Note

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

Examples

```r
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#
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#
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))
```
#### Example 3: formulae for fitting (i.e. including NAs) ####

```r
## first generate some data
model <- ~ RMexp(var=var, sc=sc) + RMnugget(var=nugg)
p <- list(var=V, nugg= ~ var * abs(cos(sc)), sc=s5)
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 ####

```r
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.
**RFractalDim**

**Usage**

```r
RFractalDim(x, y = NULL, z = NULL, data, grid,
    bin=NULL,
    vario.n=5,
    sort=TRUE,
    fft.m = c(65, 86), # in % of range of l.lambda
    fft.max.length=Inf,
    fft.max.regr=150000,
    fft.shift = 50, # in %; 50:WOSA; 100: no overlapping
    method=c("variogram", "fft"),
    mode = 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 fft.max.length. For each piece the FFT is calculated and then the average for all pieces is taken. The pieces may overlap, see the argument fft.shift.
- **fft.max.regr** If the fft.m is too large, parts of the regression fit will take a very long time. Therefore, the regression fit is calculated only if the number points given by fft.m is less than fft.max.regr.
- **fft.shift** This argument is given in percent [of fft.max.length] and defines the overlap of the pieces defined by fft.max.length. If fft.shift=50 the WOSA estimator is given; if fft.shift=100 no overlap exists.
- **method** list of implemented methods to calculate the fractal dimension; see Details.
- **mode** character. A vector with components 'nographics', 'plot' or 'interactive':...
RFfractaldim

'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.

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

Details

The function calculates the fractal dimension by various methods:

- variogram method
- Fourier transform

Value

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

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

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

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, \url{http://ms.math.uni-mannheim.de}
References

variogram method


fft

- Chan, Hall and Poskitt (1995)

See Also

*RMmodel, RFhurst*

Examples

```r
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)
```

---

**RFunction**

*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** assigns 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 is identical to RFcov.
- **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 the design matrix.
- **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

- \texttt{RFcrossvalidate} cross validation for Gaussian fields
- \texttt{RFvariogram} empirical variogram
- \texttt{RFfit} (maximum likelihood) fitting of the parameters
- \texttt{RFinterpolate} ‘kriging’ and ‘imputing’
- \texttt{RFratiotest} likelihood ratio test for Gaussian fields

Graphics for Gaussian fields

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

Coordinate transformations

- \texttt{RFearth2cartesian} transformation of earth coordinates to cartesian coordinates
- \texttt{RFearth2dist} transformation of earth coordinates to Euclidean distances

Information from and to RandomFields

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

Author(s)

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

See Also

- \texttt{RC}, \texttt{RM}, \texttt{RP}, \texttt{RR}, \texttt{R}, \texttt{RMmodelgenerator}
Examples

```r
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
```

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_`

The following methods are available:

- (random spatial) Averages
  - details soon
- Boolean functions.
  See marked point processes.
- circulant embedding.
  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 `RMmodel`, 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 implementa-
tion 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 hyermodel, see RModel1,
and A = B there.
Here, multiplicative models are not allowed (yet).

• Marked point processes.
Some methods are based on marked point process \( \Pi = \bigcup [x_i, m_i] \) where the marks \( m_i \) are
deterministic or i.i.d. random functions on \( \mathbb{R}^d \).

  – 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 var-
iables. 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 RF.options.

• particular method
  – details missing –

• Random coins.
See marked point processes.

• sequential
This method is programmed for spatio-temporal models where the field is mod-
elled 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
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**


Original work:

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

- **Intrinsic embedding and Cutoff embedding:**

- **Markov Gaussian Random Field:**

- **Turning bands method (TBM), turning layers:**

- **Random coins:**

**See Also**

*RMmodel, RFsimulate, RandomFields.*

**Examples**

```r
RFgetMethodNames()
```
**RFgetModel**

*Internally stored model*

**Description**

The function returns the stored model.

**Usage**

```r
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.
  - If `show.call` is a character it behaves as `which.submodels`.
- `origin`: 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](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 `RFgetModelInfo_register` and `RFgetModelInfo_model`; `RFgetModelInfo` 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.
RFgetModelInfo

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.

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.

The value "all" returns the whole tree of models (very advanced).

**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, RFgetModelInfo 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
RFgetModelNames

- 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

commandRFgetModel, 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**

```r
RFgetModelNames(type = RC_TYPE_NAMES, domain = RC_DOMAIN_NAMES,
                 isotropy = RC_ISO_NAMES, operator = c(TRUE, FALSE),
                 monotone = RC_MONOTONE_NAMES,
                 implied_monotonicities = 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_monotonicities`
  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 show 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 `RMmodelgenerator`.

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 correlation functions are returned if "type" is included in `group.by`.

- `type` specifies the class of functions; for the meaning of the possible values see `RMmodelgenerator`.
- `stationarity` specifies the type of stationarity; for the meaning of the possible values see `RMmodelgenerator`.
- `isotropy` specifies the type of isotropy; for the meaning of the possible values see `RMmodelgenerator`.
- `operator` indicates whether the model is an operator, i.e. it requires at least one submodel, e.g. `+` or `RMdelay` are operators; see `RMmodelgenerator`.
- `monotone` indicates what kind of monotonicity is known, e.g., whether the model is a normal scale mixture, the latter including `RMexp` or `RMcauchy`; see `RMmodelgenerator`.
- `finiterange` indicates whether the covariance of the model has finite range, e.g. `RMcircular` or `RMnugget` have covariances with finite range; see `RMmodelgenerator`. 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 `RMmodel` object; `maxdim=-2` means that the maximal possible dimension is adopted from the called submodels; see also `RMmodelgenerator`.
- `vdim` if a positive integer, `vdim` specifies, whether the model is `vdim`-variate; `vdim=-1` means that being multivariate in a certain dimension depends on the parameters of the `RMmodel` object; `vdim=-2` means that being multivariate in a certain dimension is adopted from the called submodels; see also `RMmodelgenerator`.

If `vdim` is bivariate then a range is given.

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

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

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

Value

Either a vector of model names if the argument `group.by` is not used; or a list of vectors of model names if the argument `group.by` is used (with list elements specified by the categories of the grouping argument).
In case internal or newnames is given, `RFgetModelNames` prints a table of the currently implemented covariance functions and the matching methods. `RFgetModelNames` returns NULL.

Note

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

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

```r
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
```
RFgridDataFrame-class

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 conventionalRFspDataFrame 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 \[1 \times (\text{vdim} > 1) + \text{space} - \text{time} - \text{dimension} + 1 \times (n > 1)\]

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
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

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

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

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

RFhurst

<table>
<thead>
<tr>
<th>Hurst coefficient</th>
</tr>
</thead>
</table>

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])/5))),
    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 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.
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 fft.max.length then the segments of length fft.max.length are considered, shifted by fft.max.length/2 (WOSA-estimator).
method list of implemented methods to calculate the Hurst parameter; see Details
mode character. A vector with components 'nographics', 'plot' or 'interactive':
    'nographics' no graphical output
    'plot' the regression line is plotted
    'interactive' the regression domain can be chosen interactively
Usually only one mode is given. Two modes may make sense in the combination c("plot", "interactive") in which case all the results are plotted first, and then the interactive mode is called. In the interactive mode, the regression domain is chosen by two mouse clicks with the left mouse; a right mouse click leaves the plot.
pch vector or scalar; sign by which data are plotted.
cex vector or scalar; size of pch.
cex.main font size for title in regression plot; only used if mode includes 'plot' or 'interactive'
printlevel integer. If printlevel is 0 or 1 nothing is printed. If printlevel=2 warnings and the regression results are given. If printlevel>2 tracing information is given.

height height of the graphics window

Details

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

The function calculates the Hurst coefficient by various methods:

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

Value

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

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

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

Author(s)

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

References

detrended fluctuation analysis


taggregated variation

periodogram


See Also

RMmodel, RFractaldim

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 <- RFrhurst(1:length(x), data=x)

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

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

\texttt{x}  vector of \texttt{x} coordinates, or object of class \texttt{GridTopology} or \texttt{raster}; for more options see \texttt{RFsimulateAdvanced}.

\texttt{y, z}  optional vectors of \texttt{y} (\texttt{z}) coordinates, which should not be given if \texttt{x} is a matrix.

\texttt{T}  optional vector of time coordinates, \texttt{T} must always be an equidistant vector. Instead of \texttt{T=seq(from=From, by=By, len=Len)}, one may also write \texttt{T=c(From, By, Len)}.

\texttt{grid}  logical; the function finds itself the correct value in nearly all cases, so that usually \texttt{grid} need not be given. See also \texttt{RFsimulateAdvanced}.

\texttt{distances, dim}  another alternative for the argument \texttt{x} to pass the (relative) coordinates, see \texttt{RFsimulateAdvanced}.

\texttt{data}  matrix, data.frame or object of class \texttt{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. If \texttt{given} is not given and \texttt{data} is a matrix or \texttt{data} is a data.frame, \texttt{RandomFields} tries to identify where the data and the coordinates are, e.g. by names in formulæ or by fixed names, see Coordinate systems. See also \texttt{RFsimulateAdvanced}. 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 \texttt{x} is missing, \texttt{data} may contain \texttt{NAs}, which are then replaced through imputing.

\texttt{given}  optional, matrix or list. If \texttt{given} matrix then the coordinates can be given separately, namely by \texttt{given} where, in each row, a single location is given. If \texttt{given} is a list, it may consist of \texttt{x, y, z, T, grid}. If \texttt{given} is provided, \texttt{data} must be a matrix or an array containing the data only.

\texttt{err.model, err.params}  For conditional simulation and random imputing only. In case of (assumed) error-free measurements (which is mostly the case in geo-statistics) the argument \texttt{err.model} is not given. In case of measurement errors we have \texttt{err.model=RMnugget(var=var)}. \texttt{err.param} plays the same role as \texttt{params} for model.

\texttt{ignore.trend}  logical. If \texttt{TRUE} only the covariance model of the given model is considered, without the trend part.

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

Details

In case of repeated data, they are kriged \texttt{separately}; if the argument \texttt{x} is missing, \texttt{data} may contain \texttt{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


See Also

RFmodel, 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() + RMtrend(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) + RMtrend(mean=4)
z <- RFinterpolate(model, x=x, y=x, data=dta)
plot(z, dta)

## Ordinary kriging
model <- RMexp() + RMtrend(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 <- RMparswmmX(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$variable1[:((len / 10))] <- NA
data@data$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 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.

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**

```r
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**

```r
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 T=seq(from=From, by=By, len=Len), one may also write T=seq(from=From, by=By, len=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`.
- `distances,dim`: another alternative for the argument x to pass the (relative) coordinates, see `RFsimulateAdvanced`.
RFloglikelihood

**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 params is given, then ... may include also the variables used in params.

**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[[i]] <- list(y = runif(n=pts[i], min=-1, max=1),
}
y = runif(n=pts[, min=-1, max=1))

data[[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=dt)))
str(likeli, digits=8)

L <- 0
for (p in 1:length(pts)) {
  tr <- RFtrend(x=x[[p]]$x, y=x[[p]]$y, spC = FALSE)
  C <- RFCovariance(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

\[
\gamma(r) := \frac{1}{N(r)} \sum_{(t_i, t_j), t_i, t_j = r} \left| (X(t_i) - X(t_j)) \right| \left| (Y(t_i) - Y(t_j)) \right|
\]

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**

\[\text{RFmadogram(model, x, y=NULL, z=NULL, T=NULL, grid, params, distances, dim, ..., data, bin=NULL, phi=NULL, theta = NULL, deltaT = NULL, vdim=NULL)}\]

**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`.
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.
RFmadogram

Author(s)

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

References


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 <- RBiwm(nudiag=c(1, 2), nured=1, rhored=1, cdiag=c(1, 5),
s=c(1, 1, 2))
x <- seq(0, 20, 0.1)
128

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.
RFoptions

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)

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 **Rcirculant**
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. kriging: 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** 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 `every`th iteration is printed if simulated by TBM or random coin method. The value zero means that nothing is printed.
   Default: 0

   **exactness** logical or NA. Currently only used when simulating Gaussian random fields.
   
   - TRUE: `RPcoins`, `RPhyperplane`, `RPsequential`, `RPspectral` and `RPtrbm` and *approximate* 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.
RFoptions

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 ly on a grid.

Default: 1e-6

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

Default: TRUE

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. **Rfit:** 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: 'x'.

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", op=... for the list of primitive models.
• **FALSE**: the practical range adjustment is not used.
• **TRUE**: practical range is applicable only if the value is known exactly, or, at least, can be approximated by a closed formula.
• **2**: if the practical range is not known exactly it is approximated numerically.

Default: **FALSE**.

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

seed + (k − 1) × seed_sub_incr + 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 RPtbm if the line is simulated with any circulant embedding method.

approx_maxgrid See RPcirculant
approx_step See RPcirculant
dependent See RPcirculant
force See RPcirculant
maxGB See RPcirculant
maxmem See RPcirculant
mmin See RPcirculant
strategy See RPcirculant
tolIm See RPcirculant
tolRe See RPcirculant
trials See RPcirculant
useprimes See RPcirculant

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

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 $1+\text{safety}$ 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 \text{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 \text{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: \text{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: \text{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.

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

algorithm See **RFfitOptimiser**.
Default: \text{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 \text{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.
RFOptions

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 RFOptimiser. 
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 $\chi^2$ 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.

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

upperbound_scale_factor * (maximum distance between all pairs of points).
Default: 3.

upperbound_var_factor The upper bound for the variance and the nugget is determined as upperbound_var_factor

* var(data)
Default: 10.
use_naturalscaling: logical. Only used if model is given in standard (simple) way. If TRUE then internally, rescaled covariance functions will be used for which cov(1)≈0.05. use_naturalscaling has the advantage that scale and the form parameters of the model get ‘orthogonal’, but use_naturalscaling does not work for all models.

Note that this argument does not influence the output of Rffit: the parameter vector returned by Rffit refers always to the standard covariance model as given in RMmodel. (In contrast to practicalrange in RFoptions.)

Advantages if use_naturalscaling=TRUE:

- scale and the shape parameter of a parameterised covariance model can be estimated better if they are estimated simultaneously.
- The estimated bounds calculated by means of upperbound_scale_factor and lowerbound_scale_factor, etc. might be more realistic.
- in case of anisotropic models, the inverse of the elements of the anisotropy matrix should be in the above bounds.

Disadvantages if use_naturalscaling=TRUE:

- For some covariance models with additional parameters, the rescaling factor has to be determined numerically. Then, more time is needed to perform Rffit.
- 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(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.

file character; only relevant if split_screen = TRUE. argument file in pdf If "" then no internal naming is performed.
Default: "".

filename 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 the 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 always_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.

Default: "RPcirculant".

size vector of 2 components. Grid size of the simulated stochastic processes. The two components of the vector correspond to one-dimensional and two-dimensional processes, respectively.
Default: c(1024, 64).

12. hyper: **Options for simulating hyperplane tessellations**

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

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

maxlines integer. Maximum number of allowed lines.
Default: 1000 .

superpos integer. number of superposed hyperplane tessellations.
Default: 300 .

13. krig: **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 increases the computing times for taking the maximum over a single shape
functions. Here, after every check.every considered shape function, the global minimum is
calculated. It is expected that a good choice for check.every is in in the interval [10, 100].
(For ease and for concerns of efficiency, the more adequate, local minimum is not considered.)
Default: 30.

densityRatio 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 densityRatio 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

flat hull 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 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 $\xi$ can be set globally, the shift $\mu$ and the scale $s$ can be given only locally within the process definitions, e.g., RPsmit
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 $\varepsilon =$about_zero, $s =$scatter_size and $m =$scatter_max. We distinguish 4 cases:

- $\text{scatter_size} > 0$ and $\text{scatter_max} \geq 0$
  Here, $n$ equals $(2m)^d$, and $h_i \in \mathcal{M} = \{(ks, \ldots, ks), \ldots, (ms, \ldots, ms)\}$ with $k = -m$.
- $\text{scatter_size} > 0$ and $\text{scatter_max} < 0$
  same as the previous case, but $m$ is chosen such that $f(k_i e_i s_i) \approx \varepsilon$, $-k_i \in N$, $i = 1, \ldots, d$ 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 \( i = 1, \ldots, d \), but at most \( \text{scatter}_\text{max} \) steps.
• scatter_size <= 0 and scatter_max < 0
  this option is possible only for grids. Here, \( h_i \) runs over the whole grid.

\text{shape}_\text{power}  
Shape functions are powered by \text{shape}_\text{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
  
• \text{RPnugget}
• \text{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.

\text{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 alternately, several times in a row. Then the simulation speed can be increased if several registers are used, storing=TRUE and \text{RFsimulate} is used with the only argument \( n \).
Default: 0

18. sequ: **Options for the sequential method**
back_steps  See \text{RPsequential}
initial  See \text{RPsequential}
max_variables  See \text{RPsequential}

19. solve: **Options for solving linear systems**
det_as_log  See \text{RFoptions}
eigen2zero  See \text{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 Choleskey 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 $L^tL = M$ 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_relelrro  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 RMult. 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 \texttt{RPspectral}

sigma see \texttt{RPspectral}

sp_grid see \texttt{RPspectral}

sp_lines see \texttt{RPspectral}

\begin{description}
\item[22. tbm:] \textbf{Options for the turning bands method}
\end{description}

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.

\begin{description}
\item[Default:] \texttt{NA}.
\end{description}

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.

\begin{description}
\item[Default:] 3.
\end{description}

grid Logical. The angle of the lines is random if grid=\texttt{FALSE}, and $k\pi/\text{lines}$ for $k$ in 1:lines, otherwise.

\begin{description}
\item[This option is used by both \texttt{RPspectral} and \texttt{RPtbm}, the latter only when the dimension is 2.]
\item[Default:] \texttt{TRUE}.
\end{description}

layers Logical or integer. If \texttt{TRUE} then the turning layers are used whenever a time component is given. If \texttt{NA} the turning layers are used only when the traditional TBM is not applicable. If FALSE then turning layers may never be used.

\begin{description}
\item[Default:] \texttt{TRUE}.
\end{description}

lines Number of lines used.

\begin{description}
\item[Default:] 60.
\end{description}

linesimustep If \texttt{linesimustep} is positive the grid on the line has lag \texttt{linesimustep}. See also \texttt{linesimufactor}.

\begin{description}
\item[Default:] 0.0.
\end{description}

linesimufactor \texttt{linesimufactor} or \texttt{linesimustep} must be non-negative; if \texttt{linesimustep} is positive then \texttt{linesimufactor} is ignored. If both arguments are naught then points is used (and must be positive). The grid on the line is \texttt{linesimufactor}-times finer than the smallest distance. See also \texttt{linesimustep}.

\begin{description}
\item[Default:] 2.0.
\end{description}

points integer. If greater than 0, \texttt{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 \texttt{TBMx.linesimufactor} and \texttt{TBMx.linesimustep}. If \texttt{points} is not positive the number of points is determined automatically. The use of center and \texttt{points} is highlighted in an example below.

\begin{description}
\item[Default:] 0.
\end{description}

reduceddim if positiv integer, then the value itself. If negativ, then the value is substracted from fulldim.

\begin{description}
\item[Default:] -2.
\end{description}

\begin{description}
\item[23. internal:] \textbf{Internal options mostly for warnings and messages}
\item[All these options should not be changed by the user unless he/she really known what he/she is doing.]
\item[Most of the options below change their value in a session without the user’s notice.]
\end{description}
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 if 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 \texttt{RFFit} displays the message that other values for the option \texttt{modus_operandi} are available.

Default: \texttt{TRUE}.

warn_oldstyle  internally used logical argument. If \texttt{TRUE} a warning is given if an obsolete function from Version 2 is used.

Default: \texttt{TRUE}.

warn_on_grid  internally used logical argument. If a (one-dimensional) grid is given, but the argument \texttt{grid=FALSE}, e.g. in \texttt{RFsimulate}, this contraction is reported if \texttt{warn_on_grid=TRUE}

Default: \texttt{TRUE}.

warn_scale  internally used logical argument. If \texttt{warn_scale=TRUE} then a scale less than 10 [km] is reported if earth coordinates are transformed to cartesian coordinates.

Default: \texttt{TRUE}.

warn_var  In some cases, \texttt{RandomFields} cannot detect whether the variance is non-negative. If \texttt{TRUE} then a warning is displayed in such a case.

Default: \texttt{TRUE}.

Value

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

Author(s)

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

References

Basic

• General

• \texttt{rectangular} distribution; \texttt{eps_zhou}

• \texttt{shape_power}

See Also

\texttt{RFsimulate}, \texttt{RFoptionsAdvanced}, \texttt{RFoptions}, and \texttt{RFgetMethodNames}. 
RFoptionsAdvanced

Setting control arguments of RandomFields – advanced examples

Examples

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

RFoptions()

#############################################################
## use of exactness
##
#############################################################
x <- seq(0, 1, 1/30)
model <- RMgauss()
for (exactness in c(NA, FALSE, TRUE)) {
  readline(paste("\n\nexactness: ", exactness, "; press return"))
  z <- RFsimulate(model, x, x, exactness=exactness,
                  stationary_only=NA, storing=TRUE)
  print(RFgetModelInfo(which="internal")$internal$name)
}

Description

Some more complex examples for the use of RFoptions are given.

Author(s)

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

Examples

#############################################################
## The following gives an example on the advantage of
## dependent=TRUE for simulating with RPcirculant if, in a
## study, most of the time is spent with simulating the
## Gaussian random fields. Here, the covariance at a pair
## of points is estimated for n independent repetitions
## and 2*n locally dependent repetitions.
## To get the precision, the procedure is repeated m times.
#############################################################
# In the example below, local.dependent speeds up the simulation
# by about factor 16 at the price of an increased variance of
# factor 1.5

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 <- Rcirculant
model <- RMexp(var=1.1, Aniso=matrix(nc=2, c(2,0.1,1.5,1)))

m <- 5
n <- 100

c1 <- numeric(m)
time <- system.time(  
  for (i in 1:m) {  
    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])  
  })  
print(time)
Print(true.cov, mean(c1), sd(c1), empty.lines=1)## true mean is zero

# using local.dependent=FALSE (which is the default)
c2 <- numeric(m)
time <- system.time(  
  for (i in 1:m) {  
    z <- Rfsimulate(meth(model), x, y, n=n2 * 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 n2 * n,
## the value of sd(c2) would be larger due to the local dependencies
## in the realisations.
RFpar

Graphical parameters for plots

Description

This function sets globally graphical parameters for plots of \texttt{RMmodels}, simulations and estimations.
RFpointsDataFrame-class

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`.
**RFpseudomadogram**

**Description**

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

\[
\gamma(r) := \frac{1}{N(r)} \sum_{(t_i, t_j) : |t_i - t_j| = r} |(X(t_i) - X(t_j))||Y(t_i) - Y(t_j)|
\]

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

**Usage**

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

- **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`;

- **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 `seq(from=0, to=deltat[1], by=deltat[2])`

- **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**


**See Also**

`RMstable, RMmodel, RFsimulate, RFfit, RFcov, RFmadogram, RFvariogram`.

**Examples**

```r
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)
```

**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 `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`.
- `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 `seq(from=0, to=deltaT[1], by=deltaT[2])`.
- `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 \texttt{RFempVariog}.

Author(s)

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

References


See Also

\texttt{RMstable}, \texttt{RMmodel}, \texttt{RFsimulate}, \texttt{RFit}, \texttt{RFcov}, \texttt{RFmadogram}, \texttt{RFvariogram}.

Examples

\begin{verbatim}
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)
\end{verbatim}

\begin{verbatim}
RFRatiotest
\end{verbatim}

\textit{Likelihood ratio test}

Description

The function performs an approximate chi2 test or a Monte Carlo likelihood ratio test based on \texttt{fitgauss}. Currently, it only works for Gaussian random fields.

Usage

\begin{verbatim}
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, ...)
\end{verbatim}
Arguments

nullmodel, alternative

See Details.

The set of parameters to be estimated for nullmodel should be a subset of the parameters to be estimated for alternative if alternative is given.

alpha

value in [0,1] or missing. Significance level.

n

integer. The test is based on n=1 simulations.

seed

integer. If not NULL and not NA, the .Random.seed is set to seed. Otherwise, set.seed is set to the value of RFoptions()$basic$seed if the latter is not NA.

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

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.

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 RFfit for 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

nullmodel (and the alternative) can be

- a covariance model, see \texttt{RMmodel} or type \texttt{RFgetModelNames(type=\texttt{variogram})} to get all options.
  
  If \texttt{RFoptions ratiotest\_approx} is \texttt{TRUE} the chisq approximation is performed. Otherwise a Monte Carlo ratio test is performed.
- \texttt{RFfit} or \texttt{RMmodelFit}
  
  Here, a chisq approximative test is always performed on the already fitted models.

\texttt{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

- \texttt{p}, the \textit{p}-value
- \texttt{n}
- \texttt{data\_ratio} the log ratio for the data
- \texttt{simu\_ratio} the log ratio for the simulations
- \texttt{data\_fit} the models fitted to the data
- \texttt{msg} the message that is also directly returned

It has S3 class \texttt{\texttt{RFratiotest}}.

Methods

- \texttt{print} prints the summary
- \texttt{summary} gives a summary

Note

An important \texttt{RFoptions} is \texttt{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](http://ms.math.uni-mannheim.de)

**See Also**

`RFFit`, `RMmodel`, `RandomFields`, `weather`

**Examples**

---

**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**

```r
RFsimulate(model, x = NULL, 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 `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 `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`.

distances, dim  another alternative for the argument `x` to pass the (relative) coordinates, see `RFsimulateAdvanced`.

data  For conditional simulation and random imputing only. If `data` is missing, unconditional simulation is performed.

    matrix, data.frame or object of class `RFsp`;
    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 Coordinate systems. See also `RFsimulateAdvanced`. 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 `NA`s, which are then replaced by conditionally simulated values (random imputing);

given  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.

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 `err.model` is not given. In case of measurement errors we have `err.model=RMnugget(var=var)`.`
    `err.param` plays the same role as `params` for `model`.

n  number of realizations to generate. For a very advanced feature, see the notes in `RFsimulateAdvanced`.

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

Author(s)

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

References


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 #
# ?RFsimulate.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)

# 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

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

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** object of class `RMmodel`, `RFformula` or `formula`; specifies the model to be simulated
  - if of class `RMmodel`, `model` specifies
    - the type of random field by using `RPfunctions`, e.g.,
      * `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** matrix of coordinates, or vector of x coordinates, or object of class `GridTopology` or `raster`; if matrix, `ncol(x)` is the dimension of the index space; matrix notation is required in case of more than 3 space dimensions; in this case, if `grid=FALSE`, `x_{ij}` is the i-th coordinate in the j-th dimension; otherwise, if `grid=TRUE`, the columns of `x` are interpreted as gridtriples (see `grid`); if of class `GridTopology`, `x` is interpreted as grid definition and `grid` is automatically set to `TRUE`.

- **y** optional vector of y coordinates, ignored if `x` is a matrix

- **z** optional vector of z coordinates, ignored if `x` is a matrix

- **t** optional vector of time coordinates, `T` must always be an equidistant vector or given in a gridtriple format (see argument `grid`); for each component of `T`, the random field is simulated at all location points.

- **grid** logical; determines whether the vectors `x`, `y`, and `z` or the columns of `x` should be interpreted as a grid definition (see Details). If `grid=TRUE`, either `x`, `y`, and `z` must be equidistant vectors in ascending order or the columns of `x` must be given in the gridtriple format: `c(from, stepsize, len)`. Note: If `grid` is not given, `RFsimulate` tries to guess what is meant. `c(from, stepsize, len)` (see Details)
RFsimulateAdvanced

**data**
matrix, data.frame or object of class RFsp; coordinates and response values of measurements in case that conditional simulation is to be performed; if a matrix or a data.frame, the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field; if x is missing, data may contain NAs, which are then replaced by conditionally simulated values; if data is missing, unconditional simulation is performed; for details on matching of variable names see Details; if of class RFsp

**err.model**
same as model; gives the model of the measurement errors for the measured data (which must be given in this case!), see Details. err.model=NULL (default) corresponds to error-free measurements, the most common alternative is err.model=RMnugget(); ignored if data is missing.

**distances**
object of class dist representing the upper 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 NULL, dim must be given and x, y, z and T must be missing or NULL.

If distances are given, the current value of spConform, see RFoptions, is ignored and instead spConform=FALSE is used. (This fact may change in future.)

**dim**
integer; space or space-time dimension of the field

**n**
number of realizations to generate

... further options and control arguments for the simulation that are passed to and processed by RFoptions

**Details**

**RFsimulate** simulates different classes of random fields, controlled by the wrapping model.

If the wrapping function of the model argument is a covariance or variogram model, i.e., one of 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, ...,nrow(x), k = 1, ...,length(T), even if model is not explicitly a space-time model.

If grid=TRUE, the vectors x, y, z and T or the columns of x and T are interpreted as a grid definition, i.e. the field is simulated at all locations (x[i], y[j], z[k], T[l]), as given by expand.grid(x, y, z, T). Here, “grid” means “equidistant in each direction”, i.e. all vectors must be equidistant and in ascending order. In case of more than 3 space dimensions, the coordinates must be given in matrix 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])...
If data is passed, conditional simulation is performed.

- If of class \texttt{RFsp}, \texttt{ncol(data@coords)} must equal the dimension of the index space. If \texttt{data@data} contains only a single variable, variable names are optional. If \texttt{data@data} contains more than one variable, variables must be named and \texttt{model} must be given in the tilde notation \texttt{resp \sim \ldots} (see \texttt{RFformula}) and "resp" must be contained in \texttt{names(data@data)}.

- If \texttt{data} is a matrix or a data.frame, either \texttt{ncol(data)} equals \texttt{(dimension of index space + 1)} and the order of the columns is \texttt{(x, y, z, T, response)} or, if \texttt{data} contains more than one response variable (i.e. \texttt{ncol(data) > (dimension of index space + 1)}), \texttt{colnames(data)} must contain \texttt{colnames(x)} or those of \texttt{"x", "y", "z", "T"} that are not missing. The response variable name is matched with \texttt{model}, which must be given in the tilde notation. If \texttt{"x", "y", "z", "T"} are missing and \texttt{data} contains NAs, \texttt{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 \texttt{x} is missing, \texttt{RFsimulate} searches for NAs in the data and performs a conditional simulation for them.

Specification of \texttt{errNmodel}: 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 \texttt{errNmodel} must be given separately. For sake of generality, any model (and not only the nugget effect) is allowed. Consequently, \texttt{errNmodel} is ignored when unconditional simulation is performed.

\textbf{Value}

By default, an object of the virtual class \texttt{RFsp}; result is of class \texttt{RFspatialGridDataFrame} if \texttt{[space\textendash time\textendash dimension > 1]} and the coordinates are on a grid, result is of class \texttt{RFgridDataFrame} if \texttt{[space\textendash time\textendash dimension = 1]} and the coordinates are on a grid, result is of class \texttt{RFspatialPointsDataFrame} if \texttt{[space \textendash time \textendash dimension > 1]} and the coordinates are not on a grid, result is of class \texttt{RFpointsDataFrame} if \texttt{[space \textendash time \textendash dimension = 1]} and the coordinates are not on a grid.

The output format can be switched to the "old" array format using \texttt{RFoptions}, either by globally setting \texttt{RFoptions(spConform=FALSE)} or by passing \texttt{spConform=FALSE} in the call of \texttt{RFsimulate}. Then the object returned by \texttt{RFsimulate} depends on the arguments \texttt{n} and \texttt{grid} in the following way:

If \texttt{vdim > 1} the \texttt{vdim}-variate vector makes the first dimension.

If \texttt{grid=TRUE} an array of the dimension of the random field makes the next dimensions. Here, the dimensions are ordered in the sequence \texttt{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 \texttt{n > 1} the repetitions make the last dimension.

Note: Conversion between the \texttt{sp} format and the conventional format can be done using the method \texttt{RFspDataFrame2conventional} and the function \texttt{conventional2RFspDataFrame}.

\texttt{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


Original work:

- Circulant embedding:
  The code used in `RandomFields` is based on Dietrich and Newsam (1996).

- Intrinsic embedding and Cutoff embedding:

- Markov Gaussian Random Field:
• Turning bands method (TBM), turning layers:
  Dietrich, C.R. (1995) A simple and efficient space domain implementation of the turning
  Probab. 5, 439-468.

• Random coins:

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  

\textit{Class RFsp}

Description

"RFsp" is a virtual class which contains the four classes \texttt{RFspatialGridDataFrame} (data on a full
grid and space – time – dimension \( \geq 2 \)), \texttt{RFspatialPointsDataFrame} (data not on a grid and
space – time – dimension \( \geq 2 \)), \texttt{RFgridDataFrame} (data on a full grid and space – time –
dimension = 1), \texttt{RFpointsDataFrame} (data not on a grid and space – time – dimension = 1).

The first two subclasses are summarized in "RFspatialDataFrame" whilst the latter two are sum-
grORIZED in "RFdataFrame".

Objects from the Class

are never to be generated; only derived classes can be meaningful.
RFsp-class

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](http://ms.math.uni-mannheim.de)

See Also

- RFspatialGridDataFrame, RFspatialPointsDataFrame, RFgridDataFrame, RFpointsDataFrame, spRrf

Examples

```r
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)
```
```r
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 `spRF` for an explicit transformation.

**Usage**

```r
## 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-dim 2, 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 \[1 \times (vdim > 1) + space \times time \times dimension + 1 \times (n > 1)\]; 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 \[space \times time \times dimension + 2\], 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),
  cells=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
RFspatialPointsDataFrame-class

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-dim = 2, 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 SpatialPointsDataFrame

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

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
RFvariogram

Empirical (Cross-)Variogram

Description
Calculates empirical (cross-)variogram.

Usage
RFvariogram(model, x = NULL, 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.
RFvariogram

- **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`.

- **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.

- **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 `seq(from=0, to=deltat[1], by=deltat[2])`

- **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**

`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 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 not 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


See Also

`RMstable`, `RMmodel`, `RFsimulate`, `RFfit`, `RFcov`, `RFpseudovariogram`, `RFmadogram`.

Examples

```r
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
```r
t <- RMbivm(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=n, n=n)
emp.vario <- RFvariogram(data=z)
plot(emp.vario, model=model)
```

## multivariate and anisotropic model
```r
t <- RMbivm(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)
da <- RFsimulate(model, x, x, n=n)
ev <- RFvariogram(data=da, phi=4)
plot(ev, model=model, boundaries=FALSE)
```

### 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 \((d_1, d_2)\) the anisotropy matrix \(A\) is

\[
A = \text{diag}(d_1, d_2) \quad \text{matrix(ncol=2, c(cos(a), sin(a), -sin(a), cos(a)))}
\]

In three dimensions and with angle equal to \(a\), second angle \(L\) and diag equal to \((d_1, d_2, d_3)\) the anisotropy matrix \(A\) is

\[
A = \text{diag}(d_1, d_2, d_3) \quad \text{matrix(ncol=3, c(cos(a) * cos(L), sin(a) * cos(L), sin(L), -sin(a), cos(a), 0))}
\]

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

- **angle**: angle \(a\)
- **lat.angle**: second angle; in 3 dimensions only
- **ratio**: equivalent to diag=c(1, 1/ratio); in 2 dimensions only
- **diag**: the diagonal components of the matrix

### Value

RMangle returns an object of class RMmodel.
Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>, 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, model=model)
plot(z, MARGIN.slices=3)

#* next model gives an example how to estimate the parameters back
n <- 20
x <- rnorm(n, 0, 10)
y <- rnorm(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, modusOperandi='sloppy'))

---

RMaskey Askey model

Description
Askey’s model

\[ C(x) = (1 - x)^\alpha 1_{[0,1]}(x) \]

Usage
RMaskey(alpha, var, scale, Aniso, proj)
RMtentr(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


Applications as covariance function


Tail correlation function (for \(\alpha \geq \lceil d/2 \rceil + 1\))


See Also

RMmodel, RMbigneiting, RMgengneiting, RMgneiting, RFsimulate, RFfit.
**Examples**

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

model <- RMcov()
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 \((h, u) \in \mathbb{R}^d\) 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**

```r
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


See Also

RFFit, RFsimulate, RMmodel, RMstp.

Examples

```r
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)
```
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, RMspheic, 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(RPoisson(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**

```r
Rmbcw(alpha, beta, c, var, scale, Aniso, proj)
```

**Arguments**

- **alpha**: a numerical value; should be in the interval \( (0, 2] \).
- **beta**: a numerical value; should be in the interval \( (-\infty, 2] \).
- **c**: only for experts. If given, a not necessarily positive definite function \( c - \gamma(r) \) is built.
- **var, scale, Aniso, proj**: 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 Wisjian model Rmdewijsian, respectively.

Hence its two arguments alpha and beta allow for modelling the smoothness and a wide range of tail behaviour, respectively.

**Value**

**Rmbcw** returns an object of class **Rmmodel**

**Author(s)**

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

**References**

See Also

`RMlsfbd` is equipped with Matheron’s constant \( c \) for the fractional brownian motion, `RMgenfbm`, `RMgencauchy`, `RMdewijsian`, `RMmodel`, `RFsimulate`, `RFfit`.

Examples

```r
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

```r
RMbernoulli(phi, threshold, correlation, centred, var, scale, Aniso, proj)
```

Arguments

- **phi**: covariance function of class `RMmodel`.
- **threshold**: real valued threshold, see `RPbernoulli`. Currently, only \( \text{threshold}=0.0 \) 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](http://ms.math.uni-mannheim.de)

**References**

Ballani, Schlather

**See Also**

`rpbernoulli`, `rmmodel`, `rfsimulate`.

**Examples**

```r
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")
```

### 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 $\nu \geq \frac{d-2}{2}$, $\Gamma$ denotes the gamma function and $J_\nu$ is a Bessel function of first kind.
Usage

```r
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


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

\[
C(r) = \frac{\sin(r)}{r}
\]

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](http://ms.math.uni-mannheim.de)

References


See Also

`RMdampedcos`, `Rmwave`, `RMmodel`, `RFSimulate`, `RFFit`. 
**Examples**

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

model <- RMcessel(nu=1, scale=0.1)  
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

---

**RMcicauchy**  
*Biivariate Cauchy Model*

**Description**

**RMcicauchy** 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**

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

**Arguments**

- `alpha`  
- `beta`  
- `s`  
- `rho`  
- `var, scale, Aniso, proj`

Optional arguments; same meaning for any **RModel**. If not passed, the above covariance function remains unmodified.

**Details**

Constraints on the constants: [to be done]

**Value**

**RMcicauchy** returns an object of class **RModel**.

**Author(s)**

Martin Schlather, <schlather@math.uni-mannheim.de>, [http://ms.math.uni-mannheim.de](http://ms.math.uni-mannheim.de)
References


See Also

*RMcauchy, Multivariate RMmodels.*

Examples

```r
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 \( \delta \), 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, \ldots, 3. The model is $k$ times differentiable.

mu
mu has to be greater than or equal to $\frac{d}{2}$ where $d$ is the (arbitrary) dimension of the 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 $s_{12} = s_{21} = s_{red12} \times \min\{s_{11}, s_{22}\}$.

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.

Details

A sufficient condition for the constant $c_{1j}$ is

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

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

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

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

Let

$$ a = 2\gamma_{12} - \gamma_{11} - \gamma_{22} $$

$$ b = -2\gamma_{12}(s_{11} + s_{22}) + \gamma_{11}(s_{12} + s_{22}) + \gamma_{22}(s_{11} + s_{12}) $$

$$ e = 2\gamma_{12}s_{11}s_{22} - \gamma_{11}s_{12}s_{22} - \gamma_{22}s_{11}s_{12} $$

$$ d = b^2 - 4ae $$

$$ t_j = \frac{-b + j\sqrt{d}}{2a} $$

If $d \geq 0$ and $t_j \not\in (0, s_{12})$ then $m_j = \infty$ else

$$ m_j = \frac{(1 - t_j/s_{11})^{\gamma_{11}}(1 - t_j/s_{22})^{\gamma_{22}}}{(1 - t_j/s_{12})^{2\gamma_{11}}} $$

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.
RMbistable

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).

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))
```

RMbistable                  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 \textbf{RMmodel}. If not passed, the above covariance function remains unmodified.

Details

Constraints on the constants: [to be done]

Value

\textbf{RMbistable} returns an object of class \textbf{RMmodel}.

Author(s)

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

References


See Also

\textbf{RMstable, Multivariate RMmodels}.

Examples

\begin{verbatim}
RFOptions(seed=0) ## *ANY* simulation will have the random seed 0; set 
##                    RFOptions(seed=NA) to make them all random again

## todo
\end{verbatim}
**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 \( W_i \) 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 \((\nu_{11}, \nu_{22})\).
- **nured12**: a numerical value in the interval \([1, \infty)\); \(\nu_{21}\) is calculated as \(0.5(\nu_{11} + \nu_{22})\)\(\times\)\(\nu_{red}\).
- **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

\[s_{12} = s_{21} > 0,\]
\[ \nu_{12} = \nu_{21} = 0.5(\nu_{11} + \nu_{22}) \nu_{\text{red}} \]

for some constant \( \nu_{\text{red}} \in [1, \infty) \) and

\[ c_{12} = c_{21} = \rho_{\text{red}} \sqrt{mc_{11}c_{22}} \]

for some constant \( \rho_{\text{red}} \in [-1, 1] \).

The constants \( f \) and \( m \) in the last equation are given as follows:

\[ f = \frac{\Gamma(\nu_{11} + d/2)\Gamma(\nu_{22} + d/2)}{(\Gamma(\nu_{11})\Gamma(\nu_{22}))(\Gamma(\nu_{12})/\Gamma(\nu_{12} + d/2))^2} \left( \frac{s_{12}^{2\nu_{12}}}{s_{11}^{\nu_{11}}s_{22}^{\nu_{22}}} \right)^2 \]

where \( \Gamma \) 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

\[ g(t) = \left( 1/s_{12}^2 + t^2 \right)^{2\nu_{12} + d} \left( 1/s_{11}^2 + t^2 \right)^{-\nu_{11} - d/2} \left( 1/s_{22}^2 + t^2 \right)^{-\nu_{22} - d/2} \]

(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


See Also

`rmparswm`, `RMwhittle`, `RMmodel`, `RFSimulate`, `RFFit`, `Multivariate RMmodels`.

Examples

```R
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))
```
Scale model for a few areas of different scales and/or differentiabilities

Description

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

\[
Y(x) = \sum_{i=1}^{k} Z_i \cdot 1(x \in A_i)
\]

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

Usage

\texttt{RMblend(multi, blend, thresholds, var, scale, Aniso, proj)}

Arguments

\begin{itemize}
  \item \texttt{multi} a multivariate covariance function
  \item \texttt{blend, thresholds} The threshold is a vector of increasing values. If the value of \texttt{blend} is below all thresholds up to the \( k \)-th threshold, then the \( k \)-th component of the field given by \texttt{multi} is taken. If necessary the components are recycled. Default: \texttt{threshold} = 0.5, useful for blending a bivariate field if \texttt{blend} takes only the values 0 and 1.
  \item \texttt{var, scale, Aniso, proj} optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.
\end{itemize}

Value

\texttt{RMblend} returns an object of class \texttt{RMmodel}.

Author(s)

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

References

\begin{itemize}
  \item Genton, Apanovich Biometrika.
\end{itemize}

See Also

\texttt{RMSadvanced, RMBubble, RMscale}.
Examples

```r
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, cdimg=c(1, 1),
               s=c(1, 1, 0.5))
model <- Rmblnd(multi=biwm, blend=Rmcovariate(data = as.double(m), raw=TRUE))
plot(z <- RFsimulate(model, x, x)) ## takes a while ...
```

---

**RMbr2bg**

*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

\[
C_{bg}(h) = \cos\left(\pi \left(2\Phi\left(\sqrt{\gamma(h)/2}\right) - 1\right)\right)
\]

Here, \(\Phi\) is the standard normal distribution function, and \(\gamma\) 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**

```r
RMbr2bg(phi, var, scale, Aniso, proj)
```

**Arguments**

- `phi`: covariance function of class `RMmodel`.
- `var, scale, Aniso, proj`: optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

**Details**

The binary random field `RPbernoulli` simulated with `RMbr2bg(RMmodel())` has an uncentered covariance function that equals

1. the tail correlation function of the max-stable process constructed with this binary random field
2. the tail correlation function of Brown-Resnick process with variogram \texttt{RMmodel}.

Note that the reference paper is based on the notion of the (genuine) variogram, whereas the package \texttt{RandomFields} is based on the notion of semi-variogram. So formulae differ by factor 2.

Value

object of class \texttt{RMmodel}

Author(s)

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

References


See Also

\texttt{maxstableAdvanced, RMbr2eg, RMmodel, RMM2r, RBernoulli, RBrownresnick, RPschlather}.

Examples

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

model <- RMeK(var=1.62 / 2)
x <- seq(0, 10, 0.05)
z <- RFsimulate(RPschlather(RMbr2eg(model)), x, x)
plot(z)
\end{verbatim}

\begin{verbatim}
RMbr2eg
\end{verbatim}  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, \( \Phi \) is the standard normal distribution function, and \( \gamma \) is a semi-variogram with sill

\[ 4(\text{erf}^{-1}(1/\sqrt{2}))^2 = 2 \cdot [\Phi^{-1}([1 + 1/\sqrt{2}]/2)]^2 = 4.425098/2 = 2.212549 \]
Usage

```
RMbr2eg(phi, var, scale, Aniso, proj)
```

Arguments

- `phi`: covariance function of class `RMmodel`.
- `var, scale, Aniso, proj`: optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

Details

`RMbr2eg`

The extremal Gaussian model `RPschlather` simulated with `RMbr2eg(RMmodel())` has tail correlation function that equals the tail correlation function of Brown-Resnick process with variogram `RMmodel`.

Note that the reference paper is based on the notion of the (genuine) variogram, whereas the package `RandomFields` is based on the notion of semi-variogram. So formulae differ by factor 2.

Value

Object of class `RMmodel`.

Author(s)

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

References


See Also

`maxstableAdvanced, RMbr2bg, RMmodel, RMM2r, RPsbernoulli, 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 <- RMEmp(var=1.62 / 2)
binary.model <- RPsbernoulli(RMbr2bg(model))
x <- seq(0, 10, 0.05)

z <- RFSimulate(RPschlather(binary.model), x, x)
plot(z)
```
Tail correlation function of the Brown-Resnick process

**Description**

RMBrownresnick defines the tail correlation function of the Brown-Resnick process.

\[ C(h) = 2 - 2\Phi(\sqrt{\gamma(h)/2}) \]

where \( \phi \) is the standard normal distribution function and \( \gamma \) is the semi-variogram.

**Usage**

RMBrownresnick(phi, var, scale, Aniso, proj)

**Arguments**

- **phi**: variogram of class RModel.
- **var, scale, Aniso, proj**: optional arguments; same meaning for any RModel. If not passed, the above covariance function remains unmodified.

**Details**

For a given RModel the function RMBrownresnick(RModel()) 'returns' the tail correlation function of a Brown-Resnick process with variogram RModel.

**Value**

object of class RModel

**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**

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 \texttt{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 \( \text{mins}cale = (10, 10, 10, 7, 7, 7, 0.5) \). Then for some \( d \)-dimensional vectors \( z_1, \ldots, z_7 \) we have

\[
\begin{align*}
z_s &= \{z_1, z_2, z_3\}, s \geq 10 \\
z_s &= \{z_4, z_5, z_5\}, 7 \geq s < 10 \\
z_s &= \{z_7\}, s \geq 0.5
\end{align*}
\]

Note that, in this case, all realized scaling values must be \( \geq 0.5 \). 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

\texttt{RMbubble} returns an object of class \texttt{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

See Also
RMSadvanced, RMBblend, 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 RMBblend, 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),
RMcauchy

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 $\gamma$ determines the asymptotic power law. The smaller $\gamma$, the longer the long-range dependence. The covariance function is very regular near the origin, because its Taylor expansion only contains even terms and reaches its sill slowly.

Each covariance function of the Cauchy Family is a normal scale mixture.

The generalized Cauchy Family (see RMgencauchy) includes this family for the choice $\alpha = 2$ and $\beta = 2\gamma$. The generalized Hyperbolic Family (see RMhyperbolic) includes this family for the choice $\xi = 0$ and $\gamma = -\nu/2$; in this case scale=$\delta$.

Value

RMcauchy returns an object of class RMmodel.

Author(s)

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

References


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))
RMcauchyxtbm

 Modifications of the Cauchy Family Covariance Model

Description

RMcauchyxtbm() is a shortcut of RMtbm(RMgencauchy()) and is given here for downwards compatibility.

Usage

RMcauchyxtbm(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

RMcauchyxtbm returns an object of class RMmodel.

Author(s)

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

References


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 <- RMcauchyxtbm(alpha=1, beta=1, gamma=3)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFSimulate(model, x=x))
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 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 $\psi_d$ of all real valued functions on $[0, \pi]$, with $\psi(0) = 1$ and such that the associated isotropic function

$$h(x, y) = \psi(\text{theta}) with \cos(\theta) = <x, y>$$

for $x, y \in \mathbb{R}^d : ||x|| = 1$

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 RMMmodel.

Author(s)

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


See Also

RMmodel, RFsimulate, RFfit, spherical models, RMmultiquad, RMSinepower

Examples

```r
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

$$C(r) = 1 - 2/\pi (r \sqrt{(1 - r^2)} + \arcsin(r)) \mathbb{1}_{[0,1]}(r).$$

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


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))
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)

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., $W(x) \stackrel{d}{=} Z(x) - Z(0)$ 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 $\gamma$, $a_i$, 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 = NULL, 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 \texttt{RFSimulate} to define the locations where the co-
              variates 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 \texttt{RModel}. 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

\texttt{RMcov} returns an object of class \texttt{RModel}.

Author(s)

Martin Schlather, \texttt{<schlather@math.uni-mannheim.de>}

See Also

\texttt{RModel}, \texttt{RFSimulate}, \texttt{RFfit}.

Examples

\begin{verbatim}
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
\end{verbatim}
RMcovariate

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 RMcovariate passes
  several covariates.

factor
  real value. From user’s point of view very much the same as setting the argument
  var.

Details

The function interpolates (nearest neighbour) between the values.
Value

\texttt{RMcovariate} returns an object of class \texttt{RMmodel}.

Note

- \texttt{x}, \texttt{y} also accept lists of data. However, its use is not in an advanced stage yet.

Author(s)

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

See Also

\texttt{RMfixcov}, \texttt{RMmodel}, \texttt{RMrandom}

Examples

```
RFoptions(seed=0)  ## *ANY* simulation will have the random seed 0; set
RFoptions(seed=NA)  ## RF 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))
```

---

\texttt{RMcoxisham}  \textit{Cox Isha Covariance Model}

Description

\texttt{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 \((h,t) \in \mathbb{R}^{d+1} = \mathbb{R}^d \times \mathbb{R}\) between two points in \(d + 1\)-dimensional space and is given by

\[
C(h,t) = \left| E + t^\beta D \right|^{-1/2} C_0\left(\left( h - t \mu \right)^T (E + t^\beta D)^{-1} (h - t \mu) \right)^{1/2}
\]

Here \(\mu \in \mathbb{R}^d\) 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 \(\beta\) 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 \texttt{RMmodel} whose monotone property equals 'normal mixture', see \texttt{RFgetModelNames(mixture="normal mixture")}

and whose maxdim is at least 2.

mu: a vector in \( d \)-dimensional space

\( D \): a \( d \times d \)-correlation matrix with \(|D| > 0\)

beta: numeric in the interval \((0, 2]\); default value is 2

var, scale, Aniso, proj: optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.

Details

This model stems from a rainfall model (cf. Cox, D.R., Isham, V.S. (1988)) and equals the following expectation

\[ C(h, t) = E_{V} C_{0}(h - Vt) \]

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

\texttt{RMcoxisham} returns an object of class \texttt{RMmodel}.

Author(s)

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

References


See Also

\texttt{RMmodel, RFSimulate, RFfit}.

Examples

\begin{verbatim}
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)
\end{verbatim}
RMcubic

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


See Also

RMmodel, RFsimulate, RFfit.
**Examples**

```r
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**

```r
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

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))

Description
RMcutoff is a functional on univariate stationary isotropic covariance functions \( \phi \).

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

\[
C(r) = \phi(r), 0 \leq r \leq d
\]

\[
C(r) = b_0 ((dR)^{\alpha} - r^\alpha)^2, d \leq r \leq dR
\]

\[
C(r) = 0, dR \leq r
\]

The parameters \( R \) and \( b_0 \) are chosen internally such that \( C \) is a smooth function.

Usage
RMcutoff(phi, diameter, a, var, scale, Aniso, proj)
Arguments

phi  a univariate stationary isotropic covariance model. See, for instance, RFgetModelNames(type="positive definite", domain="single variable", isotropy="isotropic").
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 RModel. 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 $\phi$ and the parameters.

For certain models $\phi$, e.g. Rmstable, Rmwhittle and Rmgencauchy, some sufficient conditions are known (cf. Gneiting et al. (2006)).

Value

RMcutoff returns an object of class RModel.

Author(s)

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

References


See Also

RModel, RFsimulate, RFfit.

Examples

ROptions(seed=0)  ## *ANY* simulation will have the random seed 0; set
## ROptions(seed=NA) to make them all random again
model <- RMexp()
plot(model, model.cutoff=RMcutoff(model, diameter=1), xlim=c(0, 4))
RMedagum

Dagum Covariance Model Family

Description

RMedagum 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

$$C(r) = 1 - (1 + r^{-\beta})^{-\gamma}.$$ 

The parameters $\beta$ and $\gamma$ can be varied in the intervals $(0, 1]$ and $(0, 1)$, respectively.

Usage

RMedagum(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 RModel. 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

RMedagum returns an object of class RModel.

Author(s)

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

References

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 <- RMdampedcos(lambda=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 \( \lambda \) has to be respected:

\[
\lambda \geq 1/\tan(\pi/(2d)).
\]

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

\texttt{RMdampedcos} returns an object of class \texttt{RMmodel}.

Author(s)

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

References


See Also

\texttt{RMbessel, RMmodel, RFsimulate, RFFit}.

Examples

\begin{verbatim}
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))
\end{verbatim}

---

\texttt{RMdeclare} 

\textit{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

\texttt{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

```r
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 forula.
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)) +
```
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 \(h \in \mathbb{R}^d\) between two points in \(d\)-dimensional space and is given by

\[
C(h) = (C_0(h - s_i + s_j))_{i,j=0,\ldots,m}
\]

where \(s \in \mathbb{R}^{d \times m}\) and \(s_0 = 0\)

Usage

```r
RMdelay(phi, s, var, scale, Aniso, proj)
```

Arguments

- **phi**: a univariate stationary covariance model, that means an **RMmodel** whose `vdim` equals 1.
- **s**: a \(d \times m\)-dimensional shift matrix, where \(d\) is the dimension of the space, giving the components \(s = (s_1, \ldots, s_m)\) where the \(s_i\) are vectors.
- **var, scale, Aniso, proj**: optional arguments; same meaning for any **RMmodel**. If not passed, the above covariance function remains unmodified.

Details

Here, a multivariate random field is obtained from 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

See Also
RMmodel, RFsimulate, RFFit.

Examples

```r
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

\[
C_{11}(x, y) = \varphi(\| x - y \|) \\
C_{ij}(x, y) = -C_{1j}(x, y) = \partial \varphi(\| x - y \|)/\partial x \\
C_{i,j}(x, y) = \partial^2 \varphi(\| x - y \|)/\partial x \partial y
\]

for \( i, j = 2, \ldots, d \) 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,

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


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))
**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**

```r
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](http://ms.math.uni-mannheim.de)

**References**

Special models for rotation like fields

**Description**

RMexxa and RMetaxxa define the auxiliary functions

\[ f(h) = h^\top AA^\top h + \text{diag}(E) \]

and

\[ f(h) = h^\top ARRA^\top h + \text{diag}(E) \]

respectively.

**Usage**

RMexxa(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

**RMexxxa** is defined in space and returns an m-variate model.

**RMetaxxxa** is a space-time model with two spatial dimensions. The matrix R is a rotation matrix with angle $\beta t$ where $t$ is the time component.

Value

**RMexxxa** and **RMetaxxxa** return an object of class **RMmodel**.

Author(s)

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

References


See Also

**RMmodel**, **S10**

Examples

```r
# 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

```r
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 $\alpha$ and a parameter $\beta$ 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 **RM1gd**):

Here, we have

$$D = d + 1 - \alpha / 2, \alpha \in (0, 2]$$

and

$$H = 1 - \beta / 2, \beta > 0.$$  

I. e. the smaller $\beta$, 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**


**See Also**

**RMcauchy**, **RMcauchytbm**, **RMmodel**, **Rfsimulate**, **Rffit**.
**Examples**

```r
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))
```

---

### 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

```r
RMexp(var, scale, Aniso, proj)
```

### Arguments

- `var`, `scale`, `Aniso`, `proj`

  optional arguments; same meaning for any `RModel`. 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 `RModel`.

### Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, [http://ms.math.uni-mannheim.de](http://ms.math.uni-mannheim.de)
RMexponential

References

Covariance model


Tail correlation function


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

$$C(h) = \exp(\phi(h)) - \sum_{k=0}^{n} \frac{\phi^k(h)}{k!}/\exp(\phi(0)) - \sum_{k=0}^{n} \frac{\phi^k(0)}{k!}$$

if $\phi$ is a covariance model, and as

$$C(h) = \exp(-\phi(h))$$

if $\phi$ is a variogram model.

Usage

RMexponential(phi, n, standardised, var, scale, Aniso, proj)
Arguments

phi         a valid \texttt{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 \texttt{TRUE} then the above formula holds. If \texttt{FALSE} then only the nominator
of the above formula is returned. Default value is \texttt{TRUE}.
var, scale, Aniso, proj         optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above
covariance function remains unmodified.

Details

If $\gamma$ is a variogram, then $\exp(-\gamma)$ is a valid covariance.

Value

\texttt{RMexponential} returns an object of class \texttt{RMmodel}.

Author(s)

Martin Schlather, \texttt{<schlather@math.uni-mannheim.de>}, \url{http://ms.math.uni-mannheim.de}

References

See, for instance,


See Also

\texttt{RMmodel}, \texttt{RFsimulate}, \texttt{RFfit}.

Examples

\begin{verbatim}
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)
\end{verbatim}
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

  Chichester: John Wiley & Sons.
  Graph. Statist. 11, 587–599.
See Also

RMgenfbm, RMmodel, RFsimulate, RFFit.

Examples

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

model <- RFfbm(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**

```r
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


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)
Description

Expressions of the form $X@\text{rmfixed}(\beta)$ can be used within a formula of the type

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

that specifies the Linear Mixed Model.

Important remark: \texttt{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 $\beta$ is a vector of coefficients.

Note that a fixed effect of the form $X$ is interpreted as $X@\text{rmfixed}(\beta)\#\text{NA}$ by default (and $\beta$ is estimated provided that the formula is used in \texttt{RFFit}). Note that the 1 in an expression 1@\text{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

\texttt{RMmodel,RFformula,RFsimulate}.

Examples

```r
## For examples see the help page of 'RFformula'. ##
```

Description

\texttt{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 \texttt{RMgenfbm}.  

\texttt{rmflatpower} \textit{Variogram Model Similar to Fractal Brownian Motion}
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 $\alpha$ 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](http://ms.math.uni-mannheim.de)

References


See Also

`RMgenfbm`, `RMmodel`, `RFSimulate`, `RFFit`.

Examples

```r
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))
```
RMfractdiff

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 $r \in \mathbb{N}$ by

$$C(r) = (-1)^r \frac{\Gamma(1 - a/2)^2}{\Gamma(1 - a/2 + r)\Gamma(1 - a/2 - r)} r \in \mathbb{N}$$

and otherwise linearly interpolated. Here, $a \in [-1, 1)$, $\Gamma$ 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`: \( 0 < \alpha \leq 2 \)
- `var`, `scale`, `Aniso`, `proj`: optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

**Details**

The model is only valid for dimension \( d = 1 \). It is the covariance function for the fractional Gaussian noise with self-affinity index (Hurst parameter) \( H = \alpha/2 \) with \( 0 < \alpha \leq 2 \).

**Value**

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

**Author(s)**

Martin Schlather, <schlather@math.uni-mannheim.de>, [http://ms.math.uni-mannheim.de](http://ms.math.uni-mannheim.de)
References


See Also

`R'Mmodel, R'Fsimulate, R'Fit`.

Examples

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

model <- R'Mfractgauss(alpha=0.5, scale=0.2)
x <- seq(0, 10, 0.02)
plot(model)
plot(R'Fsimulate(model, x=x))
```

---

**R'Mgauss**

Gaussian Covariance Model

Description

`R'Mgauss` 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

`R'Mgauss(var, scale, Aniso, proj)`

Arguments

- `var`, `scale`, `Aniso`, `proj`
  
  optional arguments; same meaning for any `R'Mmodel`. 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 unstable procedures (cf. Stein, M. L. (1999), p. 29).

The Gaussian model is included in the symmetric stable class (see `R'Mstable`) 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](http://ms.math.uni-mannheim.de)

**References**


**See Also**

`RMstable` and `RMmatern` for generalizations;

`RMmodel`, `RFsimulate`, `RFFit`.

Do not mix up with `RPGauss` or `RRgauss`.

**Examples**

```r
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))
```

**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 $\alpha$ and a parameter $\beta$ 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 RM1gd): Here, we have

$$D = d + 1 - \alpha/2, \alpha \in (0, 2]$$

and

$$H = 1 - \beta/2, \beta > 0.$$  

I. e. the smaller $\beta$, 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


Tail correlation function (for $\alpha \in (0, 1]$)

**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 \( (\gamma + 1)^{\delta/\alpha} - 1 \) on variograms \( \gamma \) for \( \delta \in (0, 1] \). This original modification allows for further generalization, cf. `RMbcw`.

**Value**

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

**Examples**

```r
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))
```
Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>, http://ms.math.uni-mannheim.de

References

See Also
RMbcw, RMfbm, 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))

RMgenfbm

Gneiting-Wendland Covariance Models

Description
RMgenfbm is a stationary isotropic covariance model family whose elements are specified
by the two parameters $\kappa$ and $\mu$ 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 RMgenfbm.
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


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:
RMgenssst <- RMgenssst(kappa=3, mu=1.5, scale = 1 / 0.301187465825)
plot(RMgenssst(), model2=model2, type=c("p", "l"), pch=20)

\begin{verbatim}
RMgenssst     Non-Separable Space-Time model
\end{verbatim}

\section*{Description}
\textbf{RMgenssst} is a univariate stationary space-isotropic covariance model on $\mathbb{R}^d$ whose corresponding covariance is given by

$$C(h,u) = \phi(\sqrt{\psi^{-1}(u)h})/\sqrt{\text{det}(\psi)}$$

\section*{Usage}
\texttt{RMgenssst(phi, psi, dim_u, var, scale, Aniso, proj)}

\section*{Arguments}
\begin{itemize}
  \item \texttt{phi} is a normal mixture \texttt{RMmodel}, cf. \texttt{RFgetModelNames(monotone="normal mixture")}.
  \item \texttt{psi} is a $d$-variate variogram model \texttt{RMmodel}.
  \item \texttt{dim_u} the dimension of the component $u$
  \item \texttt{var, scale, Aniso, proj} optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.
\end{itemize}

\section*{Details}
This model is used for space-time modelling where the spatial component is isotropic.

\section*{Value}
\texttt{RMgenssst} returns an object of class \texttt{RMmodel}.

\section*{Author(s)}
Martin Schlather, <schlather@math.uni-mannheim.de>, \url{http://ms.math.uni-mannheim.de}

\section*{References}
\begin{itemize}
\end{itemize}

\section*{See Also}
\texttt{RMnsst, RMmodel, RFSimulate, RFFit}. 
RMgneiting

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 Rmodel. 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 \( \kappa = 3 \), \( \mu = 1 \) 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 \( \mu = 1.5 \): \( s = \frac{10\sqrt{2}}{4\pi} \).
Value

`rmgneitingdiff` 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


For the version (orig=FALSE)

- this package *RandomFields*.

See Also

`RMbigneiting`, `RMgengneiting`, `RMgauss`, `RMmodel`, `RFSimulate`, `RFFit`.

Examples

```r
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 \( W_\nu \) 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

  Soc Part A 125, 2449-2464.

See Also

RMbigneiting, RMgneiting, RMcengneiting, 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)
p\text{plot(model)}
p\text{plot(RFSimulate(model, x=x))}
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

\[
C(r) = \frac{(\delta^2 + r^2)^{\nu/2} K_\nu(\xi(\delta^2 + r^2)^{1/2})}{\delta^n K_\nu(\xi)}
\]

where \( K_\nu \) denotes the modified Bessel function of second kind.

Usage

\[
\text{RMhyperbolic}(\nu, \lambda, \delta, \text{var, scale, Aniso, proj})
\]

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 \( \nu < 0 \).

- \( \text{var, scale, Aniso, proj} \)
  - optional arguments; same meaning for any \text{RModel}. 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 \( \lambda, \delta \) 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 \text{RMcauchy}) with \( \gamma = -\frac{\nu}{2} \) and scale=\( \delta \); the choice \( \delta = 0 \) yields a covariance model of type \text{RMwhittle} with smoothness parameter \( \nu \) and scale parameter \( \lambda^{-1} \).

Value

\text{RMhyperbolic} returns an object of class \text{RModel}.

Author(s)

Martin Schlather, <schlather@math.uni-mannheim.de>, \text{http://ms.math.uni-mannheim.de}
References


See Also

`rmcauchy`, `rmwhittle`, `rmodel`, `rfsimulate`, `rffit`.

Examples

```r
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))
```

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 `RModel`. If not passed, the above covariance function remains unmodified.

Value

`RMiaco` returns an object of class `RModel`. 

*iaco-Cesare model*
Author(s)

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

References


See Also

- `rmmodel`

Examples

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

model <- RMiaoco(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`
**RMidmodel**

### Examples

```r
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)
```

### Description

RMidmodel is the identical operator on models, i.e. for objects of class `RMmodel`.

### Usage

```r
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](http://ms.math.uni-mannheim.de)

### See Also

`RMmodel`, `RMid`, `RMtrafo`, `RMprod`

### Examples

```r
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__
- **ROC**: model for transforming coordinates within the cartesian system
- **ROE**: model for transforming earth coordinates to cartesian coordinates
- **ROMissing**: for error messages only
- **RMMixed**: internal representation of a mixed model
- **RMselect**: will be obsolete in future
- **RMsetparam, RMptsGivenShape, RMstandardShape, RMstatShape**: for max-stable processes and Poisson processes: models that combine shape functions with corresponding point processes
- **RP__name__Intern**: internal representations of some processes
- **RPS, RPplusp, etc.**: specific processes for RMS and RMplus etc. (For those covariance models that have specific simulation processes programmed.)
- **RMS**: internal representation of the modifying arguments var, scale, Aniso, proj

**Author(s)**

Martin Schlather, <schlather@math.uni-mannheim.de>, [http://ms.math.uni-mannheim.de](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)
\textbf{RMinexp} \hspace{1cm} \textit{Integral exponential operator}

\textbf{Description}

\texttt{RMinexp} is a univariate stationary covariance model depending on a univariate variogram model $\phi$. The corresponding covariance function only depends on the difference $h$ between two points and is given by

$$C(h) = \frac{1 - \exp(-\phi(h))}{\phi(h)}$$

\textbf{Usage}

\texttt{RMinexp(phi, var, scale, Aniso, proj)}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{phi} \hspace{1cm} a variogram \texttt{RMmodel}
  \item \texttt{var, scale, Aniso, proj} \hspace{1cm} optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.
\end{itemize}

\textbf{Value}

\texttt{RMinexp} returns an object of class \texttt{RMmodel}.

\textbf{Author(s)}

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

\textbf{References}

\begin{itemize}
\end{itemize}

\textbf{See Also}

\texttt{RMmodel, RFsimulate, RFfit}.

\textbf{Examples}

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

model <- RMinexp(RMfbm(alpha=1.5, scale=0.2))
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
\end{verbatim}
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

\[
C(r) = a_0 + a_2 r^2 + \phi(r), \quad 0 \leq r \leq \text{diameter}
\]

\[
C(r) = b_0 (\text{rawR} - r)^3 / r, \quad \text{diameter} \leq r \leq \text{rawR} \times \text{diameter}
\]

\[
C(r) = 0, \quad \text{rawR} \times \text{diameter} \leq r
\]

Usage

RMintrinsic(phi, diameter, rawR, var, scale, Aniso, proj)

Arguments

phi  
an RModel; 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 RModel. 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 \( \phi \) and the parameters.

For certain models \( \phi \), i.e. stable, whittle, gencauchy, and the variogram model fractalb some sufficient conditions are known.

Value

RMintrinsic returns an object of class RModel.
**Author(s)**

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

**References**


**See Also**

`RPintrinsic`, `RMmodel`, `RFSimulate`, `RFFit`.

**Examples**

```r
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)
plot(model)
plot(RFSimulate(model, x=x))
```

**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


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 \texttt{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 \texttt{Rmgengauchy}):

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

\texttt{Rmlgd} returns an object of class \texttt{Rmmodel}.

Author(s)

Martin Schlather, \texttt{<schlather@math.uni-mannheim.de>}, \url{http://ms.math.uni-mannheim.de}

References


See Also

\texttt{Rmmodel}, \texttt{RFSimulate}, \texttt{RFfit}.

Examples

\begin{verbatim}
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))
\end{verbatim}
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](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, _RMsfbm, RMmodel, RFsimulate, RFfit_.

**RMma**

**Examples**

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

model <- RMlsfbm(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

\[ C(h) = \left( \frac{\theta}{1 - \theta \phi(h)} \right)^\alpha \]

**Usage**

```r
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](http://ms.math.uni-mannheim.de)

**References**

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))

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 \( \phi \) 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 \( \phi \) is a covariance model.
Here \( \Gamma \) 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; \( \delta \) 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 $\phi$. (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


See Also

Rmwhittle, Rmmodel, RFsimulate, RFfit.

Examples

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

Matrix operator

Description

RMMmatrix is a multivariate covariance model depending on one multivariate covariance model, or one or several univariate covariance models \( C_0, \ldots \). 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) \( C_0, C_1, \text{ etc.} \) and the offdiagonals are all zero.

Usage

RMMmatrix(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, M, vdim, var, scale, Aniso, proj)

Arguments

\( C_0 \)  
a k-variate covariance RModel or a univariate model or a list of models joined by \( c \)

\( C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9 \)  
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

\[
C(x, y) = M(x)\phi(x, y)M(y)^t
\]

\( \text{vdim} \)  
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, \( \text{vdim} \) must equal the number of columns of \( M \))

\( \text{var, scale, Aniso, proj} \)  
optional arguments; same meaning for any RModel. If not passed, the above covariance function remains unmodified.

Value

RMMmatrix returns an object of class RModel.

Note

- RMMmatrix also allows variogram models are arguments.
RMmatrix

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: bivivariate Linear Model of Coregionalisation
x <- y <- seq(0, 10, 0.2)

t1 <- RMmatrix(M = c(0.9, 0.43), RMwhittle(nu = 0.3)) +
     RMmatrix(M = c(0.6, 0.8), RMwhittle(nu = 2))
plot(t1)
simu1 <- RFSimulate(RFdirect(t1), 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(RFdirect(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(RFdirect(model3), x, y)
stopifnot(all.equal(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**

*Covariance and Variogram Models in RandomFields (RM commands)*

**Description**

Summary of implemented covariance and variogram models

**Details**

To generate a covariance or variogram model for use within RandomFields, calls of the form

\[
RM_{name}(..., var, scale, Aniso, proj)
\]

can be used, where _name_ has to be replaced by a valid model name.

- ... can take model specific arguments.
- var is the optional variance argument \(v\),
- scale the optional scale argument \(s\),
- Aniso an optional anisotropy matrix \(A\) or given by `RMangle`, and
- proj is the optional projection.

With \(\phi\) denoting the original model, the transformed model is \(C(h) = v \ast \phi(A \ast h/s)\). See `RM` 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)

\texttt{RMfbm} fractal Brownian motion

Basic Operations

\texttt{RMMult,*} product of covariance models
\texttt{RMPlus,+} sum of covariance models or variograms

Others

\texttt{RMtrend} trend
\texttt{RMangle} defines a 2x2 anisotropy matrix by rotation and stretch arguments.

Author(s)

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

References


See Also

\texttt{RM} for an overview over more advanced classes of models
\texttt{RC, RF, RP, RR, R, RFcov, RFformula, RMmodelsAdvanced, RMmodelsAuxiliary, trend modelling}

Examples

\texttt{RFoptions(seed=0)} ## *ANY* simulation will have the random seed 0; set
\texttt{## 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 RFsp or RFempVario or RFFit or RMmodel; in the latter case, x can be any sophisticated model but it must be either stationary or a variogram model.

y  
ignored in most methods

MARGIN  
vector of two; two integer values giving the coordinate dimensions w.r.t. whether the field or the covariance model is to be plotted; in all other directions, the first index is taken.

fixed.MARGIN  
only for class(x)==CLASS_CLIST and if dim > 2; a vector of length dim-2 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 CLASS_CLIST, see Details.

dim  
must equal 1 or 2; only for class(x)==CLASS_CLIST; the covariance function and the variogram are plotted as a function of R^{dim}. 
n.points integer; only for `class(x) == CLASS_CLIST`; the number of points at which the model is evaluated (in each dimension); defaults to 200.

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

plotmethod string or function. Internal.

type character. See `points`.

zlab 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 substract 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

- `modelLformel`: 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](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 Gaussian 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 zerospacesisotropic 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 `RMmodels`.
'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](http://ms.math.uni-mannheim.de)

**References**


**See Also**

`RMmodel`, `RFgetModelNames`

**Examples**

```r
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.

<table>
<thead>
<tr>
<th>RMmodels</th>
<th>general introduction and a collection of simple models</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMmodelsAdvanced</td>
<td>includes more advanced stationary and isotropic models, variogram models, non-stationary models, hierarchical models</td>
</tr>
<tr>
<td>Bayesian</td>
<td>multivariate covariance models and multivariate trend models</td>
</tr>
<tr>
<td>RMmodelsMultivariate</td>
<td>non-stationary covariance models</td>
</tr>
<tr>
<td>RMmodelsNonstationary</td>
<td>space-time covariance models</td>
</tr>
<tr>
<td>RMmodelsSpaceTime</td>
<td>models based on the polar coordinate system, usually used in earth models</td>
</tr>
<tr>
<td>Spherical models</td>
<td>models related to max-stable random fields</td>
</tr>
<tr>
<td>Tail correlation functions</td>
<td>how to pass trend specifications</td>
</tr>
<tr>
<td>trend modelling</td>
<td>simple mathematical functions that are typically used to build non-stationary covariance models and arbitrary trends</td>
</tr>
<tr>
<td>Mathematical functions</td>
<td>rather specialized models, most of them not having positive definiteness property, but used internally</td>
</tr>
<tr>
<td>RMmodelsAuxiliary</td>
<td></td>
</tr>
</tbody>
</table>

**Author(s)**

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

**See Also**

RC, RR, RF, R.

**Examples**

```r
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

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMaskey</td>
<td>Askey model (generalized test or triangle model)</td>
</tr>
<tr>
<td>RMBcw</td>
<td>bridging model between RMcauchy and RMgenfbm</td>
</tr>
<tr>
<td>RMBessel</td>
<td>Bessel family</td>
</tr>
<tr>
<td>RMCircular</td>
<td>circular model</td>
</tr>
<tr>
<td>RMCnstant</td>
<td>spatially constant model</td>
</tr>
<tr>
<td>RMcubic</td>
<td>cubic model (see Chiles and Delfiner)</td>
</tr>
<tr>
<td>RMDagum</td>
<td>Dagum model</td>
</tr>
<tr>
<td>RMDampedcos</td>
<td>exponentially damped cosine</td>
</tr>
<tr>
<td>RMQexp</td>
<td>variant of the exponential model</td>
</tr>
<tr>
<td>RMfracdiff</td>
<td>fractionally differenced process</td>
</tr>
<tr>
<td>RMfracgauss</td>
<td>fractional Gaussian noise</td>
</tr>
<tr>
<td>RMgengneiting</td>
<td>generalized Gneiting model</td>
</tr>
<tr>
<td>RMrgeitingdiff</td>
<td>Gneiting model for tapering</td>
</tr>
<tr>
<td>RMLgd</td>
<td>Gneiting’s local-global distinguisher</td>
</tr>
<tr>
<td>RMLsfbm</td>
<td>locally stationary fractal Brownian motion</td>
</tr>
<tr>
<td>RMpenta</td>
<td>penta model (see Chiles and Delfiner)</td>
</tr>
<tr>
<td>RMPower</td>
<td>Golubov’s model</td>
</tr>
<tr>
<td>RMwave</td>
<td>cardinal sine</td>
</tr>
</tbody>
</table>

Variogram models (stationary increments/intrinsically stationary)

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMBcw</td>
<td>bridging model between RMcauchy and RMgenfbm</td>
</tr>
<tr>
<td>RMdeWijssian</td>
<td>generalized version of the DeWijssian model</td>
</tr>
<tr>
<td>RMgenfbm</td>
<td>generalized fractal Brownian motion</td>
</tr>
<tr>
<td>RMflatpower</td>
<td>similar to fractal Brownian motion but always smooth at the origin</td>
</tr>
</tbody>
</table>

General composed models (operators)

Here, composed models are given that can be of any kind (stationary/non-stationary), depending on the submodel.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMBernoulli</td>
<td>Correlation function of a binary field based on a Gaussian field</td>
</tr>
<tr>
<td>RMAxpontial</td>
<td>exponential of a covariance model</td>
</tr>
</tbody>
</table>
RMintexp integrated exponential of a covariance model (INCLUDES ma2)
RMpower powered variograms
RMqam Porcu’s quasi-arithmetic-mean model
RM details on the optional transformation arguments (var, scale, Aniso, proj)

Stationary and isotropic composed models (operators)
- RMcutoff Gneiting’s modification towards finite range
- RMintrinsic Stein’s modification towards finite range
- RMnatsc practical range
- RMstein Stein’s modification towards finite range
- Rmtbm Turning bands operator

Stationary space-time models
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 \texttt{RMsangle} or any other \texttt{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 \texttt{PrintModelList()}). Here, the argument var, scale, Aniso and proj must be passed by the model \texttt{RMs}. For instance,

  \begin{verbatim}
  model <- RMeXP(scale=2, var=5)
  is equivalent to
  model <- list("RMS", scale=2, var=5, list("RMeXP"))
\end{verbatim}

  The latter definition can be also obtained by

  \begin{verbatim}
  print(RMeXP(scale=2, var=5))
\end{verbatim}

  \begin{verbatim}
  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))).
\end{verbatim}

All models have secondary names that stem from \texttt{RandomFields} versions 2 and earlier and that can also be used as strings in the list notation. See \texttt{RFgetModelNames(internal=FALSE)} for the full list.

Author(s)

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

References


**RMmodelsMultivariate**

**See Also**

- `rfformula`, `RM`, `RMmodels`, `RMmodelsAuxiliary`.

**Examples**

```r
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 differentiabilities
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)
```

---

**Description**

Here, multivariate and vector-valued covariance models are presented.

**Details**

**Bivariate covariance models**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Rmbicauchy</code></td>
<td>a bivariate Cauchy model</td>
</tr>
<tr>
<td><code>Rmbiwm</code></td>
<td>full bivariate Whittle-Matern model (stationary and isotropic)</td>
</tr>
<tr>
<td><code>Rmbigneiting</code></td>
<td>bivariate Gneiting model (stationary and isotropic)</td>
</tr>
<tr>
<td><code>Rmbistable</code></td>
<td>a bivariate stable model</td>
</tr>
</tbody>
</table>

**Physically motivated, vector valued covariance and variogram models**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RMcurlfree</code></td>
<td>curlfree (spatial) vector-valued field (stationary and anisotropic)</td>
</tr>
<tr>
<td><code>RMdivfree</code></td>
<td>divergence free (spatial) vector-valued field (stationary and anisotropic)</td>
</tr>
<tr>
<td><code>RMkolmogorov</code></td>
<td>Kolmogorov’s model of turbulence</td>
</tr>
<tr>
<td><code>RMvector</code></td>
<td>vector-valued field (combining <code>RMcurlfree</code> and <code>RMdivfree</code>)</td>
</tr>
</tbody>
</table>

**Multivariate covariance models**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RMdelay</code></td>
<td>delay effect model (stationary)</td>
</tr>
<tr>
<td><code>RMderiv</code></td>
<td>field and its gradient</td>
</tr>
<tr>
<td><code>RMmatrix</code></td>
<td>linear model of coregionalization</td>
</tr>
<tr>
<td><code>Rmparswm</code></td>
<td>multivariate Whittle-Matern model (stationary and isotropic)</td>
</tr>
</tbody>
</table>
Operators

**RMcov** covariance structure given by a multivariate variogram
**RMexponential** functional returning $e^C$
**RMmatrix** linear model of coregionalization
**RMmqam** multivariate quasi-arithmetic mean (stationary)
**RMschur** element-wise product with a positive definite matrix
**RMtbm** turning bands operator

Trend models

**RMtrend** for explicit trend modelling
**R.models** for implicit trend modelling
**R.c** binding univariate trend models into a vector

Author(s)

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

References


See Also

**Rfformula, RMmodels, RM, RMmodelsAdvanced**

‘multivariate’, a vignette for multivariate geostatistics

Examples

```r
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)
```
**Description**

Here, non-stationary covariance models are presented.

**Details**

**Covariance models**

- `RMnonstwm`: one of Stein's non-stationary Whittle-Matern models
- `RMprod`: scalar product
- `aniso`: for space transformation, see the example in `R.models`
- `scale`: cf. `RMS`, can be any non-negative function for any scale mixture model, such as the whittle-matern-classes, the powered exponential family, and the `rmgencauchy` model.

**Trend models** See `RMmodelsTrend`.

**Author(s)**

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

### 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 `+`, `*`, and `proj`, see also the example below
- `RMave` space-time moving average model
- `RMcoxisham` Cox-Isham model
- `RMcurlfree` curlfree (spatial) field (stationary and anisotropic)
- `RMdivfree` divergence free (spatial) vector-valued field (stationary and anisotropic)
- `RMgennsst` generalization of Gneiting’s non-separable space-time model
- `RMiamo` non-separable space-time model
- `RMmastein` Ma-Stein model
- `RMnsst` Gneiting’s non-separable space-time model
- `RMstein` Stein’s non-separable space-time model
- `RMstp` Single temporal process
- `RMtbnm` Turning bands operator

### Author(s)

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

### References

See Also

`RFformula`, `RM`, `RMmodels`, `RMmodelsAdvanced`.

Examples

```r
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 $C_0, C_1, ... , C_9$. Used together with `RPsmithe`, it allows for mixed moving maxima with a finite number of shape functions.

**Usage**

```r
RMmppplus(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, p)
```

**Arguments**

- `C0`  
  an `RMmodel`
- `C1`, `C2`, `C3`, `C4`, `C5`, `C6`, `C7`, `C8`, `C9`  
  optional; each an `RMmodel`
- `p`  
  vector of probabilities for the shape functions. The probabilities should add up to 1. The length of the vector equals the number of given submodels.
Value

`RMmqamplus` 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

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

Description

`RMmqam` is a multivariate stationary covariance model depending on a submodel \( \phi \) 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 \( \phi \) 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 \texttt{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 \(\phi\) is a valid covariance function for all dimensions, e.g. \texttt{RMstable}, \texttt{RMgauss}, \texttt{RMexponential}.

Warning: RandomFields cannot check whether the combination of \(\phi\) and \(C_i\) is valid.

Value

\texttt{RMMqam} returns an object of class \texttt{RMmodel}.

Author(s)

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

References


See Also

\texttt{RMqam}, \texttt{RMmodel}, \texttt{RFSimulate}, \texttt{RFfit}.

Examples

\begin{verbatim}
RFOptions(seed=0) ## *ANY* simulation will have the random seed 0; set
## RFOptions(seed=NA) to make them all random again

RFOptions(modus Operandi="slppy")
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")
\end{verbatim}
**Description**

`RMmult` is a multivariate covariance model which depends on up to 10 submodels \( C_0, C_1, \ldots, C_9 \). 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. \( C_0 \times C_1 \).

The global arguments `scale, Aniso, proj` of `RMmult` are multiplied to the corresponding argument of the submodels (from the right side). E.g.,

\[
\text{RMmult(Aniso=A1, RMexp(Aniso=A2), RSpheric(Aniso=A3))}
\]

equals

\[
\text{RMexp(Aniso=A2 ** A1) * RSpheric(Aniso=A3 ** A1)}
\]

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](http://ms.math.uni-mannheim.de)

**See Also**

`RMplus, RMmodel, RMprod, RFSimulate, RFFit`. 
**Examples**

```r
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)
```

---

**Description**

**RMmultiquad** is an isotropic covariance model. The corresponding covariance function, the multiquadric family, only depends on the angle \( \theta \in [0, \pi] \) between two points on the sphere and is given by

\[
\psi(\theta) = (1 - \delta)^{2\tau} / \left(1 + \delta^2 - 2 \cdot \delta \cdot \cos(\theta)\right)^{\tau}
\]

where \( \delta \in (0, 1) \) and \( \tau > 0 \).

**Usage**

```r
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 `RModel`. 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'

\[
\psi(\theta) = (1 - \delta) / \sqrt{1 + \delta^2 - 2 \cdot \delta \cdot \cos(\theta)}
\]

and for fixed parameter \( \tau = 1.5 \) which gives the covariance function called 'Poisson spline'

\[
\psi(\theta) = (1 - \delta)^3 / (1 + \delta^2 - 2 \cdot \delta \cdot \cos(\theta))^{1.5}
\]

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


See Also

`RMmodel`, `RFSimulate`, `RFfit`, `RMchoquet`, `spherical models`

Examples

```r
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")
```
**RMnatsc**

**Description**

RMnatsc is a stationary isotropic covariance model that depends on a stationary isotropic covariance model $\phi$. 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)
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 $\nu$ must be a positive function.

$W_{\mu}$ is the covariance of the \texttt{RMwhittle} model or the \texttt{RMMatern} model.

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.

It cannot be checked whether $\nu$ only takes positive values. So the responsibility is completely left to the user.

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


\texttt{RMwhittle}, \texttt{RMMmodel}, \texttt{RFsimulate}, \texttt{RFfit}.

x <- seq(-1.2, 1.2, len=50)
model <- RMwhittle(nu=RMattnorm(2))
z <- RFsimulate(model=model, x, x, n=4)
plot(z)
RMnsst

Non-Separable Space-Time model

Description

RMnsst is a univariate stationary space-isotropic 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

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 <- RMsst(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))

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 $1,\ldots,vdim$ 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 de-

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.
Rmparswm

**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 \( W_\nu \) is the covariance of the Rmwhittle model.

Rmparswm is defined as

\[
\rho_{ij} C_{ij}(r)
\]

where \( \rho_{ij} \) 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 \((\nu_{11}, \nu_{22}, \ldots)\). The offdiagonal elements \(\nu_{ij}\) are calculated as \(0.5(\nu_{ii} + \nu_{jj})\).

rho any positive definite \(m \times m\) matrix; here, \(m\) equals \(\text{length}(\text{nudiag})\). For the calculation of \(c_{ij}\) see Details.

var, scale, Aniso, proj optional arguments; same meaning for any \textbf{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 \(\Gamma\) is the Gamma function and \(d\) is the dimension of the space.

Note that the definition of \textbf{RMParswm} is \textbf{RMschur}(M=rho, \textbf{RMParswm}(\text{nudiag, var, scale, Aniso, proj})

Value

\textbf{RMParswm} returns an object of class \textbf{RMmodel}.

Author(s)

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

References


See Also

\textbf{RMbiwm, RMwhittle, RMmodel, RFSimulate, RFFit}.

Examples

```r
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 <- RMParswm(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)
```
Description

\texttt{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

\[
C(r) = \left( 1 - \frac{22}{3} r^2 + 33 r^4 - \frac{77}{2} r^5 + \frac{33}{2} r^7 - \frac{11}{2} r^9 + \frac{5}{6} r^{11} \right) 1_{[0,1]}(r).
\]

Usage

\texttt{rmpenta(var, scale, Aniso, proj)}

Arguments

\texttt{var, scale, Aniso, proj}

optional arguments; same meaning for any \texttt{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

\texttt{rmpenta} returns an object of class \texttt{RMmodel}.

Author(s)

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

References


See Also

\texttt{RMmodel, RFsimulate, RFfit}.
Examples

```r
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 \( C_0, C_1, ..., C_9 \). 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

```r
RMplus(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, var, scale, Aniso, proj)
```

Arguments

```r
C0               an **RMmodel**.
C1,C2,C3,C4,C5,C6,C7,C8,C9
  optional; each an **RMmodel**.
var,scale,Aniso,proj
  optional arguments; same meaning for any **RMmodel**. If not passed, the above model remains unmodified.
```

Details

**RMmodels** can also be summed up via the +-operator, e.g. \( C_0 + C_1 \).

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](http://ms.math.uni-mannheim.de)
See Also

Rmmult, Rmmodel, RFsum, RFsimulate, RFfit.

Examples

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

model <- Rmplus(RMgauss(), RMnugget(var=0.1))
model2 <- RMgauss() + RMnugget(var=0.1)
plot(model, "model.+"=model2, type="p", "1", pch=20, xlim=c(0,3)) # the same

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)

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

References

Poisson polygons / Poisson hyperplane tessellation


Poisson storm process


Mixed Poisson storm process

See Also

*RMball, RMspheric, RFsimulate, RMmodel.*

Examples

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

Examples

```r
## 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 \( \gamma \) of the model is given by

\[
\gamma = \phi^\alpha
\]

if \( \phi \) is a variogram model. The covariance \( C \) of the model is given by

\[
C(h) = \phi(0) - (\phi(0) - \phi(h))^\alpha
\]

if \( \phi \) 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 \( \gamma \) is a variogram, then \( \gamma^\alpha \) is a valid variogram for \( \alpha \) 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](http://ms.math.uni-mannheim.de)
References


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 \( \phi \) with values in an arbitrary Hilbert space. However, as already mentioned above, \( \phi \) 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](http://ms.math.uni-mannheim.de)

**References**


**See Also**

`RMid`, `RMid`, `RMsun`, `RMmodel`, `RMmult`.

**Examples**

```r
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 $\phi$ such that $\psi(\cdot) := \phi(\sqrt{\cdot})$ is completely monotone, and depending on further stationary covariance models $C_i$. The covariance is given by

$$
C(h) = \phi(\sqrt{\sum_i \theta_i(\phi^{-1}(C_i(h)))^2})
$$

Usage

RMqam(phi, C1, C2, C3, C4, C5, C6, C7, C8, C9, theta, var, scale, Aniso, proj)

Arguments

phi a valid covariance \texttt{RMmodel} that is a normal scale mixture. See, for instance, \texttt{RFgetModelNames} (monotone="normal mixture").

C1, C2, C3, C4, C5, C6, C7, C8, C9 optional further univariate stationary \texttt{RMmodels}

theta a vector with positive entries

var, scale, Aniso, proj optional arguments; same meaning for any \texttt{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 \( \phi \) is a valid covariance function for all dimensions, e.g. \texttt{RMstable}, \texttt{RMgauss}, \texttt{RMexponential}.

Warning: \texttt{RandomFields} cannot check whether the combination of \( \phi \) and \( C_i \) is valid.

Value

\texttt{RMqam} returns an object of class \texttt{RMmodel}.

Author(s)

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

References


See Also

\texttt{RMmqam}, \texttt{RMmodel}, \texttt{RFSimulate}, \texttt{RFFit}.

Examples

\begin{verbatim}
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))
\end{verbatim}
**Description**

The covariance function is

\[ C(x) = \frac{2e^{-x} - \alpha e^{-2x}}{2 - \alpha} \]

**Usage**

\[ \text{RMqexp}(\alpha, \text{var}, \text{scale}, \text{Aniso}, \text{proj}) \]

**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](http://ms.math.uni-mannheim.de)

**References**

- ?

**See Also**

`RMmodel`.

**Examples**

```r
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))
```
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

\texttt{RMrational(A, a)}

Arguments

\begin{itemize}
  \item \texttt{A} \quad a d \times d \text{ matrix}
  \item \texttt{a} \quad a \text{ vector of one or two components; the second component has default value zero.}
\end{itemize}

Value

\texttt{RMrational} returns an object of class \texttt{RMmodel}.

Author(s)

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

See Also

\texttt{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 $\phi$ 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**

- **phi**: submodel
- **var**: is the optional variance parameter $v$.
- **scale**: scaling parameter $s$ which is positive.
- **Aniso**: matrix or RMmodel. The optional anisotropy matrix $A$, multiplied from the right by a distance vector $x$, i.e. $Ax$.
- **proj**: is the optional projection vector which defines a diagonal matrix of zeros and ones and proj gives the positions of the ones (integer values between 1 and 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. $x^T B$.

**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 RMmodel.

Most users will never call RMS directly.

**Value**

RMS returns an object of class RMmodel.

**Note**

At most one of the arguments Aniso, anisoT and proj may be given at the same time.
RMSadvanced

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. \( x^T B \).
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


See Also

RMS, RMbend 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


See Also

RMAdvanced, RMBblend, RMBbubble

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))
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 \( \phi \) 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

```r
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

```r
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
```r
x <- seq(0, 10, 0.1)
z <- RFsimulate(RFschlather(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.
```r
z <- RFsimulate(RMschlather(model), x)
plot(z)
```

---

**RMschur**  
**Schur product**

### Description

The covariance function is

\[ C(x) = M \ast \phi(x) \]

where ‘\( \ast \)’ 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), 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](http://ms.math.uni-mannheim.de)

**See Also**

RMmodel, RR.

**Examples**

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

model <- RPoisson(RMsign(RMtent(), p=0.8))
x <- seq(0, 10, 0.02)
plot(RFsimulate(model, x=x))
```

---

### 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**

```r
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.
RMspheric

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


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 <- RMsinpower(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")

RMspheric

The Spherical Covariance Model

Description

RMspheric is a stationary isotropic covariance model which is only valid up to dimension 3. The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

\[
C(r) = \left(1 - \frac{3}{2}r + \frac{1}{2}r^3 \right) 1_{[0,1]}(r)
\]
Usage

RMspheic(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

RMspheic 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


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 <- RMspheic()
x <- seq(0, 10, 0.02)
plot(model)
plot(RFSimulate(model, x=x))
Description

\texttt{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

\begin{verbatim}
RMstable(alpha, var, scale, Aniso, proj)
RMpoweredexp(alpha, var, scale, Aniso, proj)
\end{verbatim}

Arguments

- \texttt{alpha} a numerical value; should be in the interval (0,2] to provide a valid covariance function for a random field of any dimension.
- \texttt{var, scale, Aniso, proj} optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.

Details

The parameter \( \alpha \) 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 \texttt{RMexp}) for \( \alpha = 1 \) and the Gaussian model (see \texttt{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

\texttt{RMstable} returns an object of class \texttt{RMmodel}.

Author(s)

Martin Schlather, \texttt{schlather@math.uni-mannheim.de}, \url{http://ms.math.uni-mannheim.de}
References

Covariance function


Tail correlation function (for $\alpha \in (0, 1]$)


See Also

*Rmbistable, RMexp, RMgauss, RMmodel, Rfsimulate, Rffit.*

Examples

```r
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

$$C(h, t) = W_{\nu}(y) - \langle h, z \rangle t / ((\nu - 1)(2\nu + d)) \ast W_{\nu-1}(y)$$

Here, $W_{\nu}$ is the covariance of the *RMwhittle* model with smoothness parameter $\nu$; $y = \|(h, t)\|$ 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 RModel. If not passed, the above
covariance function remains unmodified.

Details


Value

RMstein returns an object of class RModel.

Author(s)

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

References


See Also

RModel, 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))
**RMstp**

**Description**

**RMstp** is a univariate covariance model which depends on a normal mixture submodel $\phi$. The covariance is given by

$$C(x, y) = |S_x|^{1/4}|S_y|^{1/4}|A|^{-1/2}\phi(Q(x, y)^{1/2})$$

where

$$Q(x, y) = c^2 - m^2 + h^t(S_x + 2(m + c)M)A^{-1}(S_y + 2(m - c)M)h,$$

$$c = -z^t h + \xi_2(x) - \xi_2(y),$$

$$A = S_x + S_y + 4Mhh^tM$$

$$m = h^tMh$$

$$h = x - y$$

**Usage**

`RMstp(xi, phi, S, z, M, var, scale, Aniso, proj)`

**Arguments**

- **xi**: arbitrary univariate function on $R^d$
- **phi**: an **RModel** that is a normal mixture model, cf. `RFgetModelNames(monotone="normal mixture")`
- **S**: functions that returns strictly positive definite $d \times d$
- **z**: arbitrary vector, $z \in R^d$
- **M**: an arbitrary, symmetric $d \times d$ matrix
- **var, scale, Aniso, proj**: optional arguments; same meaning for any **RModel**. If not passed, the above covariance function remains unmodified.

**Details**


**Value**

**RMstp** returns an object of class **RModel**.

**Author(s)**

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

**References**

See Also

`RMmodel`, `RFsimulate`, `RFfit`.

Examples

```r
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=RMexxxa(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

---

RMtbm

Turning Bands Method

Description

RMtbm is a univariate or multivariate stationary isotropic covariance model in dimension reduceddim which depends on a univariate or multivariate stationary isotropic covariance \( \phi \) in a bigger dimension fulldim. For formulas for the covariance function see details.

Usage

RMtbm(phi, fulldim, reduceddim, layers, var, scale, Aniso, proj)

Arguments

phi, fulldim, reduceddim, layers
See RPtbm.
var, scale, Aniso, proj
optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The turning bands method stems from the 1:1 correspondence between the isotropic covariance functions of different dimensions. See Gneiting (1999) and Strokorb and Schlather (2014).

The standard case is \( \text{reduceddim}=1 \) and \( \text{fulldim}=3 \). 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

\texttt{RMtbm} returns an object of class \texttt{RMmodel}.

Author(s)

Martin Schlather, \texttt{schlather@math.uni-mannheim.de}, \url{http://ms.math.uni-mannheim.de}

References

Turning bands


Turning layers


See Also

\texttt{RPtbm,RFsimate}.
Examples

```r
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 <- RMspHCpheric()
plot(model, model.on.the.line=RMtbH(MRspHmeric()), 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**

```r
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 RFoptions()$general$units != "", 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 c(start, step, length).  
  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 zenith 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:

- en.wikipedia.org/wiki/Geographic_coordinate_system
- 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_CARTESIANCOORD), 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")
**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 \((Z_1(\cdot), \ldots, Z_p(\cdot))\); 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**

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
(model10 <- RMexp(var=0.4) + trend) # exponential covariance with mean -1
(model11 <- RMexp(var=0.4) + -1) # same as model10
(model12 <- RMexp(var=0.4) + RMtrend(-1)) # same as model10
(model13 <- RExp(var=0.4) - 1) # this is a purely deterministic model
       # with exponential trend
plot(RFsimulate(model=model10, x=x, y=x)) # exponential covariance
       # and mean -1
plot(RFsimulate(model=model11, x=x, y=x)) # dito
plot(RFsimulate(model=model12, x=x, y=x)) # dito
plot(RFsimulate(model=model13, 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**

- phi: function of class RMmodel
- radius: truncation at radius

**Value**

RMtruncsupport returns an object of class RMmodel.

**Author(s)**

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

**References**


**See Also**

RMmodel, RMmatrix, RPoisson.

**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 'RPoisson'.
```
**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**

```r
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:

  - '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 `quote(myfunction(x))` or as `quote(myfunction(x, y))`, 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

```r
e <- rfoptions(seed=0)  ## ANY* simulation will have the random seed 0; set
eoptions(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) +
```
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**

\[
\text{RMvector}(\phi, a, Dspace, \text{var}, \text{scale}, \text{Aniso}, \text{proj})
\]

**Arguments**

- **phi**: an \texttt{RMmodel}; has two components \( h \) (2- or 3-dimensional and stationary) and \( t \) (arbitrary dimension).
- **a**: a numerical value; should be in the interval \([-1, 1]\).
- **Dspace**: an integer; either 2 or 3; the first \( Dspace \) coordinates give the first component \( h \).
- **\text{var, scale, Aniso, proj}**: optional arguments; same meaning for any \texttt{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**

\texttt{RMvector} returns an object of class \texttt{RMmodel}.
**Author(s)**

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

**References**


**See Also**

`rmcurlfree`, `rmdivfree`, `rmmodel`, `rfsimulate`, `Rfit`.

**Examples**

```r
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))
```

---

**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

$$C(r) = \frac{\sin(r)}{r} 1_{r>0} + 1_{r=0}.$$ 

It is a special case of `Rmbessel`.

**Usage**

```r
RMwave(var, scale, Aniso, proj)
RMcardinalsinewave(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](http://ms.math.uni-mannheim.de)

**References**


**See Also**

`RMbessel`, `RMmodel`, `RFSimulate`, `RFFit`.

**Examples**

```r
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))
```

---

**Description**

`RMwhittlematern` 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 \( K_\nu \) is the modified Bessel function of second kind.

The Matern model is given by

\[
C(r) = 2^{1-\nu} \Gamma(\nu)^{-1} r^{\nu} K_\nu(\sqrt{2\nu r})
\]

The Handcock-Wallis parametrisation is given by

\[
C(r) = 2^{1-\nu} \Gamma(\nu)^{-1} (2\sqrt{\nu r})^{\nu} K_\nu(2\sqrt{\nu r})
\]
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/ν. 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 seperate 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 \( \nu \): 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 \( \nu \) tending to \( \infty \) 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


Tail correlation function (for $\nu \in (0, 1/2]$)


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

```r
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

- **phi**: the **RMmodel**. Either a model for a process or a covariance model must be specified. In the latter case, a Gaussian process **RPgauss** is tacitely assumed.
- **stationary_only**: optional arguments; same meaning as for **RPgauss**. It is ignored if the submodel is a process definition.
- **threshold**: real valued. **RPbernoulli** returns 1 if value of the random field given by phi is equal to or larger than the value of threshold, and 0 otherwise. In the multivariate case, a vector might be given. If the threshold is not finite, then the original field is returned. threshold 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](http://ms.math.uni-mannheim.de)

See Also

Auxiliary **RMmodels**, **RP**, **RMBernoulli**.
Examples

```r
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

```r
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`. 
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)

---

**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.
  • 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, RPopit, 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, \texttt{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>, \url{http://ms.math.uni-mannheim.de}

See Also

\texttt{RMmodel}, \texttt{RP}, \texttt{RPcoins}.

Examples

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

# example 1: Poisson field based on disks with radius 1
x <- seq(0,25, 0.02)
model <- RMball()
z <- RFsimulate(RPpoisson(model), x, intensity = 2)
plot(z)
par(mfcol=c(2,1))
plot(z@data[,1:min(length(z@data), 1000)], type="l")
hist(z@data[,1], breaks=0.5 + (-1 : max(z@data)))

# example 2: Poisson field based on the normal density function
# note that
# (i) the normal density as unbounded support that has to be truncated
# (ii) the intensity is high so that the CLT holds
x <- seq(0, 10, 0.01)
model <- Rmtruncsupport(radius=5, RMgauss())
z <- RFsimulate(RPpoisson(model), x, intensity = 100)
plot(z)
\end{verbatim}

\texttt{RPprocess}  
Models for classes of random fields (\texttt{RP} commands)

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

```r
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 \texttt{RMmodel}. If a model for the distribution is not specified, \texttt{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 \texttt{RFboxcox} for details.

\nu 
non-negative number. Degree of freedom.

Value

The function returns an object of class \texttt{RMmodel}.

Author(s)

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

References

Related to the extremal t process


See Also

\texttt{Auxiliary RMmodels}, \texttt{RP}, \texttt{RPGauss}.

Examples

\begin{verbatim}
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)
\end{verbatim}
**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](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 <- Rfpdist(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 new.env().

...
  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 fist '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=model1, 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.
description

**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 **RMmodel**
mu  location shift
scale  scale modification
pow  argument for internal use only

Details

It has the same effect as RRdistr(norm(mu=mu, sd=sd, log=log)).

Value

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

```r
RFoptions(seed=0) #*ANY* simulation will have the random seed 0; set
RFoptions(seed=NA) to make them all random again
r <- RRdistr(RRgauss(mu=c(1,5)), n=1000, dim=2)
plot(r[1,], r[2,])
```
Value

**RRloc** returns an object of class **RRmodel**.

Author(s)

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

See Also

**RRmodel**, **RRdistr**, **RRgauss**.

Examples

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

## empirical density of the distribution 'RRspheric'
model <- RRspheric(balldim=2)
hist(RFrndistr(model, n=1000), 50)

## empirical density of the distribution 'RRspheric', shifted by 3
model <- RRloc(mu=3, RRspheric(balldim=2))
hist(RFrndistr(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

```r
RRmcmc(phi, mcmc_n, sigma, normed, maxdensity, rand.loc, gibbs)
```

Arguments

- `phi`: an arbitrary integrable function
- `mcmc_n`: positive integer. Every `mcmc_n`th 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: TRUE.
**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 RModel.

**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**

RModel, RR, RRdistr, RRuser.

**Examples**

```r
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 <- RRdistr(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**

```r
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 $f$ with respect to the $l_1$ norm, i.e. $f(x) = c\phi(\|x\|_{l_1})$ with some function $\phi$. Here, $c$ is a norming constant so that the integral of $f$ equals one.
In case that $\phi$ is monotonically decreasing then rejection sampling is used, else MCMC.
The function $\phi$ 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 <- RRdistr(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(RMgeneric(), approx=FALSE) 
z <- RRfdistr(distr, n=1000000)
hist(z, 200, freq=!TRUE)

x <- seq(-10, 10, 0.01)
lines(x, 4/3 * RFcov(RMgeneric(), x))

RRspheric  Random scaling used with balls

Description
This model delivers the distribution of the **radius** of a ball obtained
by the intersection of a **balldim**-dimensional ball with **diameter** R by a **spacedim**-dimensional hyperplane that has uniform
distance from the center.

Usage
RRspheric(spacedim, balldim, R)

Arguments
- spacedim  dimension of the hyperplane; defaults to 1.
- balldim   the dimension of the ball

Value
RRspheric returns an object of class **RMmodel**.

Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>, [http://ms.math.uni-mannheim.de](http://ms.math.uni-mannheim.de)

See Also
RRmodel, RMball.

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

hist(RRfdistr(RRspheric(balldim=2), n=10000), 50)
RRunif

Uniform Distribution in Higher Dimensions

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 \texttt{RRdistr(unif}(min=\texttt{min, max=}\texttt{max, log=}log)\texttt{)}.

Value

\texttt{RRunif} returns an object of class \texttt{RMmodel}.

Author(s)

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

See Also

\texttt{RMmodel}, \texttt{RRdistr}, \texttt{RRgauss}, \texttt{RRspheric}.

Examples

\texttt{RFoptions(seed=}0\texttt{)} \# *ANY* simulation will have the random seed 0; set
\# \texttt{RFoptions(seed=}NA\texttt{)} to make them all random again
\# uniform distribution on [0,2] x [-2, -1]
\texttt{RFrdistr(RRunif(c(0, -2), c(2, -1)), n=5, dim=2)}
\texttt{RFpdistr(RRunif(c(0, -2), c(2, -1)), q=c(1, -1.5), dim=2)}
\texttt{RFddistr(RRunif(c(0, -2), c(2, -1)), x=c(1, -1.5), dim=2)}
Models for stationary max-stable random fields

Description

Here, the code of the paper on ‘Models for stationary max-stable random fields’ is given.

Author(s)

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

References


Examples

RFoptions(seed=0, xi=1)
## seed = 0 : *ANY* simulation will have the random seed 0; set
## 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)

On some covariance models based on normal scale mixtures

Description

Here, the code of the paper on ‘On some covariance models based on normal scale mixtures’ is given.
Author(s)

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

References


Examples

```r
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


Examples
Description
Sequential method relying on square roots of the covariance matrix

Usage
\texttt{RPsequential(phi, boxcox, back\_steps, initial)}

Arguments

\begin{itemize}
\item \texttt{phi} \hspace{1cm} object of class \texttt{RMmodel}; specifies the covariance model to be simulated.
\item \texttt{boxcox} \hspace{1cm} the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. See \texttt{RFboxcox} for details.
\item \texttt{back\_steps} \hspace{1cm} Number of previous instances on which the algorithm should condition. If less than one then the number of previous instances equals \texttt{max} / (number of spatial points).
\hspace{1cm} Default: \texttt{QP}.
\item \texttt{initial} \hspace{1cm} First, \texttt{N=(number of spatial points) * back\_steps} number of points are simulated. Then, sequentially, all spatial points for the next time instance are simulated at once, based on the previous \texttt{back\_steps} instances. The distribution of the first \texttt{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 \texttt{initial} is non-negative, then \texttt{initial} first steps are performed. If \texttt{initial} is negative, then \texttt{back\_steps - initial} initial steps are performed. The latter ensures that none of the very first \texttt{N} variables are returned.
\hspace{1cm} Default: \texttt{-10}.
\end{itemize}

Details
\texttt{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
\texttt{RPsequential} returns an object of class \texttt{RMmodel}.

Author(s)
Martin Schlather, \texttt{<schlatter@math.uni-mannheim.de>}, \url{http://ms.math.uni-mannheim.de}
References


See Also

Gaussian, RP, RPdirect.

Examples

```r
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

RPsSmith defines a moving maximum process or a mixed moving maximum process with finite number of shape functions.

Usage

RPsSmith(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 `RMMn2r` 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 $x_i$ is always a number, i.e. $\xi$ is constant in space. In contrast, $\mu$ 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 $\mu$ and $s$ are 0 and 1, respectively. For $x_i \neq 0$, the default values of $\mu$ and $s$ are 1 and $|\xi|$, respectively, so that it defaults to the standard Frechet case if $\xi > 0$.

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 $\mathbb{R}^d \times (0, \infty)$ 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


See Also

Advanced RMmodels, Auxiliary RMmodels, RMmodel, RPbernoulli, RPgauuss, 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
soil

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


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")],
  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"),...
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**.

- **RMMppplus**
- **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**


**See Also**

Gaussian, RP.

**Examples**

```r
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.
```
Spectral

Spectral turning bands method

Description

The spectral turning bands method is a simulation method for stationary Gaussian random fields (Mantoglou and Wilson, 1982). It makes use of Bochners’s theorem and the corresponding spectral measure $\Xi$ for a given covariance function $C(h)$. For $x \in \mathbb{R}^d$, the field

$$Y(x) = \sqrt{2} \cos(<V,x> + 2\pi U)$$

with $V \in \Xi$ and $U \in \mathcal{U}((0,1))$ 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

RPspatial(phi, boxcox, sp_lines, sp_grid, prop_factor, sigma)

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.
- **sp_lines**: Number of lines used (in total for all additive components of the covariance function). Default: 2500.
- **sp_grid**: Logical. The angle of the lines is random if `grid=FALSE`, and $k\pi$/sp_lines for $k$ in 1:sp_lines, otherwise. This argument is only considered if the spectral measure, not the density is used. Default: TRUE.
- **prop_factor**: positive real value. Sometimes, the spectral density must be sampled by MCMC. Let $p$ be the average rejection rate. Then the chain is sampled every $n$th point where $n = \|\log(p)\| \cdot \text{prop_factor}$. Default: 50.
Spherical models

sigma

real. Considered if the Metropolis algorithm is used. It gives the standard deviation of the multivariate normal distribution of the proposing distribution. If sigma is not positive then RandomFields tries to find a good choice for sigma 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


See Also

Gaussian, RP, RPtbm.

Examples

RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
model <- 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).
Spherical models

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
- **Rmgencauhcy**: 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 $\alpha \leq 1$
- **RMMwittle**: Whittle-Matern model, alternative parametrization with $\nu \leq 1/2$

**Other isotropic models with arbitrary scale**

- **RMconstant**: spatially constant model
- **RMnugget**: nugget effect model

**Compactly supported covariance functions allowing for scales up to $\pi$ (or 180 degrees)**

- **RMaskey**: Askey’s model
- **RMcircular**: circular 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

See **RMmodels** for cartesian models.

**Author(s)**

Martin Schlather, <schlather@math.uni-mannheim.de>, [http://ms.math.uni-mannheim.de](http://ms.math.uni-mannheim.de)
See Also

cordinate 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) + RManiso(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 ~ min
plot(z)

z <- RFsimulate(RMwhittle(s=500, nu=0.5), coord, grid=TRUE,
    new_coord_sys = "orthographic", zenith=c(25, 25))
plot(z)

z <- RFsimulate(RMwhittle(s=500, nu=0.5), coord, grid=TRUE,
    new_coord_sys = "gnomonic", zenith=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) * RMtrans(RMexp(s=500, proj="space"), "cartesian") * 
    RMepispheric(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)
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


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)
Covariance Models for Random Vector Fields

Description

Here, the code of the paper on ‘Covariance Models for Random Vector Fields’ is given.

Author(s)

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

References


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 `RPsSmith` 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 `RPsSmith` 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 `RPsSmith` defines random hyperplane polygons that correspond to a tail correlation function belonging to Gneiting’s class $H_d$. It currently only allows for $\text{RmBrownResnick}(\text{RMFBm}(\alpha=1))$ 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](http://ms.math.uni-mannheim.de)

References


See Also

`RFsimulate`, `RMmodel`.

Examples

```r
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(RPsSmith(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
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


See Also

`coordinate systems, RM, RMmodels, RMtrafo`.

Examples

```r
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 RMTbm.
For details see RMTbm.

Usage

RPTbm(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 RFgetModelNames(type="positive definite"), which is valid in dimension fulldim.
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 fulldim. The dimension of the auxiliary hyperplane (most frequently a line, i.e. reduceddim=1) used in the simulation.
layers a boolean value; for space-time model. If TRUE then the turning layers are used whenever a time component is given. If NA the turning layers are used only when the traditional TBM is not applicable. If FALSE then turning layers may never be used.
Default: TRUE.
lines Number of lines used. Default: 60.
linessimufactor linessimufactor or linesimustep must be non-negative; if linesimustep is positive then 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 RMnsst, 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


Turning layers


See Also

Gaussian, RP, RPspectral.
Examples

```r
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
nThe 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

require(geoR)
data(ca20) ## data set from geoR
cat.R.df <- as.data.frame(ca20)
head(ca20.R.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.R.df is a
data.frame
cat.R.FRmod02 ~ 1 + altitude + M
(cat.R.fit02.RF <- RFFit(cat.R.FRmod02, data=cat.R.df, M=M))

## long definition which also allows for more general constructions
cat.R.FRmod02 <- NA + NA + RMcovariate(cat.R.df$altitude) + M
(cat.R.fit02.RF <- RFFit(cat.R.FRmod02, data=cat.R.df))

## Note that the following also works.
## Here, the covariance model must be the first summand
cat.R.FRmod02 <- M + NA + cat.R.df$altitude
print(cat.R.fit02.RF <- RFFit(cat.R.FRmod02, data=cat.R.df))

### The following does NOT work, as R assumes (NA + cat.R.df$altitude) + M
### In particular, the model definition gives a warning, and the
### Rfit call gives an error:
(cat.R.FRmod02 <- NA + cat.R.df$altitude + M)
try(cat.R.fit02.RF <- RFFit(cat.R.FRmod02, data=cat.R.df)) ### error ...

## factors:
cat.R.FRmod03 ~ 1 + area + M
cat.R.fit03.RF <- RFFit(cat.R.FRmod03, data=cat.R.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 Celsius
- **lon**: longitudinal coordinates of the locations
- **lat**: latitude coordinates of the locations

Furthermore, some results obtained from the data analysis in jss14 are delivered that are `pars.model`, `pars`, `whole.model`, `whole`.

Finally, the variable `information` contains packing information (the date and the version of RandomFields).

Details

The forecasts are from the GFS member of the University of Washington regional numerical weather prediction ensemble (UWME; Grimit and Mass 2002; Eckel and Mass 2005); they were valid on December 18, 2003 at 4 pm local time, at a forecast horizon of 48 hours.

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


See Also

A reanalysis has been performed in Section 5 of the jss14 paper.

Examples

```r
## See 'jss14'.
```
Index

*Topic **classes**
  RFempVariog-class, 77
  RFFit-class, 84
  RFgridDataFrame-class, 110
  RFpointsDataFrame-class, 153
  RFsp-class, 172
  RFspatialGridDataFrame-class, 174
  RFspatialPointsDataFrame-class, 177
  RMmodel-class, 272
  RMmodelFit-class, 274
  RMmodelGenerator-class, 276
*Topic **datasets**
  soil, 361
  weather, 378
*Topic **distribution**
  Distribution Families, 26
*Topic **hplot**
  RFempVariog-class, 77
  RFFit-class, 84
  RFgridDataFrame-class, 110
  RFpointsDataFrame-class, 153
  RFspatialGridDataFrame-class, 174
  RFspatialPointsDataFrame-class, 177
  RMmodel-class, 272
  RMmodelFit-class, 274
  RMmodelGenerator-class, 276
*Topic **htest**
  RFratioTest, 159
*Topic **methods**
  Brown-Resnick-Specific, 10
  Circulant Embedding, 15
  Coins, 19
  Hyperplane, 37
  Independent Variables, 38
  plot-method, 60
  Sequential, 358
  Specific, 364
  Spectral, 366
  Square root, 369
  Tbm, 374
*Topic **models**
  Mathematical C functions, 48
  RFcov, 69
  RFmadogram, 125
  RFpar, 152
  RFpseudomadogram, 155
  RFpseudovariogram, 157
  RFvariogram, 179
  RMangle, 182
  RMaskey, 183
  RMave, 185
  RMbcw, 188
  RMberoulli, 189
  RMBessel, 190
  RMbicauhy, 192
  RMBigneiting, 193
  RMBistable, 195
  RMBiwm, 197
  RMblend, 199
  RMbubble, 204
  RMCauhy, 207
  RMcauchytbm, 209
  RMCchoquet, 210
  RMCircular, 211
  RMCconstant, 212
  RMCov, 213
  RMCovariate, 215
  RMCoxisham, 216
  RMCubic, 218
  RMcurlfree, 219
  RMcutoff, 220
  RMDagum, 222
  RMDampedcos, 223
  RMDelay, 224
  RMDeriv, 226

380
INDEX

RMdewijsian, 228
RMdivfree, 230
RMexaxa, 231
RMepscauchy, 232
RMexp, 234
RMexpontential, 235
RMfbm, 237
RMfixcov, 238
RMfixed, 240
RMflatpower, 240
RMfractdiff, 242
RMfractgauss, 243
RMgauss, 244
Rmgencauchy, 245
RMgenfbm, 247
RMgengneiting, 248
RMgennsst, 250
RMgneiting, 251
RMgneitingdiff, 252
RMhypercubic, 254
RMIaco, 255
RMIid, 256
RMIidmodel, 257
RMintexp, 259
RMintrinsic, 260
RMkolmogorov, 261
RMlgd, 262
RMlsfbm, 264
RMma, 265
RMMastein, 266
RMMmatrix, 268
RMMmodel, 270
RMMppplus, 287
RMMqam, 288
RMMult, 290
RMMultiquad, 291
RMMonstwm, 294
RMSsst, 295
RMNugget, 296
RMparswm, 297
RMPenta, 299
RMPplus, 300
RMPolynomine, 302
RMPower, 303
RMProd, 304
RMMqam, 305
RMPexp, 307
RMrational, 308
RMrotat, 309
RMS, 310
RMSadvanced, 311
RMScale, 312
RMSchlather, 314
RMSchur, 315
RMSign, 316
RMSinepower, 317
RMSpheric, 318
RMStable, 320
RMSstein, 321
RMStp, 322
RMSum, 324
RMTbm, 325
RMTbrafo, 327
RMTrend, 329
RMTrunccsupport, 331
RMSuser, 332
RMSvector, 334
RMSwave, 335
RMSwhittlematern, 336
RRdeterm, 346
RRdistr, 346
RRgauss, 348
RRloc, 349
RRmcmc, 350
RRrectangular, 351
RRspheric, 353
RRunif, 354
Spherical models, 367
Tail Correlation Functions, 373

*Topic optimize
  fitgauss, 31
  RFFit, 81
  RFFitOptimiser, 88

*Topic print
  RFempVariog-class, 77
  RFFit-class, 84
  RFGgridDataFrame-class, 110
  RFpointsDataFrame-class, 153
  RFspatialGridDataFrame-class, 174
  RFspatialPointsDataFrame-class, 177
  RMmodel-class, 272
  RMmodelFit-class, 274

*Topic spatial
  BrownResnick, 12
  Constants, 20
conventional2RFspDataFrame, 21
Coordinate systems, 22
Distribution Families, 26
Extremal t, 28
Extremal Gaussian, 29
fitgauss, 31
GaussianFields, 32
GSPSJ06, 34
Hierarchical Modelling, 35
Internal functions, 40
jss14, 42
Mathematial C functions, 48
Max-stable random fields, 53
Max-stable random fields, advanced, 55
Obsolete Functions Version 2, 55
Obsolete Functions Version 3, 57
Others, 58
papers, 59
PrintModellist, 66
RandomFields-package, 6
RBoxcox, 67
RFCov, 69
RFCovmatrix, 72
RFCrossValidate, 73
RFdistr, 76
RFfctn, 80
RFfit, 81
RFfitoptimiser, 88
RFformula, 90
RFfractaldim, 94
RFfunction, 97
RFgetMethodNames, 99
RFgetModel, 103
RFgetModelInfo, 104
RFgetModelNames, 106
RFgui, 112
RFhurst, 113
RFinterpolate, 116
RFlinearpart, 120
RFloglikelihood, 122
RFmadogram, 125
RFfoldstyle, 128
RFoptions, 129
RFpar, 152
RFpseudomadogram, 155
RFpseudovariogram, 157
RFRatioTest, 159
RFsimulate, 162
RFsimulateAdvanced, 167
RFvariogram, 179
RMangle, 182
RMaskey, 183
RMave, 185
RMBall, 187
RMBcw, 188
RMBernoulli, 189
RMBessel, 190
RMBicauchy, 192
RMBigneiting, 193
RMBistable, 195
RMBjwm, 197
RMBblend, 199
RMBr2bg, 200
RMBr2eg, 201
RMBrownresnick, 203
RMBubble, 204
RMCauchy, 207
RMCauchytm, 209
RMCchoquet, 210
RMCircular, 211
RMCconstant, 212
RMCov, 213
RMCovariate, 215
RMCoxisham, 216
RMCubic, 218
RMCurlfree, 219
RMCutoff, 220
RMDagum, 222
RMDampedcos, 223
RMDelay, 224
RMDelay, 226
RMDeriv, 227
RMDewijsian, 228
RMDivfree, 230
RMeaxxa, 231
RMEpscauchy, 232
RMExp, 234
RMExponential, 235
RFbm, 237
RFfixcov, 238
RFflatpower, 240
RFfractdiff, 242
RFgaussian, 243
RFGauss, 244
RGMencauchy, 245
*, character, RModel-method
   (Mathematical C functions), 48
*, logical, RModel-method
   (RModel-class), 272
*, numeric, RModel-method
   (RModel-class), 272
+., 90, 108, 286
+ (RMplus), 300
++ (RMmpplus), 287
+, RModel, RModel-method
   (RModel-class), 272
+, RModel, character-method
   (Mathematical C functions), 48
+, RModel, factor-method (Mathematical C functions), 48
+, RModel, list-method (Mathematical C functions), 48
+, RModel, logical-method
   (RModel-class), 272
+, RModel, numeric-method
   (RModel-class), 272
+, character, RModel-method
   (Mathematical C functions), 48
+, data.frame, RModel-method
   (Mathematical C functions), 48
+, factor, RModel-method (Mathematical C functions), 48
+, logical, RModel-method
   (RModel-class), 272
+, numeric, RModel-method
   (RModel-class), 272
- (Mathematical C functions), 48
-, RModel, RModel-method
   (RModel-class), 272
-, RModel, character-method
   (Mathematical C functions), 48
-, RModel, logical-method
   (RModel-class), 272
-, RModel, numeric-method
   (RModel-class), 272
-, character, RModel-method
   (Mathematical C functions), 48
-, logical, RModel-method
   (RModel-class), 272
-, numeric, RModel-method
   (RModel-class), 272
.RF_fit (RFfit-class), 84
.RFfit (RFfit-class), 84
.Random.seed, 160, 171
// (Mathematical C functions), 48
/., RModel, RModel-method
   (RModel-class), 272
/., RModel, character-method
   (Mathematical C functions), 48
/., RModel, logical-method
   (RModel-class), 272
/., RModel, numeric-method
   (RModel-class), 272
/., character, RModel-method
   (Mathematical C functions), 48
/., logical, RModel-method
   (RModel-class), 272
/., numeric, RModel-method
   (RModel-class), 272
[., RFfit, ANY, ANY, ANY-method
   (RFfit-class), 84
[., RFfit, ANY, ANY-method (RFfit-class), 84
[., RFfit, ANY-method (RFfit-class), 84
[., RFgridDataframe, ANY, ANY, ANY-method
   (RFgridDataframe-class), 110
[., RFgridDataframe, ANY, ANY-method
   (RFgridDataframe-class), 110
[., RFgridDataframe-method
   (RFgridDataframe-class), 110
[., RFpointsDataframe, ANY, ANY, ANY-method
   (RFpointsDataframe-class), 153
[., RFpointsDataframe, ANY, ANY-method
   (RFpointsDataframe-class), 153
[., RFpointsDataframe-method
   (RFpointsDataframe-class), 153
[., RFsp, ANY, ANY, ANY-method (RFsp-class), 172
[., RFsp, ANY, ANY-method (RFsp-class), 172
[., RFsp-method (RFsp-class), 172
[., RFspatialGridDataFrame, ANY, ANY, ANY-method
   (RFspatialGridDataFrame-class), 174
[., RFspatialGridDataFrame, ANY, ANY-method
   (RFspatialGridDataFrame-class), 174
[., RFspatialGridDataFrame-method
   (RFspatialGridDataFrame-class), 174
[., RFspatialPointsDataFrame, ANY, ANY, ANY-method
   (RFspatialPointsDataFrame-class), 177
INDEX

[,RFspatialPointsDataFrame,ANY,ANY-method
  (RFspatialPointsDataFrame-class), 177
[,RFspatialPointsDataFrame-method
  (RFspatialPointsDataFrame-class), 177
[,RModel,ANY,ANY,ANY-method
  (RModel-class), 272
[,RModel,ANY,ANY-method
  (RModel-class), 270
[,RModel-method (RModel-class), 272
[,RModelFit,ANY,ANY,ANY-method
  (RModelFit-class), 274
[,RModelFit,ANY,ANY-method
  (RModelFit-class), 274
[,RModelFit-method (RModelFit-class), 274
[,RModelgenerator,ANY,ANY,ANY-method
  (RModelgenerator-class), 276
[,RModelgenerator,ANY,ANY-method
  (RModelgenerator-class), 276
[,RModelgenerator-method
  (RModelgenerator-class), 276
<-,RFgridDataFrame,ANY,ANY-method
  (RFgridDataFrame-class), 110
<-,RFgridDataFrame-method
  (RFgridDataFrame-class), 110
<-,RFpointsDataFrame,ANY,ANY-method
  (RFpointsDataFrame-class), 153
<-,RFpointsDataFrame-method
  (RFpointsDataFrame-class), 153
<-,RFsp,ANY,ANY-method
  (RFsp-class), 172
<-,RFsp,ANY-method (RFsp-class), 172
<-,RFsp-method (RFsp-class), 172
<-,RFspatialGridDataFrame,ANY,ANY,ANY-method
  (RFspatialGridDataFrame-class), 174
<-,RFspatialGridDataFrame-method
  (RFspatialGridDataFrame-class), 174
<-,RFspatialPointsDataFrame,ANY,ANY,ANY-method
  (RFspatialPointsDataFrame-class), 177
<-,RFspatialPointsDataFrame-method
  (RFspatialPointsDataFrame-class), 177
<-,RModel,ANY,ANY,ANY-method
  (RModel-class), 272
<-,RModel,ANY,ANY-method
  (RModel-class), 270
<-,RModel-method (RModel-class), 272
<-,RModelFit,ANY,ANY,ANY-method
  (RModelFit-class), 274
<-,RModelFit,ANY,ANY-method
  (RModelFit-class), 274
<-,RModelFit-method
  (RModelFit-class), 274
<-,RModelgenerator,ANY,ANY,ANY-method
  (RModelgenerator-class), 276
<-,RModelgenerator,ANY,ANY-method
  (RModelgenerator-class), 276
<-,RModelgenerator-method
  (RModelgenerator-class), 276
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
%,RModel-method (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
% (Mathematical C functions), 48
abs (Mathematical C functions), 48
abs (Mathematical C functions), 48
acosh (Mathematical C functions), 48
acosh (Mathematical C functions), 48
Advanced R Models, 109
Advanced RM models (RMmodelsAdvanced), 280
AIC, 7
AIC, RFfit-method (RFfit-class), 84
AIC.RF_fit (RFfit-class), 84
AICc, 7
AICc.RF_fit (RFfit-class), 84
AICc.RFfit (RFfit-class), 84
Aniso, 281, 285
anova, 7, 139
anova, RFfit-method (RFfit-class), 84
anova, RMModelFit-method (RMmodelFit-class), 274
anova.RF_fit (RFfit-class), 84
anova.RM_ModelFit (RMmodelFit-class), 274
approx_step, 32
areamat, 133, 134
areamat=1, 134
as.array.RFgridDataFrame (RFgridDataFrame-class), 110
as.array.RFpointsDataFrame (RFpointsDataFrame-class), 153
as.array.RFspatialGridDataFrame (RFspatialGridDataFrame-class), 174
as.array.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 177
as.data.frame.RFgridDataFrame (RFgridDataFrame-class), 110
as.data.frame.RFpointsDataFrame (RFpointsDataFrame-class), 153
as.data.frame.RFspatialGridDataFrame (RFspatialGridDataFrame-class), 174
as.data.frame.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 177
as.matrix.RFgridDataFrame (RFgridDataFrame-class), 110
as.matrix.RFpointsDataFrame (RFpointsDataFrame-class), 153
as.matrix.RFspatialGridDataFrame (RFspatialGridDataFrame-class), 174
as.matrix.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 177
177
as.vector.RFgridDataFrame (RFgridDataFrame-class), 110
as.vector.RFpointsDataFrame (RFpointsDataFrame-class), 153
as.vector.RFspatialGridDataFrame (RFspatialGridDataFrame-class), 174
as.vector.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 177
asin (Mathematical C functions), 48
asin, RModel-method (Mathematical C functions), 48
asinh (Mathematical C functions), 48
asinh, RModel-method (Mathematical C functions), 48
atan (Mathematical C functions), 48
atan, RModel-method (Mathematical C functions), 48
atan2 (Mathematical C functions), 48
atan2, ANY, RModel-method (Mathematical C functions), 48
atan2, RModel, ANY-method (Mathematical C functions), 48
atanh (Mathematical C functions), 48
atanh, RModel-method (Mathematical C functions), 48
Auxiliary Models (Others), 58
Auxiliary RM models, 281
Auxiliary RModels (Others), 58
AuxiliaryModels (Others), 58
Average (Coins), 19
back_steps, 33
Bayesian, 47, 68, 121, 123, 279
Bayesian (Hierarchical Modelling), 35
bayesian (Hierarchical Modelling), 35
Bayesian Modelling, 7, 26, 347
Bayesian Modelling (Hierarchical Modelling), 35
BIC, 7
BIC, RFfit-method (RFfit-class), 84
BIC.RF_fit (RFfit-class), 84
Binary field, 344
Binary fields, 162, 168
binary processes, 47
BRmethods (Brown-Resnick-Specific), 10
Brown-Resnick, 374
INDEX

Brown-Resnick (BrownResnick), 12
Brown-Resnick process (BrownResnick), 12
Brown-Resnick-Specific, 10
BrownResnick, 12
c, 51, 268
c (Mathematical C functions), 48
c,RMmodel-method (RMmodel-class), 272
cbind.RFgridDataFrame
   (RFgridDataFrame-class), 110
cbind.RFpointsDataFrame
   (RFpointsDataFrame-class), 153
cbind.RFspatialGridDataFrame
   (RFspatialGridDataFrame-class), 174
cbind.RFspatialPointsDataFrame
   (RFspatialPointsDataFrame-class), 177
cbrt (Mathematical C functions), 48
ceiling (Mathematical C functions), 48
ceiling,RMmodel-method (Mathematical C functions), 48
Changings, 8, 14
changings (Changings), 14
checkExamples (Internal functions), 40
chi2 field, 344
chi2 fields, 162, 167
Chi2 processes, 47
Choquet’s representation (RMchoquet), 210
Circulant (Circulant Embedding), 15
Circulant Embedding, 15
class, 74
coerce,RFempVariog,list-method
   (RFempVariog-class), 77
coerce,RFfit,RFempVariog-method
   (RFfit-class), 84
coerce,RFgridDataFrame,RFpointsDataFrame,ANY-method
   (RFgridDataFrame-class), 110
coerce,RFgridDataFrame,RFpointsDataFrame-method
   (RFgridDataFrame-class), 110
coerce,RFpointsDataFrame,RFgridDataFrame,ANY-method
   (RFpointsDataFrame-class), 153
coerce,RFpointsDataFrame,RFgridDataFrame-method
   (RFpointsDataFrame-class), 153
coerce,RFspatialGridDataFrame,RFspatialPointsDataFrame,ANY-method
   (RFspatialGridDataFrame-class), 174
coerce,RFspatialGridDataFrame,RFspatialPointsDataFrame-method
   (RFspatialGridDataFrame-class), 174
coerce,RFspatialPointsDataFrame,RFspatialGridDataFrame,ANY-method
   (RFspatialPointsDataFrame-class), 177
coerce,RFspatialPointsDataFrame,RFspatialGridDataFrame-method
   (RFspatialPointsDataFrame-class), 177
coerce,RFspatialPointsDataFrame,RFspatialGridDataFrame-method
   (RFspatialPointsDataFrame-class), 177
coerce,RFspatialPointsDataFrame,RFspatialGridDataFrame-method
   (RFspatialPointsDataFrame-class), 177
Coins, 19, 144
composed Poisson, 344
compound Poisson processes, 47
Condsimu (Obsolete Functions Version 2), 55
Constants, 20
constants, 107, 109, 328
constants (Constants), 20
contour, 175
contour,RFempVariog (RFempVariog-class), 84
contour,RFfit (RFfit-class), 84
contour,RFspatialGridDataFrame
   (plot-method), 60
conventional2RFspDataframe, 21, 110, 154,
   174, 177
cord_units (Changings), 14
coordinate system, 51
coordinate system (Coordinate systems), 22
Coordinate systems, 22, 117, 163
Coordinate systems, 7, 129, 135, 369, 374
coordinate systems (Coordinate systems), 22
coordinates (Coordinate systems), 22
coregionalisation (RMmatrix), 268
coregionalization (RMmatrix), 268
cos (Mathematical C functions), 48
cos, RMmodel-method (Mathematial C functions), 48
cosh (Mathematial C Functions), 48
cosh, RMmodel-method (Mathematial C functions), 48
Covariance (Obsolete Functions Version 2), 55
CovarianceFct (Obsolete Functions Version 2), 55
CovMatrix (Obsolete Functions Version 2), 55
CRS, 175, 177
Cutoff (Circulant Embedding), 15
data.frame, 110, 154, 175, 177
DeleteAllRegisters (Obsolete Functions Version 2), 55
DeleteRegister (Obsolete Functions Version 2), 55
Dependencies (Internal functions), 40
derivative (RMderiv), 227
dimensions, RFDataFrame-method (RFsp-class), 172
dimensions, RFgridDataFrame-method (RFgridDataFrame-class), 110
dimensions, RFpointsDataFrame-method (RFpointsDataFrame-class), 153
dimensions, RFsp-method (RFsp-class), 172
dimensions, RFspatialDataFrame-method (RFsp-class), 172
dimensions, RFspatialGridDataFrame-method (RFspatialGridDataFrame-class), 174
dimensions, RFspatialPointsDataFrame-method (RFspatialPointsDataFrame-class), 177
Direct (Square root), 369
dist, 169, 327, 328
Distribution Families, 26
Distributions, 47
DoSimulateRF (Obsolete Functions Version 2), 55
Earth models (Spherical models), 367
earth models (Spherical models), 367
EmpiricalVariogram (Obsolete Functions Version 2), 55
erf (Mathematial C functions), 48
erfc (Mathematial C functions), 48
errof function model (RMbrownresnick), 203
exp (Mathematial C functions), 48
exp, RMmodel-method (Mathematial C functions), 48
exp2 (Mathematial C functions), 48
expm1 (Mathematial C Functions), 48
expm1, RMmodel-method (Mathematial C functions), 48
extremal Gaussian (ExtremalGaussian), 29
extremal Gaussian process (ExtremalGaussian), 29
Extremal t, 28
extremal t (Extremal t), 28
extremal t process (Extremal t), 28
ExtremalGaussian, 29
FinalizeExample (Internal functions), 40
fitgauss, 31, 83, 159
fitvario (Obsolete Functions Version 2), 55
floor (Mathematial C functions), 48
floor, RMmodel-method (Mathematial C functions), 48
formula, 51, 69, 72, 74, 78, 80, 82, 105, 117, 121, 122, 125, 156, 158, 163, 166, 179
formula notation, 47
fractal.dim (Obsolete Functions Version 2), 55
function, 276
gamma (Mathematial C functions), 48
Gaussian, 18, 20, 38, 39, 342, 344, 359, 365, 367, 370, 376
Gaussian (GaussianFields), 32
Gaussian random fields, 73, 81, 83, 162
GaussianFields, 32
GaussRF (Obsolete Functions Version 2), 55
GetModelList (PrintModellist), 66
GetModelNames (PrintModellist), 66
GKS11, 60
GKS11 (weather), 378
gradient (RMderiv), 227
GridTopology, 22, 69, 72, 74, 80, 82, 95, 110, 114, 117, 121, 122, 126, 156, 158, 160, 163, 168, 175, 180
INDEX

GridTopology2gridVectors, GridTopology-method, lgamma, Rmmodel-method (Mathematical C functions), 48
linear models, 73, 87, 83
lines, Rmmodel-method (RMmodel-class), 272
lines, RMmodel (RMmodel-class), 272
list, 79
lm, 96, 115, 215
log (Mathematical C functions), 48
log, RMmodel-method (Mathematical C functions), 48
log1p (Mathematical C functions), 48
log1p, RMmodel-method (Mathematical C functions), 48
log2 (Mathematical C functions), 48
log2, RMmodel-method (Mathematical C functions), 48
logLik, 7
logLik.RF_fit (RFfit-class), 84
logLik.RFfit (RFfit-class), 84
M2 (Smith), 359
M3 (Smith), 359
maintainers.machine (Internal functions), 40
Major Revisions, 46
MajorRevisions, 8, 15
MajorRevisions (Major Revisions), 46
math.c (Mathematical C functions), 48
Mathematical C functions, 48
Mathematical functions, 279
matrix_methods, 32, 33
max (Mathematical C functions), 48
Max-stable random fields, 53
Max-stable random fields, advanced, 55
Maxstable, 344
Maxstable (Max-stable random fields), 53
maxstable, 12, 14, 29, 30, 360
maxstable (Max-stable random fields), 53
maxstable processes, 47
maxstableAdvanced, 12, 14, 29, 30, 53, 54, 201, 202, 360
maxstableAdvanced (Max-stable random fields, advanced), 55
MaxStableRF (Obsolete Functions Version 2), 55
method specification, 169
min (Mathematical C functions), 48
mixed model, 258, 277

GridTopology2gridVectors, GridTopology-method 174
GridTopology2gridVectors, matrix-method 174
GSPS06, 34, 60
Hierarchical (Hierarchical Modelling), 35
Hierarchical Modelling, 35
hist.RFgridDataFrame 110
hist.RFpointsDataFrame 153
hist.RFspatialGridDataFrame 174
hist.RFspatialPointsDataFrame 177
hurst (Obsolete Functions Version 2), 55
Hyperplane, 37
Hyperplanes (Hyperplane), 37
hypot (Mathematical C functions), 48
image, 79
image, RMmodel-method (RMmodel-class), 272
Independent Variables, 38
InitGaussRF (Obsolete Functions Version 2), 55
InitMaxStableRF (Obsolete Functions Version 2), 55
InitSimulateRF (Obsolete Functions Version 2), 55
interactive, 141
Internal functions, 40
Intrinsic (Circulant Embedding), 15
Inverse multiquadric (RMmultiquad), 291
jss14, 42, 60, 379
Kriging (Obsolete Functions Version 2), 55
kriging (RFinterpolate), 116
lambda, 134
lgamma (Mathematical C functions), 48
mixed moving maxima (Smith), 359
moving maxima (Smith), 359
multiquadric family (RMmultiquad), 291
Multivariate and vector-valued random fields, 47
Multivariate RMmodels, 14, 193, 196, 198
Multivariate RMmodels (RMmodelsMultivariate), 283

new.env, 347
new_coord_units (Changes), 14
non-stationary (RMmodelsNonstationary), 285
non-stationary RMmodels (RMmodelsNonstationary), 285
Nonstationary RMmodels (RMmodelsNonstationary), 285
norm, 349
Normal, 348
Nugget (Independent Variables), 38

Obsolete Functions Version 2, 55
Obsolete Functions Version 3, 57
optim, 74, 82, 83, 88, 137–139, 160
options, 47
Other models (Others), 58
Others, 58

papers, 7, 59
par, 153
pdf, 141
persp, 63, 79, 87, 175, 273
persp, RFempVarioi-method
(RFempVarioi-class), 77
persp, RFfit-method (RFfit-class), 84
persp, RFspatialGridDataFrame-method
(plot-method), 60
persp, RMmodel-method (RMmodel-class), 272
plot, 7, 8, 141
plot, RFempVarioi, missing-method
(RFempVarioi-class), 77
plot, RFfit, missing-method
(RFfit-class), 84
plot, RFgridDataFrame, data.frame-method
(plot-method), 60
plot, RFgridDataFrame, matrix-method
(plot-method), 60
plot, RFgridDataFrame, matrix-method
(plot-method), 60
plot, RFpointsDataFrame, data.frame-method
(plot-method), 60
plot, RFpointsDataFrame, matrix-method
(plot-method), 60
plot, RFspatialGridDataFrame, data.frame-method
(plot-method), 60
plot, RFspatialGridDataFrame, matrix-method
(plot-method), 60
plot, RFspatialGridDataFrame, missing-method
(plot-method), 60
plot, RFspatialGridDataFrame, RFspatialGridDataFrame-method
(plot-method), 60
plot, RFspatialPointsDataFrame, data.frame-method
(plot-method), 60
plot, RFspatialPointsDataFrame, matrix-method
(plot-method), 60
plot, RFspatialPointsDataFrame, missing-method
(plot-method), 60
plot, RFspatialPointsDataFrame, RFspatialGridDataFrame-method
(plot-method), 60
plot, RFspatialPointsDataFrame, RFspatialPointsDataFrame-method
(plot-method), 60
plot, RMmodel, missing-method
(RMmodel-class), 272
plot-method, 60, 153
plotWithCircles (Internal functions), 40
points, 273
points, RMmodel-method (RMmodel-class), 272
points, RMmodel (RMmodel-class), 272
Poisson spline (RMmultiquad), 291
powered error function
(RMBrownresnick), 203
powered exponential, 283
powered exponential (RMstable), 320
print, 8
print,RFempVariog-method
(RFempVariog-class), 77
print,RFfit-method (RFfit-class), 84
print, RMmodelFit-method
(RMmodelFit-class), 274
print.crossvalidate (RFcrossvalidate), 73
print.RF_empVariog (RFempVariog-class), 77
print.RF_fit (RFfit-class), 84
print.RFfit (RFfit-class), 84
print.RFgridDataFrame
(RFgridDataFrame-class), 110
print.RFpointsDataFrame
(RFpointsDataFrame-class), 153
print.RFratiotest (RFratiotest), 159
print.RFspatialGridDataFrame
(RFspatialGridDataFrame-class), 174
print.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class), 177
print.RM_modelFit (RMmodelFit-class), 274
print.RMModel (RMmodel-class), 272
print.RMmodelFit (RMmodelFit-class), 274
print.RMmodelgenerator
(RMmodelgenerator-class), 276
print.summary.crossvalidate
(RFcrossvalidate), 73
PrintModellist, 66
processes, 258
proj, 286
R., 10, 21, 27, 58, 98, 271, 279, 311, 344
R. (Mathematial C functions), 48
R.acos (Mathematial C functions), 48
R.acosh (Mathematial C functions), 48
R.asin (Mathematial C functions), 48
R.asinh (Mathematial C functions), 48
R.atan (Mathematial C functions), 48
R.atan2 (Mathematial C functions), 48
R.atanh (Mathematial C functions), 48
R.c, 281, 284
R.c (Mathematial C functions), 48
R.cbrt (Mathematial C functions), 48
R.ceil (Mathematial C functions), 48
R.const (Mathematial C functions), 48
R.cos (Mathematial C functions), 48
R.cosh (Mathematial C functions), 48
R.div, 273
R.div (Mathematial C functions), 48
R.erf (Mathematial C functions), 48
R.erfc (Mathematial C functions), 48
R.exp (Mathematial C functions), 48
R.exp2 (Mathematial C functions), 48
R.expm1 (Mathematial C functions), 48
R.fabs (Mathematial C functions), 48
R.fdim (Mathematial C functions), 48
R.floor (Mathematial C functions), 48
R.fmax (Mathematial C functions), 48
R.fmin (Mathematial C functions), 48
R.fmod (Mathematial C functions), 48
R.gamma (Mathematial C functions), 48
R.hypot (Mathematial C functions), 48
R.ilogb (Mathematial C functions), 48
R.is, 145
R.is (Mathematial C functions), 48
R.lat (Mathematial C functions), 48
R.lgamma (Mathematial C functions), 48
R.log (Mathematial C functions), 48
R.log1p (Mathematial C functions), 48
R.log2 (Mathematial C functions), 48
R.lon (Mathematial C functions), 48
R.minus, 273, 329
R.minus (Mathematial C functions), 48
R.models, 97, 281, 284, 285
R.models (Mathematial C functions), 48
R.mult, 273
R.mult (Mathematial C functions), 48
R.nextafter (Mathematial C functions), 48
R.nexttoward (Mathematial C functions), 48
R.p (Mathematial C functions), 48
R.plus (Mathematial C functions), 48
R.pow (Mathematial C functions), 48
R.remainder (Mathematial C functions), 48
R.round (Mathematial C functions), 48
R.sin (Mathematial C functions), 48
R.sinh (Mathematial C functions), 48
R.sq (Mathematial C functions), 48
R.sqrt (Mathematial C functions), 48
R.tan (Mathematial C functions), 48
R.tanh (Mathematial C functions), 48
INDEX

R_trunc (Mathematical C functions), 48
random parameters, 346
RandomFields, 15, 73, 83, 91, 102, 109, 119, 162
RandomFields (RandomFields-package), 6
Randomfields-package, 6
range.RFgridDataFrame
(RFgridDataFrame-class), 110
range.RFpointsDataFrame
(RFpointsDataFrame-class), 153
range.RFspatialGridDataFrame
(RFspatialGridDataFrame-class), 174
range.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class), 177
raster, 69, 72, 74, 80, 82, 95, 114, 117, 121,
122, 126, 156, 158, 160, 163, 168, 180
RC, 10, 27, 98, 271, 279, 344
RC (Constants), 20
RC_CARTESIANCOORD, 327
RC_CARTESIANCOORD (Constants), 20
RC_DOMAIN_NAMES, 109, 333
RC_DOMAIN_NAMES (Constants), 20
RC_DOUBLEISOTROPIC (Constants), 20
RC_EARTHCOORDS (Constants), 20
RC_EARTHISOTROPIC (Constants), 20
RC_GNOMONICPROJECT, 327
RC_GNOMONICPROJECT (Constants), 20
RC_ISO_NAMES, 109, 333
RC_ISO_NAMES (Constants), 20
RC_ISONAMES, 327
RC_ISOTROPIC, 327
RC_ISOTROPIC (Constants), 20
RC_LIKELIHOOD_NAMES (Constants), 20
RC_MONOTONE_NAMES (Constants), 20
RC_NLOPTR_NAMES (Constants), 20
RC_OPTIMISER_NAMES (Constants), 20
RC_ORTHORPHICPROJECT, 327
RC_ORTHORPHICPROJECT (Constants), 20
RC_SPACEISOTROPIC, 327
RC_SPACEISOTROPIC (Constants), 20
RC_SPHERICALCOORDS (Constants), 20
RC_TYPE_NAMES (Constants), 20
RC_UNREDUCED (Constants), 20
residuals.RFfit-method (RFfit-class), 84
RF, 10, 21, 27, 47, 58, 105, 271, 277, 279, 344
RF (RFfunction), 97
RF__name__, 258
RF_fit-class (RFfit-class), 84
RFboxcox, 7, 16, 19, 37, 38, 67, 67, 68, 340,
341, 345, 358, 364, 366, 370, 375
RFCalc, 7, 73, 81, 97, 377
RFCov (Mathematical C functions), 48
RFCov, 7, 14, 57, 58, 69, 70, 73, 81, 97, 127,
157, 159, 181, 271, 277
RFCovmatrix, 7, 57, 58, 72, 72, 80, 81, 97
RFCrossValidate, 7, 73, 98, 129, 130, 136
RFDataFrame (RFsp-class), 172
RFDataFrame (RFsp-class), 172
RFdist (RFdist), 76
RFdist, 27, 76, 97, 132, 347
RFEarth2cartesian, 25, 98
RFEarth2cartesian (RMtrafo), 327
RFEarth2dist, 98
RFEarth2dist (RMtrafo), 327
RFEmpiricalCovariance (Obsolete
Functions Version 3), 57
RFEmpiricalMadogram (Obsolete
Functions Version 3), 57
RFEmpiricalVarioigram (Obsolete
Functions Version 3), 57
RFEmpVario, 272
RFEmpVario, 60, 70, 78, 85, 87, 126, 157,
159, 181
RFEmpVario-class, 77
RFfctn, 7, 52, 73, 80, 80, 81, 97
RFfit, 7, 31, 46, 47, 57, 60, 68, 70, 73–75, 81,
81, 83, 85–88, 90, 91, 98, 123, 127,
129–132, 136, 139, 140, 149, 157,
159–162, 165, 172, 181, 184, 186,
189, 191, 195, 198, 208, 209, 211,
212, 214, 217, 218, 220, 221,
223–225, 227, 229, 231, 233, 235,
236, 238–242, 244, 245, 247–250,
252, 253, 255, 259, 261, 263, 264,
266, 267, 269, 272, 275, 288–290,
292–294, 296–299, 301, 302, 304,
306, 318, 319, 321, 322, 324, 333,
335, 336, 338
RFfit-class, 84
RFfitOptimiser, 83, 136, 138
RFfitOptimiser (RFfitOptimiser), 88
RFfitOptimiser, 88
RFformula, 7, 14, 51, 69, 72, 74, 78, 80, 82,
INDEX

164, 169, 170, 176, 178–180, 272, 275, 363, 364
RFsp-class, 172
RFspatialDataFrame (RFsp-class), 172
RFspatialDataFrame-class (RFsp-class), 172
RFspatialGridDataFrame, 22, 63, 111, 164, 170, 172–175, 178
RFspatialGridDataFrame
(RFspatialGridDataFrame-class), 174
RFspatialGridDataFrame-class, 174
RFspatialPointsDataFrame, 22, 155, 164, 170, 172, 173, 175, 177, 178
RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class), 177
RFspatialPointsDataFrame-class, 177
RFspDataframe2conventional
(RFsp-class), 172
RFspDataframe2conventional, RFgridDataFrame-method
(RFgridDataFrame-class), 110
RFspDataframe2conventional, RFpointsDataFrame-method
(RFpointsDataFrame-class), 153
RFspDataframe2conventional, RFsp-method
(RFsp-class), 172
RFspDataframe2conventional, RFspatialGridDataFrame-method
(RFspatialGridDataFrame-class), 174
RFspDataframe2conventional, RFspatialPointsDataFrame-method
(RFspatialPointsDataFrame-class), 177
RFspDataframe2dataArray (RFsp-class), 172
RFspDataframe2dataArray, RFgridDataFrame-method
(RFgridDataFrame-class), 110
RFspDataframe2dataArray, RFsp-method
(RFsp-class), 172
RFspDataframe2dataArray, RFspatialGridDataFrame-method
(RFspatialGridDataFrame-class), 174
RFvariogram, 7, 14, 47, 57, 58, 68, 70, 73, 80, 81, 87, 97, 98, 101, 112, 119, 127, 157, 159, 165, 172, 179, 180, 181
RM, 7, 8, 10, 21, 27, 47, 59, 98, 271, 283–285, 287, 344, 374, 377
RM (RMmodels Overview), 279
RM_COVARIMATE (RMcovariate), 215
RM_DECLARE (RMdeclare), 224
RM_DEFAULT (Internal functions), 40
RM_DISTR (RRdistr), 346
RM_modelFit-class (RMMmodelFit-class), 274
RM_MULT (RMMult), 290
RM_NUGGET (RMMnugget), 296
RM_PLUS (RMPplus), 300
RM_TREND (RMrMtrend), 329
RM_USER (RMRuser), 332
RMangle, 58, 182, 182, 270, 271, 282, 328
RMaskey, 183, 184, 195, 248, 249, 280, 368, 373
RMave, 185, 185, 186, 286
RMBall, 27, 58, 187, 302, 353
RMBcw, 188, 188, 247, 248, 264, 280, 368, 373
RMBernoulli, 189, 190, 280, 339, 374
RMBessel, 52, 190, 190, 191, 223, 224, 280, 335, 356
RMBicauchy, 192, 192, 283
RMBicigeighting, 184, 193, 193, 194, 195, 196, 218, 252, 253, 283
RMBicigeightingtable, 195, 195, 196, 283, 321
RMBiendland (RMBicigeighting), 193
RMBiwm, 195, 197, 197, 198, 283, 298, 338
RMBblend, 199, 199, 206, 312, 313
RMBiwm2D, 200, 202
RMBiseg, 201, 201, 202
RMBrownresnick, 203, 203, 372, 374
RMBrownresnick2D, 204, 205, 311, 313
RMcardinalsine (RMrMwave), 335
RMCauchytm, 208, 209, 209, 233, 247
RMchoquet, 210, 210, 291, 292, 317, 318
RMcircular, 108, 211, 211, 212, 280, 368, 373
RMCmixt, 148, 212, 212, 213, 280, 368, 373
RMCov, 213, 214, 281, 284
RMCov2D (RMcov), 213
RMCovariate, 132, 215, 216, 239, 281, 333
RMCoxisham, 216, 216, 217, 286
RMCubic, 218, 218, 280, 368, 373
RMCurvature, 219, 219, 228, 230, 231, 262, 283, 286, 335
RMCutoff, 15, 17, 220, 220, 221, 281
RMdag, 222, 222, 280, 368, 373
RMDampedcos, 191, 223, 223, 224, 280
RMdeclare, 14, 224, 225
INDEX

RMdelay, 108, 226, 228, 283
RMderiv, 220, 227, 227, 228, 231, 283
RMdewijsian, 188, 189, 228, 229, 280
RMdivfree, 219, 220, 228, 230, 230, 262, 283, 286, 335
RMexaxx, 59, 231, 232
RMeascauhy, 232, 232, 233
RMetaxxa, 59, 232
RMeetaxxa (RMexaxx), 231
RMexponential, 235, 235, 236, 280, 284, 289, 306
RMfbm, 15, 237, 237, 247, 248, 264, 271, 372
RMfixcov, 46, 132, 213, 216, 238, 238, 239, 281, 333
RMfixed, 90, 240
RMflypow, 240, 240, 241, 248, 280
RMformula, 51, 69, 72, 74, 78, 80, 82, 105, 117, 121, 122, 125, 156, 158, 163, 179
RMformula (RFormula), 90
RMfracdiff, 242, 242, 280
RMfracgauss, 243, 243, 280
RMgauss, 244, 244, 245, 251–253, 270, 289, 306, 320, 321, 337, 338, 342, 349
RMgencauhy, 188, 189, 207–209, 221, 245, 245, 246, 263, 270, 285, 368, 373
RMgenfbm, 188, 189, 237, 238, 240, 241, 247, 247, 247, 280
RMgenigeing, 184, 193, 195, 248, 248, 249, 251–253, 280, 368, 373
RMgennss, 250, 250, 286, 296
RMgneiting, 184, 195, 245, 248, 249, 251, 251, 252, 253, 270, 368, 373
RMgneitingdiff, 252, 252, 253, 280
RMhandcock (RMwhittlematern), 336
RMhyperbolic, 208, 254, 254, 280, 338
RMIaco, 255, 255, 280
RMid, 59, 256, 256, 305, 328
RMinmodel, 59, 257, 257, 328
RMINtern, 258
RMintexp, 259, 259, 281
RMINtrinsic, 15, 17, 260, 260, 281
RMIbessel (RMbessel), 190
RMkbbessel (RMwhittlematern), 336
RMkolmgorov, 261, 262, 283
RMLgd, 233, 246, 262, 262, 263, 280
RMLsfbm, 189, 264, 264, 280
RMm2r, 57, 58, 201, 202, 204, 359, 374
RMm2r (Stokrorb's Functions), 371
RMm3b, 57, 58, 204, 374
RMm3b (Stokrorb's Functions), 371
RMMa, 265, 265
RMMstein, 266, 266, 267, 286
RMMaternal, 35, 197, 245, 270, 294, 336, 368, 373
RMMaternal (RMwhittlematern), 336
RMMatrix, 268, 268, 283, 284, 316, 331
RMMixed (RMINtern), 258
RMMmodel-class, 272
RMMmodelFit, 161
RMMmodelFit-class, 274
RMMmodelgenerator-class, 21, 276
RMMmodels, 7, 48, 52, 97, 109, 279, 283–285, 287, 368, 369, 374, 377
RMMmodels (RMMmodel), 270
RMMmodels Overview, 279
RMMmodelsAdvanced, 13, 36, 66, 271, 279, 280, 284, 285, 287
RMMmodelsAuxiliary, 271, 279, 283
RMMmodelsAuxiliary (Others), 58
RMMmodelsMultivariate, 279, 283, 377
RMMmodelsNonstationary, 279, 281, 285
RMMmodelsSpaceTime, 279, 281
RMMmodelsSpaceTime (RMMmodelsSpacetime), 286
RMMmodelsSpacetime, 286
RMMmodelsSpacetime, 286
RMMmodelsSphere (Spherical models), 367
RMMmodelsTailCorrelation, 281
RMMmodelsTailCorrelation (Tail Correlation Functions), 373
INDEX

RPbrmixed, 13, 14, 53, 134
RPbrmixed (Brown-Resnick-Specific), 10
RPbrorig, 13, 14, 53
RPbrorig (Brown-Resnick-Specific), 10
RPbrownresnick, 12, 53, 201, 202, 342
RPbrownresnick (BrownResnick), 12
RPbrshifted, 13, 14, 53
RPbrshifted (Brown-Resnick-Specific), 10
RPchi2, 340, 342
RPCirculant, 15, 32, 129, 134
RPCirculant (Circulant Embedding), 15
RPCoins, 20, 32, 39, 130, 343
RPCoins (Coins), 19
RCuttoff, 32
RCuttoff (Circulant Embedding), 15
RDirect, 25, 32, 146, 359, 370
RDirect (Square root), 369
RGauss, 12, 14, 16, 29, 30, 32, 54, 97, 140, 168, 172, 245, 339, 340, 341, 342, 345, 349, 360
RHyperplane, 20, 32, 39, 130
RHyperplane (Hyperplane), 37
RPIntrinsic, 32, 33, 260, 261
RPIntrinsic (Circulant Embedding), 15
RPLoggaussnormed
(Brown-Resnick-Specific), 10
RPmaxstable, 55
RPmaxstable (Max-stable random fields), 53
RPmaxstableAdvanced (Max-stable random fields, advanced), 55
RPmodel (RPprocess), 343
RPmodels (RPprocess), 343
RPmpmpplus (RMintern), 258
RPmult (RMintern), 258
RPNugget, 32, 33, 145, 296, 297
RPNugget (Independent Variables), 38
RPpopitz, 53, 342
RPpopitz (Extremal t), 28
RPlus (RMintern), 258
RPPoisson, 36, 331, 342
RPprocess, 343
RPprocesses (RPprocess), 343
RPS (RMintern), 258
RPschlather, 53, 201, 202, 314, 342
RPschlather (ExtremalGaussian), 29
RPsequential, 32, 33, 130, 145, 358, 370
RPsequential (Sequential), 358
RPsmith, 53, 144, 168, 277, 287, 288, 372
RPsmith (Smith), 359
RPspecific, 32, 33, 365
RPspecific (Specific), 364
RPspectral, 20, 32, 33, 39, 130, 147, 376
RPspectral (Spectral), 366
Rpt, 342, 344
Rptb, 20, 32, 33, 39, 130, 134, 147, 325, 326, 365, 367
Rptb (Tbm), 374
Rptrend (RMintern), 258
RR, 10, 21, 36, 47, 58, 77, 98, 271, 277, 279, 317, 344, 347, 351
RR (Distribution Families), 26
RRarcsqrt (RMintern), 258
RRdetermine, 26, 346, 364
RRdistr, 26, 36, 76, 346, 346, 347, 349–352, 354
RRgauss, 27, 77, 245, 342, 346, 348, 349, 350, 352, 354
RRloc, 27, 349, 350
RRmcmc, 350, 351
RRmodel, 76
RRmodel (Distribution Families), 26
RRmodels, 36, 278
RRmodels (Distribution Families), 26
RRrectangular, 129, 135, 136, 351, 352, 360
RRsetdistr (RMintern), 258
RRsign (RMsign), 316
RRspheric, 27, 353, 353, 354
RRunif, 27, 349, 354, 354
S02, 60, 355
S10, 43, 60, 232, 308, 309, 355
SBS14, 60, 356
scale, 32
Schlather, 374
Schoenberg’s representation
(RMchoquet), 210
ScreenDevice (Internal functions), 40
seq, 112
Sequential, 358
set.seed, 132, 160
show, RFeuvvario metod
(RFeuvvario-class), 77
show, Rffit-method (RFFit-class), 84
show, RgridDataFrame-method
(RgridDataFrame-class), 110
showRFpointsDataFrame-method (RFpointsDataFrame-class), 153
showRFspatialDataFrame-method (RFspatialDataFrame-class), 174
showRFspatialPointsDataFrame-method (RFspatialPointsDataFrame-class), 177
showRMmodel-method (RMmodel-class), 272
showRMmodelFit-method (RMmodelFit-class), 274
showRMmodelgenerator-method (RMmodelgenerator-class), 276
showManpages (Internal functions), 40
sin (Mathematical C functions), 48
sin, RMmodel-method (Mathematical C functions), 48
sine power function (RMsinepower), 317
sinh (Mathematical C functions), 48
sinh, RMmodel-method (Mathematical C functions), 48
Smith, 359
Sobolev (RMwhittlematern), 336
soil, 7, 113, 361
sp2RF, 8, 173, 174, 176, 177, 179, 363, 366
space-time (RMmodelsSpacetime), 286
spam, 32, 33
Spatial, 173, 178
SpatialGridDataFrame, 110, 174, 175
SpatialPoints, 177, 178
SpatialPointsDataFrame, 154, 177, 178
spConform, 83
spConform=FALSE, 8
spConform=TRUE, 8
Specific, 364
Spectral, 366
sphere (Spherical models), 367
Spherical models, 279, 367
spherical models, 7, 23, 319
spherical models (Spherical models), 367
sqrt.screen, 141
sqrt (Mathematical C functions), 48
sqrt, RMmodel-method (Mathematical C functions), 48
Square root, 369
SS12, 43, 60, 371
StartExample (Internal functions), 40
stationary max-stable random fields, 162, 168
stationary Poisson fields, 162, 167
str, 8, 273
str.RMmodel (RMmodel-class), 272
Strokorb’s Functions, 371
summary, 8
summary, RFEmpVariog-method (RFEmpVariog-class), 77
summary, RFFit-method (RFFit-class), 84
summary, RFfit-method (RFfit-class), 84
summary, RFsp-method (RFsp-class), 172
summary, RMmodelFit-method (RMmodelFit-class), 274
summary.crossvalidate (RFcrossvalidate), 73
summary.RF_empVariog (RFEmpVariog-class), 77
summary.RF_fit (RFVariog-class), 84
summary.RM_modelFit (RMmodelFit-class), 274
Sweave, 131
t field, 344
t fields, 167
Tail Correlation Functions, 373
tail correlation functions, 279
Tail correlation functions (Tail Correlation Functions), 373
tail correlation functions, 47
tail correlation functions (Tail Correlation Functions), 373
tan (Mathematical C functions), 48
tan, RMmodel-method (Mathematical C functions), 48
tanh (Mathematical C functions), 48
tanh, RMmodel-method (Mathematical C functions), 48
Tbm, 374
tbmdim (Changes), 14
tcf (Tail Correlation Functions), 373
trend, 47
trend (RMtrend), 329
Trend Modelling, 377
trend modelling, 271, 279
trend modelling (Trend Modelling), 377
trunc (Mathematical C functions), 48
trunc, RMmodel-method (Mathematical C functions), 48
truncated power function (RMaskey), 183
INDEX

unif, 354
user, 47

variab_units(Changings), 14
variance,RFgridDataFrame-method
(RFgridDataFrame-class), 110
variance,RFpointsDataFrame-method
(RFpointsDataFrame-class), 153
variance,RFsp-method (RFsp-class), 172
variance,RFspatialGridDataFrame-method
(RFspatialGridDataFrame-class), 174
variance,RFspatialPointsDataFrame-method
(RFspatialPointsDataFrame-class), 177
Variogram (Obsolete Functions Version 2), 55

weather, 7, 43, 60, 75, 83, 162, 378
which.submodels, 103
whittle-matern, 285
whittle-matern (RMwhittlematern), 336

zenit (Coordinate systems), 22