Package ‘RandomFields’

January 9, 2016

Version 3.1.8

Title Simulation and Analysis of Random Fields

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

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

LinkingTo RandomFieldsUtils

Depends R (>= 3.0.2), sp, RandomFieldsUtils (>= 0.0.14)

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

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-01-09 13:56:44

R topics documented:

RandomFields-package .................................................. 6
Brown-Resnick-Specific .............................................. 10
BrownResnick ............................................................... 12
Changings .................................................................. 13
Circulant Embedding .................................................... 14
Coins ............................................................................ 18
Constants ...................................................................... 19
conventional2RFspDataFrame ........................................ 20
Coordinate systems ............................................... 21
Distribution Families ............................................ 24
Extremal t ......................................................... 26
ExtremalGaussian .................................................. 27
fitgauss ............................................................ 29
GaussianFields ....................................................... 30
GSPSJ06 .............................................................. 33
Hierarchical Modelling ............................................ 34
Hyperplane .......................................................... 35
Independent Variables ............................................. 37
Internal functions .................................................. 38
jss14 ................................................................. 40
Major Revisions .................................................... 43
Mathematical C functions ......................................... 45
Max-stable random fields ........................................ 49
Max-stable random fields, advanced ............................ 51
Obsolete Functions .................................................. 53
Others ................................................................. 55
papers ............................................................... 56
plot-method ........................................................ 57
PrintModelList ....................................................... 63
RFboxcox ............................................................ 64
RFcov ............................................................... 66
RFcrossvalidate .................................................... 69
RFdistr .............................................................. 71
RFempiricalvariogram ........................................... 72
RFempVariog-class ............................................... 75
RFfit ................................................................. 77
RFfit-class ........................................................ 81
RFfitoptimiser ...................................................... 84
RFformula ........................................................... 86
RFfractaldim ......................................................... 89
RFfunction .......................................................... 92
RfgetMethodName .................................................. 93
RFgetModel .......................................................... 97
RFgetModelInfo ..................................................... 99
RFgetModelNames .................................................. 101
RFgridDataFrame-class .......................................... 104
RFgui ............................................................... 106
RFHurst ............................................................. 108
RFinterpolate ....................................................... 111
RFlinearpart ......................................................... 114
RFloglikelihood .................................................... 116
RFoldstyle .......................................................... 119
RFoptions ........................................................... 120
RFoptionsAdvanced ............................................... 141
RFpar ............................................................... 143
R topics documented:

<table>
<thead>
<tr>
<th>Class/Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFpointsDataFrame-class</td>
<td>144</td>
</tr>
<tr>
<td>RFratiotest</td>
<td>146</td>
</tr>
<tr>
<td>RFsimulate</td>
<td>148</td>
</tr>
<tr>
<td>RFsimulate.more.examples</td>
<td>152</td>
</tr>
<tr>
<td>RFsimulate.sophisticiated.examples</td>
<td>152</td>
</tr>
<tr>
<td>RFsimulateAdvanced</td>
<td>153</td>
</tr>
<tr>
<td>RFsp-class</td>
<td>158</td>
</tr>
<tr>
<td>RFsp2conventional</td>
<td>159</td>
</tr>
<tr>
<td>RFspatialGridDataFrame-class</td>
<td>160</td>
</tr>
<tr>
<td>RFspatialPointsDataFrame-class</td>
<td>163</td>
</tr>
<tr>
<td>RMangle</td>
<td>165</td>
</tr>
<tr>
<td>RMaskey</td>
<td>166</td>
</tr>
<tr>
<td>RMave</td>
<td>168</td>
</tr>
<tr>
<td>RMBall</td>
<td>170</td>
</tr>
<tr>
<td>RMBcw</td>
<td>170</td>
</tr>
<tr>
<td>RMberoulli</td>
<td>172</td>
</tr>
<tr>
<td>RMBessel</td>
<td>173</td>
</tr>
<tr>
<td>RMBignetting</td>
<td>175</td>
</tr>
<tr>
<td>RMBiwm</td>
<td>177</td>
</tr>
<tr>
<td>RMBbr2bg</td>
<td>179</td>
</tr>
<tr>
<td>RMBbr2eg</td>
<td>180</td>
</tr>
<tr>
<td>RMBrownresnick</td>
<td>182</td>
</tr>
<tr>
<td>RMCauchy</td>
<td>183</td>
</tr>
<tr>
<td>RMCauchytbm</td>
<td>184</td>
</tr>
<tr>
<td>RMCchoquet</td>
<td>186</td>
</tr>
<tr>
<td>RMcircular</td>
<td>187</td>
</tr>
<tr>
<td>RMconstant</td>
<td>188</td>
</tr>
<tr>
<td>RMCcoord</td>
<td>189</td>
</tr>
<tr>
<td>RMcovariate</td>
<td>190</td>
</tr>
<tr>
<td>RMcoxisham</td>
<td>191</td>
</tr>
<tr>
<td>RMcubic</td>
<td>193</td>
</tr>
<tr>
<td>RMCurffree</td>
<td>194</td>
</tr>
<tr>
<td>RMCutoff</td>
<td>195</td>
</tr>
<tr>
<td>RMDagum</td>
<td>197</td>
</tr>
<tr>
<td>RMDampedcos</td>
<td>198</td>
</tr>
<tr>
<td>RMDelay</td>
<td>199</td>
</tr>
<tr>
<td>RMDewijsian</td>
<td>201</td>
</tr>
<tr>
<td>RMDivfree</td>
<td>202</td>
</tr>
<tr>
<td>RMeaxxa</td>
<td>203</td>
</tr>
<tr>
<td>RMepscauchy</td>
<td>204</td>
</tr>
<tr>
<td>RMexp</td>
<td>206</td>
</tr>
<tr>
<td>RMexponential</td>
<td>207</td>
</tr>
<tr>
<td>RMfbm</td>
<td>209</td>
</tr>
<tr>
<td>RMfixxcov</td>
<td>210</td>
</tr>
<tr>
<td>RMfixed</td>
<td>212</td>
</tr>
<tr>
<td>RMflatpower</td>
<td>212</td>
</tr>
<tr>
<td>RMfractdiff</td>
<td>214</td>
</tr>
<tr>
<td>RMfractgauss</td>
<td>215</td>
</tr>
<tr>
<td>R Topics Documented</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>RMgauss</td>
<td>216</td>
</tr>
<tr>
<td>RMcencauchy</td>
<td>217</td>
</tr>
<tr>
<td>RMgenfbm</td>
<td>219</td>
</tr>
<tr>
<td>RMgengneiting</td>
<td>220</td>
</tr>
<tr>
<td>RMgennsst</td>
<td>222</td>
</tr>
<tr>
<td>RMgneiting</td>
<td>223</td>
</tr>
<tr>
<td>RMgneitingdiff</td>
<td>225</td>
</tr>
<tr>
<td>RMhyperbolic</td>
<td>226</td>
</tr>
<tr>
<td>RMiaco</td>
<td>227</td>
</tr>
<tr>
<td>RMid</td>
<td>228</td>
</tr>
<tr>
<td>RMintern</td>
<td>229</td>
</tr>
<tr>
<td>RMintexp</td>
<td>230</td>
</tr>
<tr>
<td>RMintrinsic</td>
<td>231</td>
</tr>
<tr>
<td>RMkolmogorov</td>
<td>233</td>
</tr>
<tr>
<td>RMLgd</td>
<td>234</td>
</tr>
<tr>
<td>RMma</td>
<td>235</td>
</tr>
<tr>
<td>RMmastein</td>
<td>236</td>
</tr>
<tr>
<td>RMmatrix</td>
<td>238</td>
</tr>
<tr>
<td>RMmodel</td>
<td>239</td>
</tr>
<tr>
<td>RMmodel-class</td>
<td>241</td>
</tr>
<tr>
<td>RMmodelFit-class</td>
<td>244</td>
</tr>
<tr>
<td>RMmodelgenerator-class</td>
<td>245</td>
</tr>
<tr>
<td>RMmodels Overview</td>
<td>248</td>
</tr>
<tr>
<td>RMmodelsAdvanced</td>
<td>249</td>
</tr>
<tr>
<td>RMmodelsMultivariate</td>
<td>252</td>
</tr>
<tr>
<td>RMmodelsNonstationary</td>
<td>254</td>
</tr>
<tr>
<td>RMmodelsSpacetime</td>
<td>255</td>
</tr>
<tr>
<td>RMmppplus</td>
<td>256</td>
</tr>
<tr>
<td>RMmqam</td>
<td>257</td>
</tr>
<tr>
<td>RMmult</td>
<td>258</td>
</tr>
<tr>
<td>RMmultiquad</td>
<td>259</td>
</tr>
<tr>
<td>RMnatsc</td>
<td>261</td>
</tr>
<tr>
<td>RMnonstwm</td>
<td>262</td>
</tr>
<tr>
<td>RMnsst</td>
<td>263</td>
</tr>
<tr>
<td>RMnugget</td>
<td>264</td>
</tr>
<tr>
<td>RMParswm</td>
<td>266</td>
</tr>
<tr>
<td>RMPenta</td>
<td>267</td>
</tr>
<tr>
<td>RMPplus</td>
<td>268</td>
</tr>
<tr>
<td>RMPolygone</td>
<td>269</td>
</tr>
<tr>
<td>RMPolynome</td>
<td>270</td>
</tr>
<tr>
<td>RMPower</td>
<td>271</td>
</tr>
<tr>
<td>RMProd</td>
<td>273</td>
</tr>
<tr>
<td>RMQam</td>
<td>274</td>
</tr>
<tr>
<td>RMQexp</td>
<td>275</td>
</tr>
<tr>
<td>RMRational</td>
<td>276</td>
</tr>
<tr>
<td>RMRotat</td>
<td>277</td>
</tr>
<tr>
<td>RMS</td>
<td>278</td>
</tr>
<tr>
<td>RMschlather</td>
<td>279</td>
</tr>
</tbody>
</table>


R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMschur</td>
<td>281</td>
</tr>
<tr>
<td>RMsign</td>
<td>282</td>
</tr>
<tr>
<td>RMsinepower</td>
<td>283</td>
</tr>
<tr>
<td>RMspheric</td>
<td>284</td>
</tr>
<tr>
<td>RMstable</td>
<td>286</td>
</tr>
<tr>
<td>RMstein</td>
<td>287</td>
</tr>
<tr>
<td>RMstp</td>
<td>288</td>
</tr>
<tr>
<td>RMsum</td>
<td>290</td>
</tr>
<tr>
<td>RMTbm</td>
<td>291</td>
</tr>
<tr>
<td>RMtrafo</td>
<td>293</td>
</tr>
<tr>
<td>RMtrend</td>
<td>295</td>
</tr>
<tr>
<td>RMtruncksupport</td>
<td>297</td>
</tr>
<tr>
<td>RMuser</td>
<td>298</td>
</tr>
<tr>
<td>RMvector</td>
<td>300</td>
</tr>
<tr>
<td>RMwave</td>
<td>301</td>
</tr>
<tr>
<td>RMwhittlematern</td>
<td>302</td>
</tr>
<tr>
<td>RPBernoulli</td>
<td>305</td>
</tr>
<tr>
<td>RPCI2</td>
<td>306</td>
</tr>
<tr>
<td>RPgauss</td>
<td>307</td>
</tr>
<tr>
<td>RPPoisson</td>
<td>308</td>
</tr>
<tr>
<td>RPProcess</td>
<td>309</td>
</tr>
<tr>
<td>RPT</td>
<td>310</td>
</tr>
<tr>
<td>RRdeterm</td>
<td>312</td>
</tr>
<tr>
<td>RRdistr</td>
<td>312</td>
</tr>
<tr>
<td>RRgauss</td>
<td>314</td>
</tr>
<tr>
<td>RRloc</td>
<td>315</td>
</tr>
<tr>
<td>RRmcmc</td>
<td>316</td>
</tr>
<tr>
<td>RRRectangular</td>
<td>317</td>
</tr>
<tr>
<td>RRspheric</td>
<td>319</td>
</tr>
<tr>
<td>RRuniform</td>
<td>320</td>
</tr>
<tr>
<td>S02</td>
<td>321</td>
</tr>
<tr>
<td>S10</td>
<td>321</td>
</tr>
<tr>
<td>SBS14</td>
<td>322</td>
</tr>
<tr>
<td>Smith</td>
<td>324</td>
</tr>
<tr>
<td>soil</td>
<td>325</td>
</tr>
<tr>
<td>sp2RF</td>
<td>327</td>
</tr>
<tr>
<td>Specific</td>
<td>329</td>
</tr>
<tr>
<td>Spectral</td>
<td>330</td>
</tr>
<tr>
<td>Spherical models</td>
<td>332</td>
</tr>
<tr>
<td>Square roots</td>
<td>334</td>
</tr>
<tr>
<td>SS12</td>
<td>336</td>
</tr>
<tr>
<td>Stokorb's Functions</td>
<td>336</td>
</tr>
<tr>
<td>Tail Correlation Functions</td>
<td>338</td>
</tr>
<tr>
<td>Tbm</td>
<td>339</td>
</tr>
<tr>
<td>Trend Modelling</td>
<td>342</td>
</tr>
<tr>
<td>weather</td>
<td>343</td>
</tr>
</tbody>
</table>

Index 345
Description

The package RandomFields offers various tools for

1. **model estimation (ML) and inference (tests)** for regionalized variables and data analysis,
2. **simulation** of different kinds of random fields, including
   - multivariate, spatial, spatio-temporal, and non-stationary Gaussian random fields,
   - Poisson fields, binary fields, Chi2 fields, t fields and
   - max-stable fields.

It can also deal with non-stationarity and anisotropy of these processes and conditional simulation (for Gaussian random fields, currently).

See [http://ms.math.uni-mannheim.de/de/publications/software/](http://ms.math.uni-mannheim.de/de/publications/software/) for **intermediate updates**.

Details

The following features are provided by the package:

1. **Bayesian Modelling**
   - See [Bayesian Modelling](#) for an introduction to hierarchical modelling
2. **Coordinate systems**
   - Cartesian, earth and spherical coordinates are recognized, see [coordinate systems](#) for details.
   - a list of valid models is given by [spherical models](#).
3. **Data and example studies**: Some data sets and published code are provided to illustrate the syntax and structure of the package functions.
   - **soil**: soil physical data
   - **weather**: UWME weather data
   - **papers**: code used in the papers published by the author(s)
4. **Estimation of parameters (for second-order random fields)**
   - **Rffit**: general function for estimating parameters; (for Gaussian random fields)
   - **RFhurst**: estimation of the Hurst parameter
   - **RFFractaldim**: estimation of the fractal dimension
   - **RFempiricalvariogram**: calculates the empirical variogram
5. **Graphics**
   - Fitting a covariance function manually **RFgui**
   - the generic function **plot**
   - global graphical parameters with **RFpar**
6. **Inference (for Gaussian random fields)**
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.
- To apply the offered package procedures to mixed models—e.g. appearing in genetical data analysis—see RFformula.
- 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.
Update

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

Contributions

• Contributions to version 3.0 and following:
  Felix Ballani (TU Bergakademie Freiberg; Poisson Polygons, 2014)
  Daphne Boecker (Univ. Goettingen; RFgui, 2011)
  Katharina Burmeister (Univ. Goettingen; testing, 2012)
  Sebastian Engelke (Univ. Goettingen; RFempiricalvariogram, 2011-12)
  Sebastian Gross (Univ. Goettingen; tilde formulae, 2011)
  Alexander Malinowski (Univ. Mannheim; S3, S4 classes 2011-13)
  Juliane Manitz (Univ. Goettingen; testing, 2012)
  Johannes Martini (Univ. Goettingen; RFempiricalvariogram, 2011-12)
  Ulrike Ober (Univ. Goettingen; help pages, testing, 2011-12)
  Marco Oesting (Univ. Mannheim; Brown-Resnick processes, Kriging, Trend, 2011-13)
  Paulo Ribeiro (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 Riberio : comments on Version 1
Martin Maechler : advice for Version 1

Financial support

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

The following packages enable further choices for the optimizer (instead of optim) in RandomFields: optimx, soma, GenSA, minqa, pso, DEoptim, nloptr, RColorBrewer, colorspace

Author(s)

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

References

- 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
# seed
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))
data <- matrix(nrow=nrow(xy), as.vector(z))[pts, ]
data <- cbind(xy[pts, ], data)
plot(z, data)

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

## show a kriged field based on the estimated parameters
kriged <- RFInterpolate(fit, x, x, data=data)
plot(kriged, data)
```
Simulation methods for Brown-Resnick processes

Description

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

Usage

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

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

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

Arguments

phi object of class RMmodel; specifies the covariance model to be simulated.
tcf the extremal correlation function; either phi or tcf must be given.
xi, mu, s the shape parameter, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.
lambda positive constant factor in the intensity of the Poisson point 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 or matrix of values in [0, 1] with odd length (odd number of rows and columns, respectively). Each value represents the portion of processes whose maximum is located at a specific location on a grid taken into account for the simulation of the shape function in the M3 representation. The center of areamat represents the value for the origin, the other entries belong to the corresponding locations on a 1D or 2D grid. areamat can be used for dimensions 1 and 2 only; can be optimized by setting optim_mixed if unknown. Default value is 1.

meshsize, vertnumber, optim_mixed, optim_mixed_tol, optim_mixed_maxpo, variobound further arguments for simulation via the mixed moving maxima (M3) representation; see RFOptions

Details

The argument \( xi \) is always a number, i.e. \( \xi \) is constant in space. In contrast, \( \mu \) and \( s \) might be constant numerical value or given a RMmodel, in particular by a RMTrend model.

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

\[
Z(x) = \max_{i=1}^{\infty} X_i \exp(W_i(x) - \gamma),
\]
where the $X_i$ are the points of a Poisson point process on the positive real half-axis with intensity $x^{-2}dx$, $W_i \sim W$ are iid centered Gaussian processes with stationary increments and variogram $\gamma$ given by `model`. The functions correspond to the following ways of simulation:

`RPbrorig` simulation via using the original definition (method 0 in Oesting et al., 2012)
`RPbrshifted` simulation using a random shift (similar to method 1 and 2)
`RPbrmixed` simulation using M3 representation (method 4)

**Value**

The functions return an object of class `RMmodel`.

**Note**

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

**Author(s)**

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

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

**References**


**See Also**

`RPbrownresnick`, `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

## currently does not work
```
BrownResnick  

Brown-Resnick process

Description

Rpbrownresnick defines a Brown-Resnick process.

Usage

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

Arguments

phi  
  specifies the covariance model or variogram, see RMmodel and RMmodelsAdvanced.

tcf  
  the extremal correlation function; either phi or tcf must be given.

xi, mu, s  
  the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.

Details

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

The functions Rpborig, Rpbshifted and Rpbmixed perform the simulation of a Brown-Resnick process, which is defined by

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

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

For simulation, internally, one of the methods Rpborig, Rpbshifted and Rpbmixed is chosen automatically.

Note

Advanced options are maxpoints and max_gauss, see RFoptions.

Further advanced options related to the simulation methods Rpborig, Rpbshifted and Rpbmixed can be found in the paragraph ‘Specific method options for Brown-Resnick Fields’ in RFoptions.

Author(s)

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


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 'maxstamableAdvanced'
```

Description

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

See Also

MajorRevisions, RandomFields

Examples

### no examples given

---

### Circulant Embedding methods

**Description**

Circulant embedding is a fast simulation method for stationary (possibly anisotropic) Gaussian random fields on regular grids based on Fourier transformations. It is guaranteed to be an exact method for covariance functions with finite support, e.g. the spherical model. The method is admissible for any dimension apart from memory restrictions.

The simulation is performed on a torus which represents the bended grid. To remove wrong dependencies 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 Rcpirculant 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 Rcpirculant method, for further variogram models, e.g. RmFbm.

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

Here multiplicative models are not allowed (yet).

For details see RmIntrinsic.

**Usage**

```R
Rcircularant(\phi, boxcox, force, mmin, strategy, maxGB, maxmem, tolIm, tolRe, trials, useprimes, dependent, approx_step, approx_maxgrid)

Rcutoff(\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
is 32 (=4 * sizeof(double)) time as large (factor 2 is needed as complex numbers must be considered for calculating the fft of the covariance matrix; another factor 2 is needed for storing the simulated result).
The value of maxmem must be at least $2^d$ times as large as the number of points to be simulated. Here $d$ is the space dimension. In some cases even much larger. Note that maxmem can be used to control the automatic choice of the simulation algorithm. Namely, in case of huge circulant matrices, other simulation methods (TBM) might be faster and might be preferred by the user.
If this argument is set then maxGB is set to Inf.
Default: MAXINT
**tolIm**
If the modulus of the imaginary part is less than `tolIm` then the eigenvalue is always considered as real (independently of the value of `force`).
Default: 1E-3

**tolRe**
Eigenvalues between `tolRe` and 0 are always considered as 0 and set 0 (independently of the value of `force`).
Default: -1E-7

**trials**
Integer. A larger circulant matrix is likely to make more eigenvalues non-negative. If at least one of the thresholds `tolRe` and `tolIm` are missed then the matrix size is doubled according to `strategy`, and the matrix is checked again. This procedure is repeated up to `trials` - 1 times. If there are still negative eigenvalues, the simulation method fails if `force=FALSE`.
Default: 3

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

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

**approx_step**
Real value. It gives the grid size of the approximating grid in case circulant embedding is used although the points do not lie on a grid. It defaults to `maxmem`.

**approx_maxgrid**
It defaults to `maxmem`.

**diameter**
See `RMcutoff` or `RMintrinsic`

**a**
See `RMcutoff`

**rawR**
See `RMintrinsic`

**Details**
Here, the algorithms by Dietrich and Newsam are implemented.

**Value**
an object of class `RMmodel`

**Author(s)**
Martin Schlather, <schlather@math.uni-mannheim.de>
References

Circulant Embedding


Cutoff and Intrinsic


See Also

Gaussian, RP

Examples

```r
RFoptions(seed=8) # *ANY* simulation will have the random seed 8; set
                 RFoptions(seed=NA) to make them all random again

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

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

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

RPcoins(phi, shape, boxcox, intensity, method)

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

Arguments

- **phi**: object of class RMmodel; specifies the covariance function of the Poisson process; either phi or shape must be given.
- **shape**: object of class RMmodel; specifies the function which is attached to the Poisson points; note that this is not the covariance function of the simulated random field.
- **boxcox**: the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. see RFboxcox for Details.
- **intensity**: positive number, intensity of the underlying Poisson point process.
- **method**: integer. Default is the value 0 which addresses the current standard procedure. There might be further methods implemented mainly for internal purposes.

Value

RPcoins returns an object of class RMmodel

Author(s)

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

References


See Also

Gaussian, RP, RHyperplane, 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
```

---

**Constants**  
*Constants used in RandomFields (RC constants)*

**Description**

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

**Value**

```r
RC_TYPENAMES = c("tail correlation function", "positive definite", "variogram", "negative definite")
RC_DOMAIN_NAMES = c("single variable", "kernel", "framework dependent", "mismatch")
RC_ISONAMES = c("isotropic", "space-isotropic", "zero-space-isotropic", "vector-isotropic", "symmetric")
RC_MONOTONE_NAMES = c("mismatch in monotonicity", "submodel dependent monotonicity")

RC_ISOTROPIC gives the numerical code for option "isotropic"
RC_SPACEISOTROPIC gives the numerical code for option "space-isotropic"
RC_CARTESIANCOORD gives the numerical code for option "cartesian system"
RC_GNOMONIC_PROJ gives the numerical code for the gnomonic projection, see also `zenit` in `RFoptions`.
RC_ORTHOGRAPHIC_PROJ gives the numerical code for the orthographic projection, see also `zenit` in `RFoptions`.
RC_EARTH_COORDS gives the numerical code for option "earth coordinates"
RC_SPHERICAL_COORDS gives the numerical code for option "earth coordinates"
RC_OPTIMISER_NAMES and RC_NLOPTR_NAMES give the names for the options optimiser and algorithm, respectively, `RFfitoptimiser`.
RC_LIKELIHOOD_NAMES = c("auto", "full", "composite", "tesselation") gives the names of the ML variants: (i) internal choice according to the number of data, (ii) full likelihood, (iii) (pairwise) composite likelihood, and (iv) composite likelihood based on a tesselation of the space.
```
Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software

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]
RFgetModelNames(isotropy=RC_ISO_NAMES[RC_ISOTROPIC:RC_CARTESIAN_COORD + 1])

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)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>array; of dimension c(vdim, space-time-dim, n); contains the values of the random fields</td>
</tr>
<tr>
<td>coords</td>
<td>matrix of coordinates</td>
</tr>
<tr>
<td>gridTopology</td>
<td>3-row-matrix or of class GridTopology; specifies the grid vectors; either coords or gridTopology must be NULL</td>
</tr>
<tr>
<td>n</td>
<td>number of iid copies of the random fields, default is 1</td>
</tr>
<tr>
<td>vdim</td>
<td>number of dimensions of the values of the random field, default is 1</td>
</tr>
<tr>
<td>T</td>
<td>time component if any. The length of the temporal grid is needed by as.array if the spatial locations are randomly scattered.</td>
</tr>
<tr>
<td>vdim.Close_together</td>
<td>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”.</td>
</tr>
</tbody>
</table>
Coordinate systems

Value

object of class \code{RFspatialGridDataFrame}, \code{RFspatialPointsDataFrame}, \code{RFgridDataFrame} or \code{RFpointsDataFrame}

Author(s)

Alexander Malinowski \textless{}Alexander.Malinowski@web.de\textgreater{}

Examples

\begin{verbatim}
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)
\end{verbatim}

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 an 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.


Coordinate systems

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"

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

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

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

1. "keep"
   The coord_system is kept (except an explicit transformation is given, see RMtrafo.
   Note that some classes of models, e.g. completely monotone functions and compactly
   supported covariance models with range less than π are valid models on a sphere. In this
   case the models are considered as models on the sphere. See spherical models for lists.
2. "cartesian"
   If coord_system is "earth" the coordinates are transformed to cartesian coordinates
   before any model is considered.
3. "orthographic", "genomic"
   If coord_system is "earth" the locations are projected to a plane before any model is
   considered.
Default: "keep"

new_coordunits internal and should not be set by the user.
Default: ""

polar_coord logical. If FALSE the spherical coordinates agree with the earth coordinate parametri-
sation, except that we radians are used for spherical coordinates instead of degrees for the earth
coordinates.
   If TRUE the spherical coordinates signify polar coordinates.
Default: FALSE

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

xyz_notation logical or NA. Used by RMuser only.
  NA : automatic choice (if possible)
  false : notation (x, y) should not be understood as as kernel definition, not as xyz notation
  true : xyz notation used

  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)

Author(s)

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

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

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) models

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

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

## now comparable the scale chosen sucht that the covariance
## values are comparable to those int the cartesian case
RFcov(RMRS(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:
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 comparable to cartesian case
## here the (standard) units are [km]
RFcov(RMRS(model, s= 6350 / 180 * pi), x, y, grid=FALSE,
    coord_sys="earth", new_coord_sys="orthographic")

## as above, but in miles
RFcov(RMRS(model, s= 3750 / 180 * pi), x, y, grid=FALSE,
    coord_sys="earth", new_coord_sys="orthographic",
    new_coordunits="miles")

Distribution Families  Distribution families (RR commands)

Description

Distribution families to specify random parameters in the model definition.
Details

See Bayesian Modelling for some 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 realisation of the random parameters are performed. See the examples below.

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

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

Implemented models

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

Note

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

Note

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

Author(s)

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

See Also

`RC`, `RF`, `RM`, `RP`, `R. Other models, RFDistr, RMModelGenerator`

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(1))
```
for (i in 1:4) {
  RFoptions(seed = i)
  # each leads to a simulation with a different scale parameter
  plot(model)  ## random
  plot(RFsimulate(model, x=seq(0,10,0.1)))
  readline("press return")
}

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

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

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

---

**Extremal t**  

**Extremal t process**

---

**Description**

RPopitz defines an extremal t process.

**Usage**

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

**Arguments**

- **phi**: an RModel; covariance model for a standardized Gaussian random fields, or the field itself.
- **xi, mu, s**: the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.
- **alpha**: originally referred to the \(\alpha\)-Frechet marginal distribution, see the original literature for 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 value or given a `RMmodel`, in particular by a `RMtrend` model. The default values of \( \mu \) and \( s \) are 1 and \( z\xi \), respectively.

**Author(s)**

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

**References**


**See Also**

`RMmodel`, `RPGauss`, `maxstable`, `maxstableAdvanced`

**Examples**

```r
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)
model <- RPopitz(RMgauss(), alpha=2)
z1 <- RFsimulate(model, x)
plot(z1, type="l")
```

---

**ExtremalGaussian**  
*Extremal Gaussian process*

**Description**

`RPschlather` defines an extremal Gaussian process.

**Usage**

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

phi  an RMmodel, see Details.
tcf  an RMmodel specifying the extremal correlation function; either phi or tcf must be given.

xi, mu, s  the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.

Details

The argument xi is always a number, i.e. $\xi$ is constant in space. In contrast, $\mu$ and $s$ might be constant numerical value or given a RMmodel, in particular by a RMtrend model. The default values of $\mu$ and $s$ are 1 and $z\xi$, respectively.
The argument phi can be any random field for which the expectation of the positive part is known at the origin.
It simulates Extremal Gaussian process $Z$ (also called “Schlather model”), which is defined by

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

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

Note

Advanced options are maxpoints and max_gauss, see ROptions.

Author(s)

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

See Also

RMmodel, RPgauss, maxstable, maxstableAdvanced

Examples

ROptions(seed=0, xi=0)
# seed=0: *ANY* simulation will have the random seed 0; set
# ROptions(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")
## fitgauss

**Details on fitting Gaussian random fields, including Box-Cox transformation**

### Description

Here some details of \texttt{RFFit} are given concerning the fitting of models for Gaussian random fields. **This documentation is far from being complete.**

### Maximum likelihood

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

### Least squares

The weighted least squares methods minimise

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

over all parametrised models of $\gamma$. 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{RFFit} are implemented.
'self'  \( w_i = (\gamma(h_i))^{-2} \)
'plain'  \( w_i = 1 \) for all \( i \).
'sqrt.nr'  \( w_i^2 \) equals the number of points \( n_i \) in bin \( i \).
'sd.inv'  \( 1/w_i \) equals the standard deviation of the variogram cloud within bin \( i \).
'internal'  Three subvariants are implemented:
'internal1'  \( w_i^2 = (N - i + 1)n_i \)
'internal2'  \( w_i = N - i + 1 \)
'internal3'  \( w_i^2 = N - i + 1 \)

Author(s)

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

See Also

RFFit, RFFit-class

Examples

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

GaussianFields  Methods for Gaussian Random Fields

Description

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

Implemented models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPCirculant</td>
<td>simulation by circulant embedding</td>
</tr>
<tr>
<td>RPCutoff</td>
<td>simulation by a variant of circulant embedding</td>
</tr>
<tr>
<td>RPCoins</td>
<td>simulation by random coin / shot noise</td>
</tr>
<tr>
<td>RPDirect</td>
<td>through the square root of the covariance matrix</td>
</tr>
<tr>
<td>RPGauss</td>
<td>generic model that chooses automatically among the specific methods</td>
</tr>
<tr>
<td>RPHyperplane</td>
<td>simulation by hyperplane tessellation</td>
</tr>
<tr>
<td>RPintrinsic</td>
<td>simulation by a variant of circulant embedding</td>
</tr>
<tr>
<td>RPNugget</td>
<td>simulation of (anisotropic) nugget effects</td>
</tr>
<tr>
<td>RPSpecific</td>
<td>model specific methods (very advanced)</td>
</tr>
<tr>
<td>RPSpectral</td>
<td>spectral method</td>
</tr>
<tr>
<td>RPTbm</td>
<td>turning bands</td>
</tr>
</tbody>
</table>
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 comment</th>
<th>memory comment</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPcirculant</td>
<td>yes</td>
<td>any</td>
<td>≤ 13</td>
<td>$O(v^3 f(n, d))$</td>
<td>$O(v^2 f(n, d))$</td>
<td>$k \sim \text{approx_step}^{-d}$</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>any</td>
<td>≤ 13</td>
<td>$O(v^3 f(k, d))$</td>
<td>$O(v^2 f(k, d))$</td>
<td>see RPcirculant above</td>
</tr>
<tr>
<td>RPCutoff</td>
<td>yes</td>
<td>1</td>
<td>≤ 4</td>
<td>$O(kn)$</td>
<td>$O(n)$</td>
<td>$k \sim \text{lattice_spacing}^{-d}$</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>1</td>
<td>≤ 4</td>
<td>$O(kn)$</td>
<td>$O(n)$</td>
<td>$k$ depends on the geometry</td>
</tr>
<tr>
<td>RPdirect</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>$O(1) O(v^2 n^2)$</td>
<td>$O(v^2 n^2)$</td>
<td>effort to investigate the covariance matrix, if $n$ covariance matrix is diagonal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$O(vn)$</td>
<td>$O(vn)$</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>$O(v^2 n^3)$</td>
<td>arbitrary covariance matrix (simulation)</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>only the selection process; $O(1)$ if first method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$O(1) O(v^2 n^2)$</td>
<td>$O(1) O(n^2)$</td>
<td>$s = \text{scale}$</td>
</tr>
<tr>
<td>RPgauss</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>$O(1) O(v^3 n^3)$</td>
<td>$O(1) O(n^2)$</td>
<td>see RPcirculant above</td>
</tr>
<tr>
<td>RPhyperplane</td>
<td>any</td>
<td>1</td>
<td>2</td>
<td>$O(n/s^d)$</td>
<td>$O(n/s^d)$</td>
<td>$n = ST$; $S$ and $T$ the number of spatial and temporal locations, respectively</td>
</tr>
<tr>
<td>RPintrinsic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(preparation)</td>
</tr>
<tr>
<td>RPNugget</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>$O(vn)$</td>
<td>$O(vn)$</td>
<td>(simulation)</td>
</tr>
<tr>
<td>RPspectral</td>
<td>any</td>
<td>1</td>
<td>≤ 2</td>
<td>$O(C(d)n)$</td>
<td>$O(n)$</td>
<td>$C(d)$: large constant increasing in $d$</td>
</tr>
<tr>
<td>Rptbm</td>
<td>any</td>
<td>1</td>
<td>≤ 4</td>
<td>$O(C(d)(n + L))$</td>
<td>$O(n + L)$</td>
<td>$C(d)$: large constant increasing in $d$; $L$ is the only the specific part</td>
</tr>
<tr>
<td>RPspecific</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>** RMplus</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>$O(vn)$</td>
<td>$O(vn)$</td>
<td></td>
</tr>
<tr>
<td>** RMS</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>$O(1)$</td>
<td>$O(vn)$</td>
<td></td>
</tr>
<tr>
<td>** RMmult</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>$O(vn)$</td>
<td>$O(vn)$</td>
<td></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>Method</th>
<th>grid</th>
<th>$v$</th>
<th>$d$</th>
<th>time comment</th>
<th>memory comment</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>$O(1) O(v^2 n^2)$</td>
<td>$O(v^2 n^2)$</td>
<td>effort to investigate the covariance matrix, if $\text{matrix_methods}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$O(v^2 nk)$</td>
<td>$O(v(n + k))$</td>
<td>$\text{covariance matrix is diagonal}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$O(v^2 nk)$</td>
<td>$O(v^2 n^2 + v n k)$</td>
<td>$\text{covariance matrix is sparse matrix with } z \text{ non-zeros}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$O(v^2 nk)$</td>
<td>$O(v^2 n^2 + v n k)$</td>
<td>$\text{arbitrary covariance matrix}$</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 simulated at \( k \) location \( y_1, \ldots, y_k \), the computing demand equals the sum of the demand for interpolating and the demand for simulating on the \( k + n \) location. (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/de/publications/software

References


See Also

RP, Other models, RMmodel, RFgetMethodNames, RFSimulateAdvanced

Examples

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

library(RandomFields, lib="~/TMP")
RFoptions(print = 3)
set.seed(1)

x <- rnorm(9000, 0, 500)
z <- RFSimulate(RM spherical(), x)
z <- RFSimulate(RM spherical(), x, max_variab=10000)
Description

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

Author(s)

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

References


Examples

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

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

alphas <- seq(1.52, 2.0, 0.02)
thetas <- seq(0.05, 3.5, 0.05)

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

Bayesian Spatial Modelling

Description

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

Details

A Bayesian approach can be taken for scalar, real valued model parameters, e.g. the shape parameter \( \nu \) in the \texttt{RMmatern} model. A random parameter can be passed through a distribution of an existing family, e.g. \( (\texttt{dnorm}, \texttt{pnorm}, \texttt{qnorm}, \texttt{rnorm}) \) or self-defined. It is passed without the leading letter \( \texttt{d}, \texttt{p}, \texttt{q}, \texttt{r} \), but as a function call e.g \( \texttt{norm()} \). This function call may contain arguments that must be named, e.g. \( \texttt{norm(mean=3, sd=5)} \).

Usage:

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

The family can be passed in three ways:

- implicitly, e.g. \texttt{RMwhittle(nu=exp())} or
- 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{RPPoisson}, a new realisation of the prior distribution(s) is drawn for each shape function
  - in all the other cases: a realisation of the prior(s) is only drawn once. This effects, in particular, Gaussian fields with argument \( n>1 \), where all the realisations are based on the same realisation out of the prior distribution(s).

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

Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>

See Also
RMmodelsAdvanced For hierarchical modelling see RR

Examples

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

## See 'RMmodels' 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)) + Rm nugget(var=NA)

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

Hyperplane Hyperplane method

Description

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

Usage

RPhyperplane(phi, boxcox, superpos, maxlines, mar_distr, mar_param ,additive)

Arguments

phi object of class RMmodel; specifies the covariance function to be simulated; only exponential covariance functions and scale mixtures of it are allowed.
boxcox the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. see RFboxcox for Details.
superpos integer. number of superposed hyperplane tessellations. Default: 300.
maxlines integer. Maximum number of allowed lines.
Default: 1000.

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

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

additive logical. If TRUE independent realisations are added, else the maximum is taken.
Default: TRUE.

Value
RPhyperplane returns an object of class Rmmodel

Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>

References

See Also
Gaussian, RP.

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

model <- 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.

Usage

RPnugget(phi, boxcox, tol, vdim)

Arguments

phi object of class Rmmodel; specifies the covariance model to be simulated. The only possible model for phi is RMnugget.

boxcox the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. see RFboxcox for Details.

tol points at a distance less than or equal to nugget.tol are considered as being identical. This strategy applies to the simulation method and the covariance function itself. Hence, the covariance function is only positive definite if nugget.tol=0.0. However, if the anisotropy matrix does not have full rank and nugget.tol=0.0 then, the simulations are likely to be odd. The value of 
nugget.tol should be of order 10^{-15}.
Default: 0.0

vdim positive integer; the model is treated vdim-variate, vdim=1 (default) corresponds to a univariate random field. Mostly, the value of vdim is set automatically. Default is that it takes the value of the submodel phi

Details

This method only allows RMnugget as a submodel.

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

Value

RPnugget returns an object of class Rmmodel

Author(s)

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

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
model <- RMnugget()
z <- RFsimulate(model=model, 0:10, 0:10, n=4)
plot(z)

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

Description

These functions are internal and should not be used.

Usage

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

```r
rfGenerateConstants(
    RFpath = "~/RF/svn/RandomFields",
    RCauto.file = paste(RFpath, "RandomFields/R/RCauto.R", sep="/")
)
```

```r
rfGenerateTest(files = NULL,
RFpath = "~/RF/svn/RandomFields/RandomFields")
```

```r
rfGenerateMaths(files = "/usr/include/tgmath.h", Cfile = "QMath",
    RFpath = "~/RF/svn/RandomFields/RandomFields")
```

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

ScreenDevice(height, width)

FinalizeExample()
StartExample(reduced = TRUE)

Dependencies(install = all(pkgs == all.pkgs),
    check=TRUE, pkgs = all.pkgs, dir = "Dependencies")

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

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

Arguments
assigning, RPath, RModels.file, RAuto.file, files, Cfile
    internal
exclude, include, ask, echo, halt, ignore.all, path, package, read.rd.files, libpath, single.runs
    internal; ignore.all refers to the ‘all’ export statement in the namespace – whether
this should be ignored. if read.rf.files is TRUE or a path to the Rd files, then
the man pages are analysed to get all examples; ignore.all is then ignored. If
FALSE only examples of functions (which are search in the environments) are
run.
install, check, pkgs, dir
    internal
height,width window sizes
data, factor, xlim, ylim, col, fill, ...
    internal
reduced internal

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

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

## internal functions: no examples given

# for (i in dep.packages) cat(maintainer(i), "\n")
Covariance models for multivariate and vector valued fields

Description

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

Author(s)

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

References


See Also

weather, SS12, S10

Examples

```r
## Not run:

# SECTION 4: UNCONDITIONAL SIMULATION

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

## Fig. 1: linear model of coregionalization
M1 <- c(0.9, 0.6)
M2 <- c(sqrt(0.19), 0.8)
model <- RMmatrix(M = M1, RMwhittle(nu = 0.3)) +
    RMmatrix(M = M2, RMwhittle(nu = 2))
x <- y <- seq(-10, 10, 0.2)
simu <- RFsimulate(model, x, y)
plot(simu)
```
## Fig. 2: Wackernagel’s delay model
```r
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
```r
model <- RMdelay(RMstable(alpha = 1.9, scale = 2), s = c(0, 4)) +
          RMdelay(RMstable(alpha = 1.9, scale = 2), s = c(4, 0))
simu <- RFsimulate(model, x, y)
plot(simu, zlim = 'joint')
```

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

## Fig. 5: Gneiting’s bivariate Whittle-Matern model
```r
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 coregionalisation
```r
M1 <- c(0.9, 0.6)
M2 <- c(sqrt(0.19), 0.8)
A1 <- RMangle(angle = pi/4, diag = c(0.1, 0.5))
A2 <- RMangle(angle = 0, diag = c(0.1, 0.5))
model <- RmMatrix(M = M1, RMgenGneiting(kappa = 0, mu = 2, Aniso = A1)) +
            RmMatrix(M = M2, RMgenGneiting(kappa = 3, mu = 2, Aniso = A2))
simu <- RFsimulate(model, x, y)
plot(simu)
```

## Fig. 7: random vector field whose path are curl free
```r
# 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")
```


```r
## 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(REarth2dist(weather[ , 3 : 4]))
All <- TRUE
\dontshow{if(ROptions()$internal$examples_reduced)(warning("reduced data set"))
All <- 1:10
PT <- weather[All , 1 : 2]
Dist.mat <- as.vector(REarth2dist(weather[All , 3 : 4]))
}

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

###########################################################################
## model fitting and testing
###########################################################################
## ‘parsimoneous model’
## fitting takes a while ( > 10 min)
pars <- RFfit(pars.model, distances = Dist.mat, dim = 3, data = PT)
print(pars)
```
print(pars, full=TRUE)
RFratietest(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)
RFratietest(whole)
RFcrossvalidate(whole, x = weather[All , 3 : 4], data = PT, full = TRUE)

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

# kriging

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

x <- RFearth2cartesian(weather[All , 3 : 4])

plot(z)

# End(Not run)
Description

This man pages documents some major changings in RandomFields.

Changes done in 3.1.0 (Summer 2015)

• full (univariate) trend modelling
• error in particular in RFit corrected
• RFit 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 then the functions in version 2. E.g., RFsimulate can perform unconditional simulation, conditional simulation and random imputing.
  – Only those arguments are kept in the functions that are considered as being absolutely necessary. All the other arguments can be included as options.
  – RFgui is an instructive interface based on tcl/tk, replacing the former ShowModels
• Inference for Gaussian random fields
  – RFit has undergone a major revision. E.g.:
    (i) estimation random effects model with spatial covariance structure
    (ii) automatic estimation of 10 and more arguments in multivariate and/or space-time models
  – RFempiricalvariogram is now based on an fft algorithm if the data are on a grid, even allowing for missing values.
– **RFraciostest** has been added.

**Processes**

– Maxstable processes modelling of maxstable processes has been enhanced, including
  (i) the simulation of Brown-Resnick processes
  (ii) initial support of tail correlation functions;
– Further processes chi2 processes, compound Poisson processes, binary processes added.

**Models**

– the formula notation for linear models may now be defined
– Novel, user friendly definition of the covariance models
– Multivariate and vector valued random fields are now fully included
– The user may now define his own functions, to some extend.
– The trend allows for much more flexibility
– Distributions may now included which will be extended to Bayesian modelling in future.

---

**Examples**

```r
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 <- RMe xp()
plot(RFsimulate(model, x)) # S4
plot(RFsimulate(model, x, spConform=FALSE)) # no class
```

---

**Mathematical C functions**

*Transformation of coordinate systems*

**Description**

The functions provide mathematical c functions as **RMmodels**

**Usage**

```r
RFcalc(model)
R.minus(a, b, factor)
R.plus(a, b, factor)
R.div(a, b, factor)
R.mult(a, b, factor)
R.const(a)
R.c(a, b, c, d, e, f, g, h, i, j, factor)
R.p(proj, new, factor)
```
Mathematial C functions

R.is(a, is, b)
R.lon()
R.lat()

R.acos(a)
asin(x)
R.asin(a)
atan(x)
R.atan(a)
atan2(y, x)
R.atan2(a, b)
cos(x)
R.cos(a)
sin(x)
R.sin(a)
tan(x)
R.tan(a)
acosh(x)
R.acosh(a)
asinh(x)
R.asinh(a)
atanh(x)
R.atanh(a)
cosh(x)
R.cosh(a)
sinh(x)
R.sinh(a)
tanh(x)
R.tanh(a)
exp(x)
R.exp(a)
log(x)
R.log(a)
expm1(x)
R.expm1(a)
log1p(x)
R.log1p(a)
logb(x)
R.logb(a)
R.exp2(a)
log2(x)
R.log2(a)
R.pow(a, b)
sqrt(x)
R.sqrt(a)
R.hypot(a, b)
R.cbrt(a)
R.ceil(a)
abs(x)
R.fabs(a)
floor(x)
R.floor(a)
R.fmod(a, b)
R.nearbyint(a)
round(x, ...)
R.round(a)
trunc(x)
R.trunc(a)
R.lrint(a)
R.lrint(a)
R.lround(a)
R.copysign(a, b)
R.erf(a)
R.erfc(a)
gamma(x)
R.tgamma(a)
lgamma(x)
R.lgamma(a)
R.rint(a)
R.nextafter(a, b)
R.nexttoward(a, b)
R.remainder(a, b)
R fdim(a, b)
max(...)
R.fmax(a, b)
min(...)
R.fmin(a, b)

Arguments

model object of class Rmmodel, in particular R.model
x, y, a, b, c, d, e, f, g, h, i, j, ...
constant or object of class Rmmodel, in particular R.model
is one of "=". "%", "<=", "<", ">", ">"
factor constant factor multiplied with the function. This is useful when linear models are built
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 should always be given if the coordinates are not cartesian.

Details

R.plus adds two values
R.minus substracts two values
Mathematical C functions

- **R.mult** multiplies two values
- **R.div** divides two values
- **R.const** defines a constant
- **R.c** builds a vector
- **R.is** evaluates equalities and inequalities; note that TRUE is returned if the equality or inequality holds up to a tolerance given by `RFoptions()$nugget$tol`
- **R.p** takes a component out of the vector giving the location
- **R.lon, R.lat** longitudinal and latitudinal coordinate, given in the spherical system, i.e. in radians. (Earth system is in degrees).

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

**Note**

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

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

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

**Author(s)**

Martin Schlather, <schlather@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))

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

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

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

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

---

**Max-stable random fields**

*Simulation of Max-Stable Random Fields*

---

**Description**

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

See also `maxstableAdvanced` for more advanced examples.

**Implemented models and methods**

**Models**

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

**Methods**

- **RPbrmixed**: simulation of Brown-Resnick processes using M3 representation
- **RPbrorig**: simulation of Brown-Resnick processes using the original definition
- **RPbrshifted**: simulation of Brown-Resnick processes using a random shift
Max-stable random fields

Author(s)

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

References


See Also

RP, RMmodel, RPgauss, RPbernoulli maxstableAdvanced

Examples

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

### currently not programmed

```r
# 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(RP schlather(model, xi=0), x, n=1000)
plot(z)
hist(unlist(z@data), 50, freq=FALSE)
```
Max-stable random fields, advanced

Simulation examples of advanced Max-Stable Random Fields

Description

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

Author(s)

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

References


See Also

RPmaxstable

Examples

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

# RFoptions(seed=NA) to make them all random again

n <- 100

model <- RMexp(var=1.62 / 2)
x <- seq(0, 5, 0.2)
y <- seq(0, 10, 0.2)

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

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

## Extremal Binary Gaussian

```r
binary.model <- RPinoulli(RMbr2bg(model))
z <- RFsimulate(RPschlather(binary.model), y, y)
plot(z)
```
Obsolete Functions

```r
simu <- RFSimulate(RPsclatherr(binary.model), x, n=n, max_gauss=5)
auswertung(simu, model)
```

## Description

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

## Usage

```r
Covariance(x, y = NULL, model = NULL, param = NULL, dim = if
  (!missing(Distances)) { if (is.matrix(x)) ncol(x) else 1},
  Distances, fctcall = c("Cov", "Variogram", "CovMatrix"))
CovarianceFct(x, y = NULL, model, param = NULL, dim = if
  (!missing(Distances)) { if (is.matrix(x)) ncol(x) else 1},
  Distances, fctcall = c("Cov", "Variogram", "CovMatrix"))
CovMatrix(x, y = NULL, model, param = NULL, dim = if
  (!missing(Distances)) { if (is.matrix(x)) ncol(x) else 1}, Distances)
DeleteAllRegisters()
DeleteRegister(nr=0)
DoSimulateRF(n = 1, register = 0, paired=FALSE, trend=NULL)
InitSimulateRF(x, y = NULL, z = NULL, T=NULL, grid=!missing(gridtriple),
  model, param, trend, method = NULL, register = 0,
  gridtriple, distribution=NA)
InitGaussRF(x, y = NULL, z = NULL, T=NULL, grid=!missing(gridtriple),
  model, param, trend=NULL, method = NULL, register = 0, gridtriple)
GaussRF(x, y = NULL, z = NULL, T=NULL, grid=!missing(gridtriple), model,
  param, trend=NULL, method = NULL, n = 1, register = 0, gridtriple,
  paired=FALSE, PrintLevel=1, Storing=TRUE, ...)
Variogram(x, model, param = NULL, dim = if (!missing(Distances))
  { if (is.matrix(x)) ncol(x) else 1}, Distances)
InitMaxStableRF(x, y = NULL, z = NULL, grid=NULL, model, param, maxstable,
  method = NULL, register = 0, gridtriple = FALSE)
MaxStableRF(x, y = NULL, z = NULL, grid=NULL, model, param, maxstable,
  method = NULL, n = 1, register = 0, gridtriple = FALSE, ...)
EmpiricalVariogram(x, y = NULL, z = NULL, T=NULL, data, grid=NULL, bin,
  gridtriple = FALSE, phi, theta, deltaT)
Kriging(krige.method, x, y=NULL, z=NULL, T=NULL, grid=NULL, gridtriple=FALSE,
```
model, param, given, data, trend=NULL,pch=".", return.variance=FALSE, allowdistanceZero = FALSE, cholesky=FALSE)
CondSimu(krig.method, x, y=NULL, z=NULL, T=NULL, grid=NULL, gridtriple=FALSE, model, param, method=NULL, given, data, trend=NULL, n=1, register=0, err.model=NULL, err.param=NULL, err.method=NULL, err.register=1, tol=1E-5, pch=".", paired=FALSE, na.rm=FALSE)
RParameters(...)
hurst(x, y = NULL, z = NULL, data, gridtriple = FALSE, sort=TRUE, block.sequ = unique(round(exp(seq(log(min(3000, dim[1] / 5)),
          log(dim[1]), len=min(100, dim[1]))) ),
          fft.m = c(1, min(1000, (fft.len - 1) / 10)),
          fft.max.length = Inf,
          method=c("dfa", "fft", "var"), mode=c("plot", "interactive"),
          pch=16, cex=0.2, cex.main=0.85,
          PrintLevel=RFOptions()$general$printlevel,height=3.5, ...)
fractal.dim(x, y = NULL, z = NULL, data, grid=TRUE, gridtriple = FALSE,
          bin, vario.n=5, sort=TRUE, fft.m = c(65, 86), fft.max.length=Inf,
          fft.max.regr=1500000, fft.shift = 50, method=c("variogram", "fft"),
          mode=c("plot", "interactive"), pch=16, cex=0.2, cex.main=0.85,
          PrintLevel = RFOptions()$general$printlevel, height=3.5, ...)
fitvario(x, y=NULL, z=NULL, T=NULL, data, model, param, lower=NULL,
        upper=NULL, sill=NA, grid=!missing(gridtriple), gridtriple=FALSE, ...)

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

Value
See ‘version2’ for details on these obsolete commands.

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

See Also
The functions that should be used instead are, for instance, RFCov, RFCovmatrix, RFvariogram,
RFsimulate,RFinterpolate,RFempiricalvariogram,RFfit,RFoptions,RFhurst,RFfractaldim
See ‘version2’ for details on the obsolete commands.

Examples
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
## no examples given, as command is obsolete

---

### Others

#### Auxiliary and other Models

#### Description

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

#### Implemented models

- **Distribution families** See RR.
- **Evaluation operators** See RF.
- **Random Fields / Random Processes** See RP.
- **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 $H_n$.
  - `RMmpplus`: operator to define mixed moving maxima (M3) processes.
  - `RMsps`: spectral functions belonging to a tail correlation function of the Gneiting class $H_n$.
  - `Rmpolygon`: Indicator of a typical Poisson polygon.
  - `RMrational`: shape function used in the Bernoulli paper given in the references.
  - `RMrotat`: shape function used in the Bernoulli paper given in the references.
  - `RMsign`: random sign.
  - `RMtruncsupport`: truncates the support of a shape in a Poisson based model.

#### Special transformations

- `RMeaxxa`: shape function used in the Bernoulli paper given in the references.
- `RMetaxxa`: shape function used in the Bernoulli paper given in the references.
- `RMid`: identity.
- `RMrotation`: shape function used in the Bernoulli paper given in the references.
Other models

\begin{itemize}
\item \texttt{RMcoord} passing new coordinates in a mixed model
\item \texttt{RMuser} User defined covariance model
\end{itemize}

Author(s)

Alexander Malinowski, <malinowski@math.uni-mannheim.de>
Martin Schlather, <schlather@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)  ## *ANY* simulation will have the random seed 0; set
RFoptions(seed=NA)  ## *ANY* simulation will have the random seed 0; set
RFgetModelNames()
\end{verbatim}

Description

Here, an overview is given over the papers co-authored by M. Schlather that involve \texttt{RandomFields}.

Author(s)

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

References

\begin{itemize}
  See GKS11 for the code.
  See GSPSJ06 for the code.
  See SS12 for the code.
\end{itemize}
  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

   # for examples, see the specific papers.

---

### plot-method

*Methods for function plot in package RandomFields*

#### Description

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

The plot methods not described here are described together with the class itself, for instance, `RFFit`, `RFempVariog` `RMmodel`.

#### Usage

```r
## S4 method for signature 'RFspatialDataFrame,missing'
plot(x, y,
    MARGIN=c(1,2), MARGIN.slices=NULL,
    n.slices = if (is.null(MARGIN.slices)) 1 else 10,
    nmax=6,
    plot.variance = (!is.null(x@RFparams$has.variance) &&
                   x@RFparams$has.variance), select.variables, zlim, legend=TRUE,
    MARGIN.movie = NULL, file=NULL, speed = 0.3, height.pixel=300, width.pixel=300,
    ...)```
## S4 method for signature 'RFspatialDataFrame,RFspatialGridDataFrame'
plot(x, y,
    MARGIN=c(1,2), MARGIN.slices=NULL,
    n.slices = if (is.null(MARGIN.slices)) 1 else 10,
    nmax=6,
    plot.variance = (!is.null(x@RFparams$has.variance) &&
    x@RFparams$has.variance), select.variables, zlim, legend=TRUE,
    MARGIN.movie = NULL, file=NULL, speed = 0.3, height.pixel=300, width.pixel=300, ...)

## S4 method for signature 'RFspatialDataFrame,RFspatialPointsDataFrame'
plot(x, y,
    MARGIN=c(1,2), MARGIN.slices=NULL,
    n.slices = if (is.null(MARGIN.slices)) 1 else 10,
    nmax=6,
    plot.variance = (!is.null(x@RFparams$has.variance) &&
    x@RFparams$has.variance), select.variables, zlim, legend=TRUE,
    MARGIN.movie = NULL, file=NULL, speed = 0.3, height.pixel=300, width.pixel=300, ...)

## S4 method for signature 'RFspatialDataFrame,matrix'
plot(x, y,
    MARGIN=c(1,2), MARGIN.slices=NULL,
    n.slices = if (is.null(MARGIN.slices)) 1 else 10,
    nmax=6,
    plot.variance = (!is.null(x@RFparams$has.variance) &&
    x@RFparams$has.variance), select.variables, zlim, legend=TRUE,
    MARGIN.movie = NULL, file=NULL, speed = 0.3, height.pixel=300, width.pixel=300, ...)

## S4 method for signature 'RFspatialDataFrame,data.frame'
plot(x, y,
    MARGIN=c(1,2), MARGIN.slices=NULL,
    n.slices = if (is.null(MARGIN.slices)) 1 else 10,
    nmax=6,
    plot.variance = (!is.null(x@RFparams$has.variance) &&
    x@RFparams$has.variance), select.variables, zlim, legend=TRUE,
    MARGIN.movie = NULL, file=NULL, speed = 0.3, height.pixel=300, width.pixel=300, ...)

## S4 method for signature 'RFspatialGridDataFrame'
persp(x, y,
    MARGIN=c(1,2), MARGIN.slices=NULL,
    n.slices = if (is.null(MARGIN.slices)) 1 else 10,
    nmax=6,
    plot.variance = (!is.null(x@RFparams$has.variance) &&
    x@RFparams$has.variance), select.variables, zlim, legend=TRUE,
    MARGIN.movie = NULL, file=NULL, speed = 0.3, height.pixel=300, width.pixel=300,
## Arguments

- **x**
  - object of class `RFsp` or `RMmodel`; in the latter case, `x` can be any sophisticated model but it must be either stationary or a variogram model.

- **y**
  - ignored in most methods.

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

- **MARGIN.slices**
  - integer value; if `$\text{space} - \text{time} - \text{dimension} > 2$`, `MARGIN.slices` can specify a third dimension w.r.t. which a sequence of slices is plotted. Currently only works for grids.

- **n.slices**
  - integer value. The number of slices to be plotted; if `n.slices > 1`, `nmax` is set to `1`. Or `n.slices` is a vector of 3 elements: start, end, length. Currently only works for grids.

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

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

- **file, speed, height.pixel, width.pixel**
  - In case `MARGIN.movie` and `file` is given an 'avi' movie is stored using the `mencoder` command with speed argument `speed`. As temporary files `file___##.png` of size `width.pixel x height.pixel` are created.
arguments to be passed to methods; mainly graphical arguments, or further models in case of class 'RModel', see Details.

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

select.variables vector of integers or list of vectors. The argument is only of interest for multivariate models. Here, length(select.variables) gives the number of pictures shown (excluding the plot for the variances, if applicable). If select.variables is a vector of integers then exactly these components are shown. If select.variables is a list, each element should be a vector up to length \( l \leq 3 \):

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

legend logical, whether a legend should be plotted

zlim vector of length 2 with the usual meaning. In case of multivariate random fields, zlim can also be a character 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.

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 'RModel'. 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 = "RFspatialGridDataFrame", y = "matrix") Similar to method for y="missing", but additionally adds the points of y. Requires MARGIN.slices=NULL and all.equal(x@RFparams, y@RFparams).
signature(x = "RFspatialPointsDataFrame", y = "RFspatialGridDataFrame") Throws an error. Probably x and y have been interchanged.

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

signature(x = "RFspatialPointsDataFrame", y = "RFspatialPointsDataFrame") Similar to method for 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 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 = "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 = "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 simulation results.

Author(s)

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

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)

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

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

x1D <- GridTopology(from, step, len)
x2D <- GridTopology(rep(from, 2), rep(step, 2), rep(len, 2))
x3D <- GridTopology(rep(from, 3), rep(step, 3), rep(len, 3))

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

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

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

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

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

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

###########################################################################
## plot Fitting results
x <- seq(0, 1, len=21)

model <- RMexp(Aniso = cbind(c(10,0), c(0,1)))
X <- RFsimulate(x=cbind(x,x), model=model)
fit <- RFfit(~RMexp(Aniso=diag(c(NA, NA))), data=X, fit.nphi = 2,
        modus="easygoing")
plot(fit)

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

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

# options for the zlim argument
model <- RDelay(RMstable(alpha=1.9, scale=2), s=c(0, 4)) +
          RDdelay(RMstable(alpha=1.9, scale=2), s=c(4, 0))
simu <- RFsimulate(model, x, y)
plot(simu, zlim=list(data=cbind(c(-6,2), c(-2,1)), var=c(5,6)))
plot(simu, zlim=cbind(c(-6,2), c(-2,1)))
plot(simu, zlim=c(-6,2))
plot(simu, zlim="joint")

PrintModellist

Information about the implemented covariance models

Description

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

Usage

PrintModellist(operators=FALSE, internal=FALSE, newstyle=TRUE)
**Arguments**

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

**Details**

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

**Value**

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

**Note**

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

**Author(s)**

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

**See Also**

`RFgetModelNames`

**Examples**

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

**Description**

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

**Usage**

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

---

**Linear part of RMmodel**
**RFboxcox**

**Arguments**
- **data**: matrix or list of matrices.
- **boxcox**: the one or two parameters of the box cox transformation, in the univariate case. If not given, the globally defined parameters are used, see Details. In the multivariate case boxcox should be a $2 \times m$ matrix.
- **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, RFempiricalvariogram. 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 parameters $\mu$ 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/de/publications/software](http://ms.math.uni-mannheim.de/de/publications/software)

**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)
)
data <- soil["moisture"]

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

## Assuming log-Gaussian Data
print(fit <- RFFit(model, data=data, loggaus=TRUE))

## main Parameter in the Box Cox transformation to be estimated
print(fit <- RFFit(model, data=data, boxcox=NA))

---

**RFcov**

*Evaluate Covariance and Variogram Functions*

**Description**

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

**Usage**

RFcov(model, x, y = NULL, z = NULL, T = NULL, grid, distances, dim, ...)

RFvariogram(model, x, y = NULL, z = NULL, T = NULL, grid, distances, dim, ...)

RFpseudovariogram(model, x, y = NULL, z = NULL, T = NULL, grid, distances, dim, ...)

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

RFFctn(model, x, y = NULL, z = NULL, T = NULL, grid, distances, dim, ...)

Arguments

model  object of class \texttt{RMmodel}; the covariance or variogram model, which is to be evaluated

x  vector or \((n \times \text{dim})\)-matrix, where \(n\) is the number of points at which the covariance function is to be evaluated; in particular, if the model is isotropic or \text{dim}=1 then \(x\) is a vector. \(x\)

y  second vector or matrix for non-stationary covariance functions

z  \(z\)-component of point if \(xyzT\)-specification of points is used

T  \(T\)-component of point if \(xyzT\)-specification of points is used

grid  boolean; whether \(xyzT\) specify a grid

distances  vector; only if the function \texttt{RFcovmatrix} is used; the lower triangular part of the distance matrix column-wise; equivalently the upper triangular part of the distance matrix row-wise; either \(x\) or \(\text{distances}\) must be missing

dim  dimension of the coordinate space in which the model is applied

...  arguments passed to \texttt{RFcov} (\texttt{RFcovmatrix}) and arguments passed to \texttt{RFoptions}

Details

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

Value

\texttt{RFcov} returns a vector of values of the covariance function.

\texttt{RFvariogram} returns a vector of values of the variogram model.

\texttt{RFpseudovariogram} returns a vector of values of the variogram model.

\texttt{RFcovmatrix} returns a covariance matrix.

Author(s)

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

See Also

\texttt{RMmodel}, \texttt{Rfsimulate}, \texttt{RFfit}, \texttt{RFcalc}.

Examples

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

# locations:
x <- matrix(runif(15), ncol=3)
# coordinate matrix of 5 arbitrary points
\end{verbatim}
RFcov

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

# get available models
RFgetModelNames(type=c("tail correlation", "positive definite", "variogram", "shape function", "trend"))

# Example 1: a stationary covariance model
model <- RMexp()

# covariance only depends on differences
# of locations hi=pi-q1 in 3-dimensional space
# therefore, the following 2 commands yield the same
RFcov(model=model, x=x, y=y)
RFcov(model=model, x=x-y)
# yields 5 values C(hi)=C(pi,qi) for i=1,2,3,4,5

# Example 2: get covariance matrix C(x_i,x_j)
# at given locations x_i, i=1,...,n
# here for an isotropic stationary covariance model
# yields a 4 times 4 covariance matrix of the form
# C(0)  C(5)  C(3)  C(2.5)
# C(5)  C(0)  C(4)  C(2.5)
# C(3)  C(4)  C(0)  C(2.5)
# C(2.5) C(2.5) C(2.5) C(0)
model <- RMexp() # the covariance function C(x,y)=C(r) of this model
# depends only on the distance r between x and y
RFcovmatrix(model=model, distances=c(5,3,2.5,4,2.5,2.5), dim=4)

# Example 3: distinguish the different uses of x and y
x <- c(1,2,1)
y <- c(4,5,6)
# coordinate space 1-dimensional, evaluated at 3 points:
RFcov(model=model, x=as.matrix(x), y=as.matrix(y))
# coordinate space is 3-dimensional, evaluated at a pair of points
RFcov(model=model, x=t(x), y=t(y))
RFcrossvalidate  

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

Description

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

- Gaussian random fields
- linear models

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

Usage

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

Arguments

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

Value

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

- data the original data.
- predicted the values predicted by cross-validation.
- krig.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.
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/de/publications/software

References

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

See Also

RFratiotest RFfit RMmodel, RandomFields, weather.

Examples

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

## currently disabled!
RFdistr


evaluating distribution families


description

through RRdistr distribution families can the passed to RandomFields to create distributions available in the Rmodel definitions

usage

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

arguments

model an RRmodel

x the location where the density is evaluated

q the location there the probability function is evaluated

p the value where the quantile function is evaluated

n the number of random values to be drawn

dim the dimension of the vector to be drawn

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

details

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

value

as described in the arguments

author(s)

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

RRgauss, RR

Examples

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

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

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

## note that a conditional covariance function given the
## random parameters is given here:
model <- RRgauss(scale=exp())
for (i in 1:3) {
  RFoptions(seed = i + 10)
  readline(paste("Model no.", i, ": press return", sep=""))
  plot(model)
  readline(paste("Simulation no.", i, ": press return", sep=""))
  plot(RFsimulate(model, x=seq(0,10,0.1)))
}

RFempiricalvariogram

Empirical (Cross-)Variogram

Description

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

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

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

Usage

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

- **x**: matrix of coordinates, or vector of x coordinates, or object of class `GridTopology` or `raster`. If matrix, `ncol(x)` is the dimension of the index space. Matrix notation is required in case of more than 3 spatial dimensions; in this case, if `grid=FALSE`, `x_ij` is the i-th coordinate in the j-th dimension. Otherwise, if `grid=TRUE`, the columns of `x` are interpreted as gridtriples (see `grid`). If of class `GridTopology`, `x` is interpreted as grid definition and `grid` is automatically set to `TRUE`. Coordinates are not required if the data is an object of class `RFsp`, as these objects already contain their coordinates.

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

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

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

- **grid**: logical; determines whether the vectors `x`, `y`, and `z` or the columns of `x` should be interpreted as a grid definition (see Details). If `grid=TRUE`, either `x`, `y`, and `z` must be equidistant vectors in ascending order or the columns of `x` must be given in the gridtriple format `c(from, stepsize, len)` (see Details); Not required if data is of class `RFsp`.

- **data**: matrix, data.frame or object of class `RFsp`;

- **bin**: a vector giving the borders of the bins; If not specified an array describing the empirical (pseudo-)(cross-) variogram in every direction is returned.

- **phi**: an integer defining the number of sectors one half of the X/Y plane shall be 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**: 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`.

- **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.

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

Details

`RFempiricalvariogram` computes the empirical cross variogram for given (multivariate) spatial data.
The spatial coordinates \( x, y, z \) should be vectors. For random fields of spatial dimension \( d > 3 \) write all vectors as 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{RFempiricalvariogram} to calculate the pseudovariogram (see \texttt{RFoptions}).

**Value**

\texttt{RFempiricalvariogram} returns objects of class \texttt{RFempvariog}.

**Author(s)**

Sebastian Engelke, \texttt{<sebastian.engelke@unil.ch>}
Johannes Martini, \texttt{<jmartin2@uni-goettingen.de>}
Martin Schlather, \texttt{<schlather@math.uni-mannheim.de> http://ms.math.uni-mannheim.de/de/publications/software}

**References**


**See Also**

\texttt{RMstable}, \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

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 <- RFempiricalvariogram(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 <- RFempiricalvariogram(data=z, phi=4)
plot(emp.vario, model=model)
\end{verbatim}
## RFempVariog-class

**Class** RFempVariog

### Description

Class for RandomField’s representation of empirical variograms

### Usage

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

### Arguments

- **x**: object of class RFempVariog

---

## Example Code

```r
# 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 <- RFempiricalvariogram(data=z, deltaT=c(10, 1))
plot(emp.vario, model=model, nmax.T=3)

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

# multivariate and anisotropic model
model <- RMbiwm(A=matrix(c(1,1,1,2), nc=2),
                 nudiag=c(0.5,2), s=c(3, 1, 2), c=c(1, 0, 1))
x <- seq(0, 20, 0.1)
data <- RFsimulate(model, x, x, n=n)
ev <- RFempiricalvariogram(data=data, phi=4)
plot(ev, model=model, boundaries=FALSE)
```
model: object of class `RMmodel` or `class(x)=="RFempVario"` or `class(x)=="RFFit"`; a list of covariance or variogram models that are to be plotted into the same plot as the empirical variogram (and the fitted models)

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

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

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

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

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

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

boundaries: logical; only for `class(x)=="RFempVario"` and the anisotropic case where `model` is given. As the empirical variogram is calculated on a sector of angles, no exact variogram curve corresponds to the mean values in this sector. If `boundaries=TRUE` the values of the variogram on the sector boundaries are plotted. If `FALSE` some kind of mean model values are plotted. Neither the boundaries may contain the values of 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

emp.vario: value of the empirical variogram

var: the empirical (overall) variance in the data

sd: standard deviation of the variogram cloud within each bin

n.bin: number of bins

phi.centers: centres of the bins with respect to the (first) angle (for anisotropic 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
RFfit

Methods

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

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

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

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

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

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

summary provides a summary

Details

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

Author(s)
Alexander Malinowski <Alexander.Malinowski@web.de>

See Also

RFempiricalvariogram, plot-method

Examples

# see 'RFempiricalvariogram'

---

**RFfit**

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

**Description**

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

- Gaussian random fields
- linear models

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

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

Arguments

- `model`: covariance model, see `RMmodel` or type `RFgetModelNames(type="variogram")` to get all options. All parameters that are set to NA will be estimated; see the examples below.

- `x`: vector of x coordinates, or object of class `GridTopology` or `raster`; For more options see `RFsimulateAdvanced`.

- `y`: vector of y coordinates

- `z`: vector of z coordinates

- `T`: vector of T coordinates; these coordinates are given in triple notation, see `RFsimulate`.

- `data`: vector or matrix of values measured at coord; If a matrix is given then the columns are interpreted as independent realisations. If also a time component is given, then in the data the indices for the spatial components run the fastest. If an m-variate model is used, then each realisation is given as m consecutive columns of data.

- `lower`: list or vector. Lower bounds for the parameters. If `param` is a vector, `lower` has to be a vector as well and its length must equal the number of parameters to be estimated. The order of `param` has to be maintained. A component being NA means that no manual lower bound for the corresponding parameter is set. If `param` is a list, `lower` has to be of (exactly) the same structure.

- `upper`: list or vector. Upper bounds for the parameters. See also lower.

- `grid`: boolean. Whether coordinates build a grid. In nearly all cases the value of `grid` is self-detected, so that `grid` need not be given, in general.

- `methods`: Main methods to be used for estimating. If several methods, estimation will be performed with each method and the results reported.

- `sub.methods`: variants of the least squares fit of the variogram. See Details. variants of the maximum likelihood fit of the covariance function. See Details.

- `users.guess`: User's guess of the parameters. All the parameters must be given using the same rules as for either `param` (except that no NA's should be contained) or `model`.

- `distances,dim`: Instead of x-coordinates, distances might be given. The dimension of the space dim must be given explicitly.

- `optim.control`: control list for `optim`, which uses 'L-BFGS-B'. However `parscale` may not be given.

- `transform`: this is an attempt to allow binding between parameters, e.g. one parameter value is supposed to equal another one. See examples below. `transform=list()` is not valid for estimating, but returns structural information to set up the correct function. See examples below.
... further options and control arguments for the simulation that are passed to and processed by \texttt{RFoptions}.

Details

For details on the simulation methods see

- \texttt{fitgauss} for Gaussian random fields
- \texttt{fitgauss} for linear models

If \textit{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 \texttt{RFoptions} for details.

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

Value

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

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

Note

- An important optional argument is \texttt{boxcox} which indicates A Box-Cox transformation see \texttt{boxcox} in \texttt{RFoptions} and \texttt{RFboxcox} for details.
- Instead of \texttt{optim}, other optimiser can be used, see \texttt{RFfitOptimiser}.
- Several advanced options can be found in sections ‘General options’ and ‘fit’ of \texttt{RFoptions}.
- In particular, \texttt{boxcox}, \texttt{boxcox_lb}, \texttt{boxcox_ub} allow Box-Cox transformation.
- This function does not depend on the value of \texttt{RFoptions()$PracticalRange}. The function \texttt{RFfit} always uses the standard specification of the covariance model as given in \texttt{RMmodel}.

Author(s)

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

References


See Also

\texttt{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")

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

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

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

Class RFfit

Description

Class for RandomField's 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, model=NULL, method="ml",
     nmax.phi=NA, nmax.theta=NA, nmax.T=NA,
     plot.nbin=TRUE, plot.sd=FALSE, variogram = TRUE, boundaries = TRUE,...)
rfhessian(model)
```

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
- `model`: character; only for `class(x)=="RF_fit"` or `class(x)=="RFfit"`, obtained from RFfit
- `method`: character; only for `class(x)=="RFfit"`; a vector of slot names for which the fitted covariance or variogram model is to be plotted; should be a subset of slotNames(x) for which the corresponding slots are of class "RMmodelfit"; by default, the maximum likelihood fit ("ml") will be plotted
- `nmax.phi`: integer; only for `class(x)=="RFempVario"`; the maximal number of bins of angle phi that are to be plotted
- `nmax.theta`: integer; only for `class(x)=="RFempVario"`; the maximal number of bins of angle theta that are to be plotted
- `nmax.T`: integer; only for `class(x)=="RFempVario"`; the maximal number of different time bins that are to be plotted
- `plot.nbin`: logical; only for `class(x)=="RFempVario"`; indicates whether the number of pairs per bin are to be plotted
- `plot.sd`: logical; only for `class(x)=="RFempVario"`; indicates whether the calculated standard deviation (x@sd) is to be plotted (in form of arrows of length +-1*sd)
- `variogram`: logical; This argument should currently not be set by the user. If TRUE then the empirical variogram is plotted, else an estimate for the covariance function
- `boundaries`: logical; only for `class(x)=="RFempVario"` and the anisotropic case where model is given. As the empirical variogram is calculated on a sector of angles, no exact variogram curve corresponds to the mean values in this sector.
If boundaries=TRUE the values of the variogram on the sector boundaries are plotted. If FALSE some kind of mean model values are plotted. Neither the boundaries may contain the values of empirical variogram nor does the mean values need to be close the empirical variogram.

... in case of plot: arguments to be passed to methods; mainly graphical arguments, or further models in case of class ‘RMmodel’, see Details.

object see the generic function

full logical. if TRUE submodels are reported as well (if available).

Creating Objects

Objects are created by the function `Rffit`

Slots

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

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

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

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

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

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

`internal1`: RMmodelFit; analog to slot ‘autostart’

`internal2`: RMmodelFit; analog to slot ‘autostart’

`internal3`: RMmodelFit; analog to slot ‘autostart’

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

`ml`: RMmodelFit; analog to slot ‘autostart’

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

`n.covariates`: number of covariates

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

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

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

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

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

`plain`: RMmodelFit; analog to slot ‘autostart’
report: if not empty, it indicates that this model should be reported and gives a standard name of the model. Various function, e.g. print.RMmodelFit uses this information if their argument full equals TRUE.

self: RMmodelFit; analog to slot 'autostart'

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

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

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

table: matrix; summary of estimation results of different methods

transform: function;

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

ttrue.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 codesignature(obj = "RFfit"): generates persp plots

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

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

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

anova performs a likelihood ratio test base on a chisq approximation

summary provides a summary

logLik provides an object of class "logLik"

AIC,BIC provides the AIC and BIC information, respectively

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'

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

Author(s)

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

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

References

AICc:


See Also

- `RFfit`, `RFempiricalvariogram`, `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.GlideRay", "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.GN.ISRES", see `nloptr` for a description.
RFfitoptimiser

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

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

## 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"]

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

## standard optimiser 'optim'
print(unix.time(fit <- RFfit(model, data=data)))
print(fit)

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

opt <- "soma" # 450 sec % sehr schlecht
print(unix.time(fit2 <- try(RFfit(model, data=data, optimiser=opt))))
print(fit2)

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

opt <- "nloptr"
algorithm <- RC_NLOPTR_NAMES
for (i in 1:length(algorithm)) {
    print(algorithm[i])
    print(unix.time(fit2 <- try(RFFit(model, data=data, 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(unix.time(fit2 <- try(RFFit(model, data=data, optimiser=opt))))
  print(fit2)
  opt <- "GenSA" # 10*4 sec
  print(unix.time(fit2 <- try(RFFit(model, data=data, optimiser=opt))))
  print(fit2)
}

---

**RF formula**

**RF formula - syntax to design random field models with trend or linear mixed models**

**Description**

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

In general, the created formula serves two purposes:

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

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

**Details**

The formula should be of the type

\[
response \sim fixedeffects + randomeffects + errorterm
\]
or

\[ \text{response} \sim \text{trend} + \text{zero} - \text{meanrandomfield} + \text{nuggeteffect}, \]

respectively.

Thereby:

- **response**
  optional; name of response variable

- **fixed effects/trend**:
  optional, should be a sum (using +) of components either of the form \( X \text{@} \text{RMfixed}(\beta) \) or \( \text{RMtrend}(\ldots) \) with \( X \) being a design matrix and \( \beta \) being a vector of coefficients (see \text{RMfixed} and \text{RMtrend}).
  Note that a fixed effect of the form \( X \) is interpreted as \( X \text{@} \text{RMfixed}(\beta=\text{NA}) \) by default (and \( \beta \) is estimated provided that the formula is used in \text{RFfit}).

- **random effects/zero-mean random field**:
  optional, should be a sum (using +) of components of the form \( Z \text{@} \text{model} \) where \( Z \) is a design matrix and \( \text{model} \) is an object of class \text{RMmodel}.
  \( Z \text{@} \text{model} \) describes a vector of random effects which is normally distributed with zero mean and covariance matrix \( ZZ^T \Sigma \) where \( ZZ^T \) is the transpose of \( Z \) and \( \Sigma \) is the covariance matrix according to \( \text{model} \).
  Note that a random effect/random fluctuation of the form \( \text{model} \) is viewed as \( I \text{@} \text{model} \) where \( I \) is the identity matrix of corresponding dimension.

- **error term/nugget effect**
  optional, should be of the form \text{RMnugget}(\ldots). \text{RMnugget} describes a vector of iid Gaussian random variables.
  Please note that the character “@” in the RFformula-context can only be used to multiply design-matrices with corresponding vectors of fixed or random effects, whereas in the context of S4-classes “@” is used to access slots of corresponding objects.

**IMPORTANT**

Note that in formula constants are interpreted as part of a mixed model, i.e. the corresponding parameter has to be estimated (e.g. \( \sim 1 + \ldots \)) whereas 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 \text{RFoptions}.

**Author(s)**

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

**References**


**See Also**

`RMmodel, RFSimulate, RFfit, RandomFields`.

**Examples**

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

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

V <- 10
S <- 0.3
M <- 3
model <- RMexp(var=V, scale=S) + M
x <- y <- seq(1, 3, 0.1)

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

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

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

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

**Description**

The function estimates the fractal dimension of a process.

**Usage**

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

**Arguments**

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

References

variogram method


fft

- Chan, Hall and Poskitt (1995)

See Also

\texttt{RMmodel, RFhurst}

Examples

\begin{verbatim}
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)
\end{verbatim}
Description
Here, all the RF_name_ commands are listed.

Functionals of RMmodels
The user’s RMmodel is supplemented internally by operators that are tacitely assumed, e.g. RPgauus. 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.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFcalc</td>
<td>performs some simple calculations based on R.models</td>
</tr>
<tr>
<td>RFcov</td>
<td>assigns to a covariance model the covariance values at given locations</td>
</tr>
<tr>
<td>RFcovmatrix</td>
<td>assigns to a covariance model the matrix of covariance values at given locations</td>
</tr>
<tr>
<td>RFdistr</td>
<td>generic function assigning to a distribution family various values of the distribution random sample</td>
</tr>
<tr>
<td>RFfctn</td>
<td>assigns to a model the value of the function at given locations. In case of a covariance model RFfctn is identical to RFcov.</td>
</tr>
<tr>
<td>RFLikelihood</td>
<td>assigns to a model and a dataset the (log)likelihood value.</td>
</tr>
<tr>
<td>RFLinearpart</td>
<td>assigns to a model and a set of coordinates the linear part of the model, i.e. the deterministic trend and the design matrix.</td>
</tr>
<tr>
<td>RFpseudovariogram</td>
<td>assigns to a model the values of the pseudo variogram at given locations</td>
</tr>
<tr>
<td>RFSimulate</td>
<td>assigns to a model a realisation of the corresponding random field</td>
</tr>
<tr>
<td>RFvariogram</td>
<td>assigns to a model the values of the (cross-)variogram at given locations</td>
</tr>
</tbody>
</table>

Estimation and Inference

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFcrossvalidate</td>
<td>cross validation for Gaussian fields</td>
</tr>
<tr>
<td>RFempiricalvariogram</td>
<td>empirical variogram</td>
</tr>
<tr>
<td>RFfit</td>
<td>(maximum likelihood) fitting of the parameters</td>
</tr>
<tr>
<td>RFinterpolate</td>
<td>‘kriging’ and ‘imputing’</td>
</tr>
<tr>
<td>RFratiosimtest</td>
<td>likelihood ratio test for Gaussian fields</td>
</tr>
</tbody>
</table>

Graphics for Gaussian fields

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFgui</td>
<td>educational tool for</td>
</tr>
<tr>
<td></td>
<td>* manual selection of a covariance model</td>
</tr>
<tr>
<td></td>
<td>* manual fitting to the empirical variogram</td>
</tr>
<tr>
<td>RFFractaldim</td>
<td>determination of the fractal dimension</td>
</tr>
<tr>
<td>RFhurst</td>
<td>determination of the Hurst effect (long range dependence)</td>
</tr>
</tbody>
</table>
Coordinate transformations

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

Information from and to RandomFields

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

Author(s)

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

See Also

RC, RM, RP, RR, R, Rmodelgenerator

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 implementation can use the Cholesky decomposition and the singular value decomposition. It allows for arbitrary points and arbitrary grids.

- **hyperplane method.**
  
  The method is based on a tessellation of the space by hyperplanes. Each cell takes a spatially constant value of an i.i.d. random variables. The superposition of several such random fields yields approximatively a Gaussian random field.

- **intrinsic embedding.**
  
  Modified circulant embedding so that exact simulation is guaranteed for further variogram models, e.g. the fractal brownian one. Note that the simulated random field is always non-stationary. In fact, the circulant embedding is called with the Stein hypermodel, see `RMmodel`, and $A = B$ there.

  Here multiplicative models are not allowed (yet).

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

- addMPP (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.

- maxMPP (Boolean functions).
  If the random functions are multiplied by suitable, independent random values, and then the maximum is taken, a max-stable random field with unit Frechet margins is obtained - this is the max.mpp method.

- nugget.
  The method allows for generating a random field of independent Gaussian random variables. In the isotropic case and if the simple notation of a model (with model and param) is used, this method is called automatically if the nugget effect is positive except the method "circulant embedding" or "direct" have been explicitly.
  The method has been extended to zonal anisotropies, see also argument nugget.tol in RFoptions.

- particular method
  - details missing -

- Random coins.
  See marked point processes.

- sequential This method is programmed for spatio-temporal models where the field is modelled sequentially in the time direction conditioned on the previous $k$ instances. For $k = 5$ the method has its limits for about 1000 spatial points. It is an approximative method. The larger $k$ the better. It also works for certain grids where the last dimension should contain the highest number of grid points.

- spectral TBMM (Spectral turning bands).
  The principle of spectral TBMM does not differ from the other turning bands methods. However, line simulations are performed by a spectral technique (Mantoglou and Wilson, 1982).
  The standard method allows for the simulation of 2-dimensional random fields defined on arbitrary points or arbitrary grids. Here realisation is given as the cosine with random amplitude and random phase.

- TBMM2, TBMM3 (Turning bands methods; turning layers).
  It is generally difficult to use the turning bands method (TBMM2) directly in the 2-dimensional space. Instead, 2-dimensional random fields are frequently obtained by simulating a 3-dimensional random field (using TBMM3) and taking a 2-dimensional cross-section. TBMM3 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). TBMM2 and TBMM3 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 TBMM*.linesimustep and TBMM*.linesimufactor that can be set by RFoptions. For covariance models with larger values of the scale parameter, TBMM*.linesimufactor=2 is too small.
The turning layers are used for the simulations with time component. Here, if the model is a multiplicative covariance function then the product may contain matrices with pure time component. All the other matrices must be equal up to a factor and the temporal part of the anisotropy matrix (right column) may contain only zeros, except the very last entry.

**Value**

an invisible string vector of the Gaussian methods.

**Automatic selection algorithm**

— details coming soon —

**Note**

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

**Author(s)**

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

**References**


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

*RFgetMethodNames()*

---

**RFgetModel**

*Internally stored model*

**Description**

The function returns the stored model

**Usage**

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

**Arguments**

- `register` 0, ..., 21 or an evaluating function, e.g. `RFsimulate`. Place where intermediate calculations are stored. See also section Registers in `RFoptions`. 
explicate.natscale
  logical. Advanced option. If missing, then the model is returned as stored. If FALSE then any RMnatsc is ignored. If TRUE then any Rm\natsc 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.

Details

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

Value

  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 initialisation of a random field simulation.

Author(s)

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

See Also

  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)
**Description**

The function returns information about a 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)

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

**Arguments**

... any of the arguments below

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","call+internal" return also this leading model if existent.

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).
RFgetModelInfo

**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**
an `RMmodel` with `NAs` where information on the parameters is requested.

**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.

**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 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`: is case of an additive model it gives the number of NAs in each submodel
- `minmax`: a data frame containing information on all arguments set to `NAs`
  - `pmin`, `pmax`: lower and upper endpoint of the parameter values usually found in practice
  - `type`: integer; recognized particularities of a parameter; an explanation of the values is given after the table, if printed.
  - `NAN`: the number of NANS found
  - `min`, `max`: mathematically valid lower and upper endpoints of the parameter values
  - `omin`, `omax`: logical. If `FALSE` the respective mathematical endpoint is included
  - `col`, `row`: the dimension of the parameter. If the parameter is a scalar then `col = row = 1`.
    If it is a vector then `col = 1`.
  - `bayes`: currently not used (always `FALSE`)

Else a list of internal structure is returned.

**Note**

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

Author(s)

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

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

RFgetModelNames(type = RC_TYPENAMES, domain = RC_DOMAIN_NAMES, isotropy = RC_ISONAMES, 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, simpleArguments = FALSE, internal, newnames)

Arguments

type, domain, isotropy, operator, monotone, finiterange, vdim
see constants for the definition of RC_TYPENAMES, 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.

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 RModelgenerator
monotone indicates what kind of monotonicity is known, e.g., whether the model is a normal scale mixture, the latter including RMinexp or RMcauchy; see RModelgenerator
finnerange indicates whether the covariance of the model has finite range, e.g. RMcircular or RMnugget have covariances with finite range; see RModelgenerator. NA is used if the finiteness depends on the submodel.
valid.in.dim If valid.in.dim=n is passed, all models which are valid in dimension n are displayed. Otherwise valid.in.dim should be bivariate vector giving the range of requested dimensions.
maxdim if a positive integer, it specifies the maximal possible dimension of the coordinate space; note that a model which is valid in dimension n is also valid in dimension n – 1; maxdim=-1 means that the maximal possible dimension depends on the parameters of the RModel object; vdim=-2 means that the maximal possible dimension is adopted from the called submodels; see also RModelgenerator
vdim if a positive integer, vdim specifies, whether the model is vdim-variate; vdim=-1 means that being multivariate in a certain dimension depends on the parameters of the RModel object; vdim=-2 means that being multivariate in a certain dimension is adopted from the called submodels; see also RModelgenerator
If vdim is bivariate then a range is given.
group.by If group.by="propertyname" is passed, the displayed models are grouped according to propertyname.

All arguments allow also for vectors of values. In case of valid.in.dim the smallest value is taken. The interpretation is canonical.
Note that the arguments stationarity, isotropy, operator, monotone, finiterange, maxdim, vdim are also slots (attributes) of the SP4-class RModelgenerator.

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

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

• internal:
  if TRUE also RModels 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 RModels for the latter.
If the internal or newnames is given, then these old names are shown; if newnames=TRUE then also the usual names are shown. Default: FALSE.

In fact, both internal and public models can have different variants implemented. These variants are also shown if internal has a value greater than or equal 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/de/publications/software

See Also
costants, RMmodelgenerator, RMmodel, RandomFields, RC_DOMAIN_NAMES, RC_ISONAMES

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")

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

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

---

**RFgridDataFrame-class**

*Class RFgridDataFrame*

**Description**

Class for attributes in one-dimensional space.
RFgridDataFrame-class

Creating Objects

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

Slots

.RFparams: list of up to 5 elements;

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

data: object of class data.frame, containing attribute data
grid: object of class GridTopology.

Methods

plot signature(obj = "RFgridDataFrame"): generates nice plots of the random field; if space − time − 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("RFgridDataFrame")

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Author(s)

Alexander Malinowski <Alexander.Malinowski@web.de>

See Also

RFspatialGridDataFrame, which is for point locations in higher dimensional spaces, RFpointsDataFrame-class, which is for one-dimensional arbitray locations, RFsp

Examples

RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##           RFoptions(seed=NA) to make them all random again
x <- seq(0,10,length=100)
f <- RFsimulate(model=RMgauss(), x=x, n=3)

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

plot(f, nmax=2)
Arguments

Arguments

- **data**: see `RFempiricalvariogram`. If data is given, the empirical variogram is shown
- **x**: a sequence of the locations of the simulated process; if not given, x is determined by data and if data is not given by default values
- **y**: a sequence of numbers if a simulation on \( R^d \) is performed. Default is \( y = x \); see x for details.
- **same.algorithm**: Force the picture being simulated with the same algorithm so that the pictures are always directly comparable. The disadvantage is that some models are simulated only (very) approximatively.
- **ev**: instead of the data, the empirical variogram itself might be passed
- **bin**: only considered if data is given. See `RFempiricalvariogram` 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 return NULL.

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

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

Value

- If `wait < 0` the the function returns NULL else it returns the last chosen `RMmodel`.
- If `wait < 0`, a side effect effect of `RFgui` is the creation of the variable `RFgui.model` on `.GlobalEnv`.

Author(s)

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

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

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

**Hurst coefficient**

**Description**

The function estimates the Hurst coefficient of a process

**Usage**

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

**Arguments**

- `x`: matrix of coordinates, or vector of x coordinates
- `y`: vector of y coordinates
- `z`: vector of z coordinates
- `data`: the data
- `sort`: logical. If TRUE then the coordinates are permuted such that the largest grid length is in x-direction; this is of interest for algorithms that slice higher dimensional fields into one-dimensional sections.
- `block.sequ`: ascending sequences of block lengths for which the detrended fluctuation analysis and the variance method is performed.
- `fft.m`: vector of 2 integers; lower and upper endpoint of indices for the frequency which are used in the calculation of the regression line for the periodogram near the origin.
fft.max.length  if the number of points in x-direction is larger than fft.max.length then the
segments of length fft.max.length are considered, shifted by fft.max.length/2
(WOSA-estimator).

method       list of implemented methods to calculate the Hurst parameter; see Details
mode         character. A vector with components 'nographics', 'plot', or 'interactive':
             'nographics' no graphical output
             'plot' the regression line is plotted
             'interactive' the regression domain can be chosen interactively

Usually only one mode is given. Two modes may make sense in the combination
c("plot", "interactive") in which case all the results are plotted first, and then the
interactive mode is called. In the interactive mode, the regression domain is
chosen by two mouse clicks with the left mouse; a right mouse click leaves the
plot.

pch           vector or scalar; sign by which data are plotted.
cex           vector or scalar; size of pch.
cex.main      font size for title in regression plot; only used if mode includes 'plot' or
             'interactive'
printlevel    integer. If printlevel is 0 or 1 nothing is printed. If printlevel=2 warnings
             and the regression results are given. If printlevel>2 tracing information is
given.
height        height of the graphics window
...           graphical arguments

Details

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

The function calculates the Hurst coefficient by various methods:
  • detrended fluctuation analysis (dfa)
  • aggregated variation (var)
  • periodogram or WOSA estimator (fft)

Value

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

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

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

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

References

detrended fluctuation analysis


aggregated variation


periodogram


See Also
RMmodel, RFfractaldim

Examples

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

x <- runif(1000)
h <- RFhurst(1:length(x), data=x)
Description

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

Usage

RFinterpolate(model, x, y = NULL, z = NULL, T = NULL, grid=NULL, distances, dim, data, given=NULL, err.model,
       ignore.trend = FALSE, ...)

Arguments

model string: covariance model, see RMmodel, or type RFgetModelNames(type="variogram") to get all options.
x (n × d) matrix or vector of x coordinates, or object of class GridTopology or raster; coordinates of n points to be kriged. For more options see RFsimulateAdvanced.
y optional vector of y coordinates
z optional vector of z coordinates
T optional vector of time coordinates, T must always be an equidistant vector. Instead of T=seq(from=From, by=By, len=Len) one may also write T=c(From, By, Len).
grid logical; determines whether the vectors x, y, and z should be interpreted as a grid definition; RandomFields can find itself the correct value in nearly all cases. See also RFsimulateAdvanced.
distances another alternative to pass the (relative) coordinates, see RFsimulateAdvanced.
dim Only used if distances are given.
data Matrix, data.frame or object of class RFsp; coordinates and response values of measurements; given is not given and data is a matrix or data is a data.frame, the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field which are kriged separately; if the argument x is missing, data may contain NAs, which are then replaced by the kriged values (imputing); for details on matching of variable names see RFsimulateAdvanced; if of class RFsp
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.
err.model For conditional simulation and random imputing only. Usually err.model=RMnugget(var=var), or not given at all (error-free measurements).
RFinterploate

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

... for options, etc.

Details

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

Value

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

If variance.return=FALSE (default), Kriging returns a vector or matrix of kriged values corre-
sponding 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.frame are used to tell which column contains the data and the coordinates, respectively.

Author(s)

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

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

References

See Also
RMmodel, RFempiricalvariogram, RandomFields.

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

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

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

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

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

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

close.screen(all = TRUE)

---

### RFlinearpart

**Linear part of RMmodel**

**Description**

*RFlinearpart* returns the linear part of a model

**Usage**

```r
RFlinearpart(model, x, y = NULL, z = NULL, T = NULL, grid,
data, distances, dim, set=0, ...)```

**Arguments**

- **model**: object of class `RMmodel`; the covariance or variogram model, which is to be evaluated
- **x**: vector or \((n \times \text{dim})\)-matrix, where \(n\) is the number of points at which the covariance function is to be evaluated; in particular, if the model is isotropic or \(\text{dim}=1\) then \(x\) is a vector. \(x\)
- **y**: second vector or matrix for non-stationary covariance functions
- **z**: \(z\)-component of point if \(\text{xyzT}\)-specification of points is used
- **T**: \(T\)-component of point if \(\text{xyzT}\)-specification of points is used
- **grid**: boolean; whether \(\text{xyzT}\) specify a grid
- **data**: vector or matrix of values measured at \(\text{coord}\); If a matrix is given then the columns are interpreted as independent realisations. If also a time component is given, then in the data the indices for the spatial components run the fastest. If an \(m\)-variate model is used, then each realisation is given as \(m\) consecutive columns of \(\text{data}\).
distances  vector; the lower triangular part of the distance matrix column-wise; equivalently the upper triangular part of the distance matrix row-wise; either \( x \) or distances must be missing

dim  dimension of the coordinate space in which the model is applied; only necessary for given distances

set  integer. See section Value for details.

...  for advanced further options and control arguments for the simulation that are passed to and processed by \texttt{RFoptions} 

Value

\texttt{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. \( \text{NA} \ast \sin(R.p(new="isotropic")) + \text{NA} + R.p(new="isotropic") \) is OK, but not \( \sin(R.p(new="isotropic")) \ast \text{NA} + \text{NA} + R.p(new="isotropic") \)

Author(s)

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

See Also

\emph{Bayesian, RMmodel, RFSimulate, RFlikelihood}.

Examples

\begin{verbatim}
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))
\end{verbatim}
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, distances, dim, likelihood,
             estimate_variance = NA, ...)
```

**Arguments**

- `model`  
  object of class `RMmodel`; the covariance or variogram model, which is to be evaluated
- `x`  
  vector or \((n \times \text{dim})\)-matrix, where \(n\) is the number of points at which the covariance function is to be evaluated; in particular, if the model is isotropic or \(dim=1\) then \(x\) is a vector. \(x\)
- `y`  
  second vector or matrix for non-stationary covariance functions
- `z`  
  \(z\)-component of point if \(xyzT\)-specification of points is used
- `T`  
  \(T\)-component of point if \(xyzT\)-specification of points is used
- `grid`  
  boolean; whether \(xyzT\) specify a grid
- `data`  
  vector or matrix of values measured at \(coord\); If a matrix is given then the columns are interpreted as independent realisations.
  If also a time component is given, then in the data the indices for the spatial components run the fastest.
  If an \(m\)-variate model is used, then each realisation is given as \(m\) consecutive columns of \(data\).
- `distances`  
  vector; the lower triangular part of the distance matrix column-wise; equivalently the upper triangular part of the distance matrix row-wise; either \(x\) or \(distances\) must be missing
- `dim`  
  dimension of the coordinate space in which the model is applied; only necessary for given \(distances\)
- `likelihood`  
  not programmed yet. Character. choice of kind of likehood ("full", "composite", etc.), see also likelihood for `RFFit` in `ROptions`.
- `estimate_variance`  
  logical or NA. See Details.
- `...`  
  for advanced further options and control arguments for the simulation that are passed to and processed by `RFOptions`
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/de/publications/software

See Also

`Bayesian`, `RMmodel`, `RFfit`, `RFsimulate`, `RFlinearpart`.

Examples

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

require("mvtnorm")

pts <- 5
repet <- 3
model <- RMexp()
x <- runif(n=pts, min=-1, max=1)
y <- runif(n=pts, min=-1, max=1)
data <- as.matrix(RFsimulate(model, x=x, y=y, n=repet, spC = FALSE))
print(cbind(x, y, data))
print(unix.time(likei <- RFlikelihood(model, x, y, data=data)))
str(likei, digits=8)

L <- 0
C <- RFcovmatrix(model, x, y)
for (i in 1:ncol(data)) {
  print(unix.time(dn <- dmnorm(data[,i], mean=rep(0, nrow(data)),
                          sigma=C, log=TRUE)))
  L <- L + dn
}
print(L)
stopifnot(all.equal(likei$log, L))
```
pts <- 5
repet <- 1
trend <- 2 * sin(R.p(new="isotropic")) + 3
#trend <- RMtrend(mean=0)
model <- 2 * RMexp() + trend
x <- seq(0, pi, len=10)
data <- as.matrix(RFsimulate(model, x=x, n=repet, spC = FALSE))
print(cbind(x, y, data))

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

L <- 0
tr <- RFfctn(trend, x=x, spC = FALSE)
C <- RFcovmatrix(model, x)
for (i in 1:ncol(data)) {
    print(unix.time(dn <- dmvnorm(data[,i], mean=tr, sigma=C, log=TRUE)))
    L <- L + dn
}
print(L)
stopifnot(all.equal(likeli$log, L))

pts <- c(4, 5)
repet <- c(2, 3)
trend <- 2 * sin(R.p(new="isotropic")) + 3
model <- 2 * RMexp() + trend
x <- y <- data <- list()
for (i in 1:length(pts)) {
    x[i] <- list(x = runif(n=pts[i], min=-1, max=1),
                  y = runif(n=pts[i], min=-1, max=1))
    data[i] <- as.matrix(RFsimulate(model, x=x[i]$x, y=x[i]$y,
                                     n=repet[i], spC = FALSE))
}

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

L <- 0
for (p in 1:length(pts)) {
    tr <- RFfctn(trend, x=x[[p]]$x, y=x[[p]]$y, spC = FALSE)
    C <- RFcovmatrix(model, x=x[[p]]$x, y=x[[p]]$y)
    for (i in 1:ncol(data[[p]])) {
        print(unix.time(dn <- dmvnorm(data[[p]][,i], mean=tr, sigma=C,
RFoldstyle

log=TRUE)))
L <- L + dn
)
print(L)

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

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/de/publications/software

See Also

See ‘version2’ for details on the commands of version 2.
Examples

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

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

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

---

**RFoptions**

Setting control arguments

Description

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

Usage

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

Arguments

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

Details

The subsections below comment on

1. general: General options
2. br: Options for Brown-Resnick Fields
3. circulant: Options for circulant embedding methods `RPcirculant`
4. coords: Options for coordinates and units, see coordinate systems
5. direct: Options for simulating by simple matrix decomposition
6. distr: Options for distributions, in particular `RRrectangular`
7. empirvario: Options for calculating the empirical variogram
8. fit: Options for `Rffit`, `Rfraciostest`, 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. krig: Options for Kriging
14. maxstable: Options for simulating max-stable random fields
15. mpp: Options for the random coins (shot noise) methods
### General options

#### allowdistancezero

- **Boolean.** Only used in \texttt{RFinterpolate} and in \texttt{RFFit}. If true, then multiple observations or identical locations are allowed within a single data set. In this case, the coordinates are slightly scattered, so that the points have some tiny distances.
- **Default:** FALSE.

#### cPrintlevel

- **cPrintlevel** is automatically set to \texttt{printlevel} when \texttt{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 \texttt{printlevel} works on the R level whereas \texttt{cPrintlevel} works on the C level.**
- **Default:** 1

#### detailed_output

- **Logical.** If TRUE some function, e.g. \texttt{RFcrossvalidate} will return additional information.

#### every

- **Integer.** If greater than zero, then every everyth iteration is printed if simulated by TBM or random coin method. The value zero means that nothing is printed.
- **Default:** 0

#### exactness

- **Logical or NA.** Currently only used when simulating Gaussian random fields.
  - **TRUE:** \texttt{RPcoins}, \texttt{RPhyperplane}, \texttt{RPsequential}, \texttt{RPspectral} and \texttt{RPtmb} and approximative circulant embedding are excluded. If the circulant embedding method is considered as badly behaved, then the matrix decomposition methods are preferred.
  - **FALSE:** all the methods are allowed. If the circulant embedding method is considered as badly behaved or the number of points to be simulated is large, the turning bands methods are rather preferred.
• NA: Similar to FALSE, but some inexact algorithms get less preference.

Default: NA.

expected_number_simu  positive integer which is usually set internally as the value of the argument 
\textit{n} in \texttt{RFsimulate}. The argument \texttt{expected_number_simu} should be set only by an advanced 
users and only if \texttt{RFsimulate} will be called with argument \texttt{n} alone.

gridtolerance  used in \texttt{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

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

Default: TRUE

\texttt{modus_operandi}  character. One of the values "careless", "sloppy", "easygoing", "normal", 
"precise", "pedantic", "neurotic". \textbf{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 algo-
rithms, 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 huge computing times.

Default: "normal"

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

Default: FALSE.

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

The output mode \texttt{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 \texttt{spConform}.

\texttt{pch}  character. \texttt{Rffit}: shown before evaluating any method; if \texttt{pch='^'} then one or two additional 
steps in the MLE methods are marked by "+" and 

\texttt{Simulation:} 
The character is printed after each performed simulation if more than one simulation is per-
fomed at once. If \texttt{pch='!' then an absolute counter is shown instead of the character. If} 
\texttt{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 
\texttt{R} functions, e.g. \texttt{Sweave}.

Default: 

\texttt{practicalrange}  logical or integer. If not \texttt{FALSE} the range of primitive covariance functions is 
adjusted so that \texttt{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 \texttt{cov(1)} is about 0.05 (for \texttt{scale=1}) for models without range. See \texttt{RMmodel} or 
type 
\texttt{RFgetModelNames(type="positive definite", domain="single variable", isotropy="isotropic", op} for the list of primitive models.
RFoptions

- 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 RFfit)
5: function flow information of large functions
6: errors that are internally treated
7: details on intermediate calculations
8: further details on intermediate calculations

Default: 1

reportcoord character. Current values are "always", "important", "warn", "never".
Both "warn" and "important" have any effect only if the coordinate system is changed internally. In this case "warn" yields a displayed warning message whereas "important" adds an attribute to the result as in the case "always".
If "always" or "important" the reports are added as attribute to the results. Note that in this case the class of the result may change (e.g. from "numeric" to "atomic").

Default: "warn"

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

Default: TRUE

seed integer. If NULL or NA set.seed is not called. Otherwise, set.seed(seed) is set before simulations are performed, e.g. by RFSimulate or RFdistr.
If the argument is set locally, i.e., within a function, it has the usual local effect. If it is set globally, i.e. by RFoptions the seed is fixed for all subsequent calls.
If the number of simulations n is greater than one and if RFoptions(seed=seed) is set, the ith simulation is started with the seed 'seed+i-1'.
Note also that RFratiotest has its own argument seed with a slightly different meaning.

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 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 occured. If storing=TRUE, then additional simulations can be performed by calling RFsimulate with at most the argument n. This call can then be much faster, but the a rather large amount of memory could be kept.

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

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

vdim_close_together logical. Used especially in functions that create covariance matrices. If the model is multivariate, then two ways of ordering the matrix exist. To consider first all variables at a certain location (vdim_close_together=TRUE) or to consider first all locations keeping the variable fixed (vdim_close_together=FALSE). Note that several simulation methods rely on the value FALSE, so that these methods will not work anymore if vdim_close_together=TRUE. Default: FALSE.

2. Options for Brown-Resnick Fields
corr_factorrr to do
deltaAM to do

maxtrendmem integer; the maximal number of trends for shifted locations that may be stored at the same time when simulating BR processes via RPbrshifted; if maxtrendmem is large, multiple trend evaluation may be avoided. Default: 1e8.

meshsize positive; width of the grid on which the shape functions in the M3 representation of BR processes are simulated; only used for simulation of BR processes via RPbrmixed. Default: 0.1.

optim_mixed positive integer; only used for simulation of BR processes via RPbrmixed.

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

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

If optim_mixed=2, areamat is optimized and lambda is estimated. Default: 1.

optim_mixed_maxpoints positive integer; only used for simulation of BR processes via RPbrmixed with optim_mixed>0. Maximal number of Poisson points used for the optimization of areamat and the estimation of lambda. Default: 10000.
optim_mixed_tol value in $[0, 1]$; only used for simulation of BR processes via \texttt{RPbrmixed} with optim_mixed=2. In this case, \texttt{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.

\texttt{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.

\texttt{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 \texttt{RPbrmixed} with optim_mixed=2. Default: 7.

3. circulant: \textbf{Options for circulant embedding methods, cf. \texttt{RPcirculant}}

These options influence the standard circulant embedding method, cutoff circulant embedding intrinsic circulant embedding. It can also influence \texttt{RPTbm} if the line is simulated with any circulant embedding method.

approx_maxgrid See \texttt{RPcirculant}
approx_step See \texttt{RPcirculant}
dependent See \texttt{RPcirculant}
force See \texttt{RPcirculant}
maxGB See \texttt{RPcirculant}
maxmem See \texttt{RPcirculant}
mmin See \texttt{RPcirculant}
strategy See \texttt{RPcirculant}
tolIm See \texttt{RPcirculant}
tolRe See \texttt{RPcirculant}
trials See \texttt{RPcirculant}
useprimes See \texttt{RPcirculant}

4. coords: \textbf{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
5. direct: **Options for simulating by simple matrix decomposition**

max_variab See RPdirect

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

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

- Default: 1e-20.

maxit Default value to simulate from the RRrectangular distribution. The number of iterative steps where the 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 before the \( n \) member is returned. See also maxsteps.

- Default: 15.

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

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

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

- Default: 20.

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

- Default: 8.

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

- Default: 1000.

safety Default value to simulate from the RRrectangular distribution. First, at the origin, the first power function of the Taylor expansion is taken as potential upper function. The constant of the power function are increased by factor \( 1+\text{safety} \) 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 \texttt{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 \texttt{innermin}.
Similar procedure is performed for the tail.
Default: 0.08.

7. \texttt{empvario}: \textbf{Options for calculating the empirical variogram}

\texttt{fft} Logical. Determines whether FFT should be used for data on a grid Default: TRUE.
\texttt{phi0} numeric. In case of anisotropic fields directional cones are considered. The argument \texttt{phi0}
determines the starting angle.
Default: 0.
\texttt{pseudovariogram} logical. Only in the multivariate case. Whether the pseudovariogram or the
crossvariogram should be calculated.
Default: FALSE.
\texttt{theta0} numeric. In case of anisotropic fields directional cones are considered. The argument
\texttt{theta0} determines one of the boundaries, hence all boundaries for a given fixed number of
cones. The argument \texttt{theta0} determines the starting value of the second angle in polar
coordinate representation in 3 dimensions.
Default: 0.
\texttt{tol0} numeric. Estimated values of the empirical variogram below \texttt{tol0} times the grid step in the
third dimension are considered to be zero. Hence the respective values are set to zero.

8. \texttt{fit}: \textbf{Options for \texttt{RFfit}, \texttt{RFratiotest}, and \texttt{RFcrossvalidate}}

\texttt{algorithm} See \texttt{RFfitOptimiser}.
Default: NULL
\texttt{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 \texttt{approximate_functioncalls}.
Default: 50
\texttt{boxcox_lb} lower bound for the Box-Cox transformation
Default: -10.
\texttt{boxcox_ub} upper bound for the Box-Cox transformation
Default: 10.
\texttt{bin_dist_factor} numeric. The empirical variogram is calculated up the distance \texttt{bin_dist_factor}
times (maximum distance among any pair of locations)
Default: 0.5.
\texttt{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 \texttt{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 RFlkelihood.

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

max_neighbours integer. Maximum number of locations (with depending values) that are allowed.
   Default: 5000.

minbounddistance If any value of the parameter vector returned from the ML estimation is closer
   than minbounddistance to any of the bounds or if any value has a relative distance smaller
   than minboundreldist, then it is assumed that the MLE algorithm has dropped into a local
   minimum, and it will be continued with evaluating the ML target function on a grid, cf. the
   beginning paragraphs of the Details.
   Default: 0.001.
**minboundreldist**  
Relative distance to the bounds below which a part of the algorithm is considered as having failed. See `minbounddistance`.
Default: 0.02.

**min_diag**  
Minimal value of any estimated diagonal matrix element.
Default: 1e-7.

**n_crit**  
integer. The approximate profiles that are considered.
Default: 10.

**nphi**  
Scalar or vector of 2 components. If it is a vector then the first component gives the first angle of the xy plane and the second one gives the number of directions on the half circle. If scalar then the first angle is assumed to be zero. Note that a good estimation of the variogramm by LSQ with a anisotropic model a large value for `ntheta` might be needed (about 20).
Default: 1.

**ntheta**  
Scalar or vector of 2 components. If it is a vector then the first component gives the first angle in the third direction and the second one gives the number of directions on the half circle. If scalar then the first angle is assumed to be zero.
Note that a good estimation of the variogramm by LSQ with a anisotropic model a large value for `ntheta` might be needed (about 20).
Default: 1.

**ntime**  
Scalar or vector of 2 components. If `ntimes` is a vector, then the first component are the maximum time distance (in units of the grid length $T[3]$) and the second component gives the step size (in units of the grid length $T[3]$). If scalar then the step size is assumed to 1 (in units of the grid length $T[3]$).
Default: 20.

**only_users**  
Boolean. If true then only `users_guess` is used as a starting point for the fitting algorithms.
Default: FALSE.

**optimiser**  
See `RFfitOptimiser`.
Default: "optim".

**pgtol, pgtol_recall**  
See the argument `control` in `optim`. `pgtol_recall` is used for intermediate calculations.

**refine_onborder**  
Logical. If TRUE and an estimated parameter of the model is close to the boundary, a second search for the optimum is started.
Default: TRUE

**minmixedvar**  
Lower bound for variance in a mixed model; so, the covariance model for mixed model part might be calibrated appropriately.
Default: 1/1000

**ratiotest_approx**  
Logical. If TRUE the approximative formula that twice the difference of the likelihoods follow about a $\chi^2$ 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 \( \frac{\text{minimum distance between different pairs of points}}{\text{scale_max_relative_factor}} \), 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 \( \frac{\text{minimum distance between different pairs of points}}{\text{lowerbound_scale_ls_factor}} \).

Default: 1000

scale_ratio Rfit uses parscale and fnscale in the calls of optim. As these arguments should have the magnitude of the estimated values, Rfit 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: T.

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 Rfit 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.

clique.size integer. Rfit 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).

split_factor_neighbours The total number of neighbouring boxes in each direction \( 1+2\text{splitfactor} \), including the current box itself.

Default: 2.

split_refined logical. If TRUE then also submodels are fitted if splitted. This takes more time, but anova and RFRatiotest, for instance, will give additional information.

Default: TRUE.

upperbound_scale_factor The upper bound for the scale is determined as

\[ \text{upperbound_scale_factor} \times \text{(maximum distance between all pairs of points)} \]

Default: 3.

upperbound_var_factor The upper bound for the variance and the nugget is determined as 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 \( \text{cov}(1) \approx 0.05 \). use_naturalscaling has the advantage that scale and the form parameters of the model get ‘orthogonal’, but use_naturalscaling does not work for all models.

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

Advantages if use_naturalscaling=TRUE:
RFoptions

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

Disadvantages if use_naturalscaling=TRUE:
- For some covariance models with additional parameters, the rescaling factor has to be determined numerically. Then, more time is needed to perform RFFit.

Default: TRUE.

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 \( c(1,0) \)

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_screen logical. If FALSE the current device is kept as it is. Otherwise the action depends on the value of height: if height is not positive then close.screen is performed on the current device. Else the current device is closed.
Default: FALSE.

always_open_screen logical. if TRUE a new graphical window is opened for every plot if a standard graphical output is used. If NA then the value is set to interactive().
Default: TRUE.

grPrintlevel integer values 0, 1, 2. The higher the more text is shown in the plot.
Default: 1.
height: real number. If height is greater than zero then it gives the height of a single figure in a plot created by RandomFields; always a new window is opened. See also always_close_screen. If plots with multiple figures are shown, the height and width of the plot will be increased by a factor up the ones given by increase_up_to.
Default: 6.
pdffile: argument file in pdf If "" then no internal naming is performed.
Default: "".
jpegfile: argument file in jpeg If "" then no internal naming is performed.
Default: "".
filenumber: integer. Starting number of the file if onefile=FALSE. It is set to 0 whenever pdffile is changed and onefile=FALSE.
Default 0.
onefile: about the behaviour of argument onefile in pdf
Default: FALSE.
increase_up_to: See height.
Default: c(3, 4).

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. **krige: Options for Kriging**

- **cholesky_R** obsolete
- **fillall** logical value for imputing. If true all the components are estimated whether they are NA or not.
  
  Default: TRUE.
- **locmaxn** Kriging is conditions on maximal locmaxn points. If the data contain more points, neighbourhood kriging is performed.
  
  Default: 8000.
- **locsplitfactor** In case of neighbourhood kriging, the area is split into small boxes. The complete neighbourhood contains \((2 \times \text{locsplitfactor} + 1)\) 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 the interval \([10, 100]\).
  
  (For ease and for concerns of efficiency, the more adequate, local minimum is not considered.)
  
  Default: 30.
- **density_ratio** value in \([0, 1]\). This argument is considered only if flat=-1 and the simulation is performed on a grid. Then, the ratio between the highest and the lowest value is calculated within the convex hull of the grid. If the value is less than density_ratio then the grid points are considered separately. Else the density is considered to be constant in the convex hull of the grid.
  
  Default: 0.0.
- **eps_zhou** positive real number, which gives the aimed relative precision. E.g. if eps_zhou=0.01 then the first 2 digits should be correct.
  
  Default: 0.01
- **flat** -1, FALSE, TRUE. The argument is considered only if the simulation is performed on a grid.
  
  If flat is logical, then the density is considered to flat in the convex hull of the grid. If flat=-1 the choice is done automatically.
  
  Default: -1.
max_gauss The simulation of the max-stable process based on random fields uses a stopping rule that necessarily needs a finite upper endpoint of the marginal distribution of the random field. In the case of Brown-Resnick processes, extremal Gaussian fields, and extremal t fields, the upper endpoint is approximated by standardmax.
Default: 3.0.

max_n_zhou positive integer. The overall constant $c$ in the paper of Oesting, Schlather, Zhou (2014) has to be determined by MCMC, if the shape functions are random.
The two arguments, min_n_zhou and max_n_zhou, give the minimal and the maximal number of simulations that are performed. To economize computer time the values of $c$ is partially estimated when the shape functions are simulated. If the number of shape functions is larger than the number of simulations given by eps_zhou then no further simulation is performed to determine $c$.
Default: 1000 and 1000000, respectively.

maxpoints positive integer; the maximal number of Poisson points to be simulated for one realization of the max-stable random field. This option will not be considered for most of the users.
Default: 2e9.

mcmc_zhou positive integer. In case of random shape functions, an MCMC step is required. mcmc_zhou-1 equals the number of members of the MCMC chain that are left out before the next value of the chain is returned.
Default: 20

min_n_zhou see max_n_zhou

xi Extreme value index. Default: 2e9.

mcmc_zhou positive integer. In case of random shape functions, an MCMC step is required. mcmc_zhou-1 equals the number of members of the MCMC chain that are left out before the next value of the chain is returned.
Default: 20

min_n_zhou see max_n_zhou

xi Extreme value index.
Default: 1.0.

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

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_size,scatter_max Used in function RMscatter that calculates $\sum_{i=1}^{n} f(x + h_i)$ for some function $f$ and for some distances $h_i$.
Real valued and integer valued, respectively, or NA. Let $\varepsilon =$about_zero, $s =$scatter_size and $m =$scatter_max. We distinguish 4 cases:
• scatter_size > 0 and scatter_max >= 0
  Here, $n$ equals $(2m)^d$, and $h_i \in M = \{(ks,...,ks),..., (ms,...,ms)\}$ with $k = -m$.  

RFoptions

- scatter_size > 0 and scatter_max < 0
  same as the previous case, but \( m \) is chosen such that \( f(k_i c_i s_i) \approx \varepsilon \), \( -k_i \in \mathbb{N} \), \( i = 1, \ldots, d \) and \( f(m_i c_i s_i) \approx \varepsilon \), \( m \in \mathbb{N} \).
- 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 scatter_max steps.
- scatter_size <= 0 and scatter_max < 0
  This option is possible only for grids. Here, \( h_i \) runs over the whole grid.

shape_power  Shape functions are powered by shape_power before used as intensity function for the point process.
Default: 2.0.

16. nugget: Options for the nugget effect
Simulating a nugget effect is per se trivial. However, it gets complicated and best methods (including direct and circulant embedding!) fail if zonal anisotropies are considered, where sets of points have to be identified that belong to the same subspace of eigenvalue 0 of the anisotropy matrix.
tol  The nugget tolerance influences two different kind of models
  - RPnugget
  - R.is
See there for more information.

17. registers: Register numbers
Model for different purposes are or can be stored at different places. They are called registers and have non-negative numbers up to 21 (currently). The user can use the registers 0..9.
register number in 0:9; place where intermediate calculation for random field simulation are stored; the number refers to 10 internal registers 0..9.
Changing the register number only makes sense, when two different random fields, say, are to be simulated alternatly, several times in a row. Then the simulation speed can be increased if several registers are used, storing=TRUE and RFsimulate is used with the only argument n.
Default: 0

18. sequ: Options for the sequential method
back_steps  See RPsequential
initial  See RPsequential
max_variables  See RPsequential

19. solve: Options for solving linear systems
max_chol  integer. Maximum number of rows of a matrix in a Cholesky decomposition
  Default: 8192
max_svn  integer. Maximum number of rows of a matrix in a svd decomposition
  Default: 6555
matrix_methods vector of at most 3 integers that gives the sequence of methods in order to inverse a matrix or to calculate its square root:
0 : Choleskey decomposition
1 : SVD
2 : spam (sparse matrix algorithm)
3 : OR
4 : LU
5 : <none>
Note that if use_spam is not false the algorithm checks whether a sparse matrix algorithm should be used and which is then tried first.
Values larger than 4 are used internally:
5 : no further method available
6 : not initialised
7 : diagonal matrix found

Default: 5.

spam_factor integer. See argument spam_sample_n.
Default: 4294967

spam_min_n integer. Has the matrix
Default: 400

spam_min_p number in (0, 1) giving the proportion of zero about which an sparse matrix algorithm is used.
Default: 0.8

See package spam for details.
Default: 1

spam_sample_n Whether a matrix is sparse or not is tested by a ‘random’ sample of size spam_sample_n; The selection of the sample is iteratively obtained by multiplying the index by spam_factor modulo the size of the matrix.
Default: 500.

spam_tol largest absolute value being considered as zero. Default: DBL_EPSILON

svd_tol When the svd decomposition is used for calculating the square root of a matrix then the absolute componentwise difference between this matrix and the square of the square root must be less than svd_tol. No check is performed if svd_tol is negative.
When the svd decomposition is used for calculating the inverse of a matrix then a diagonal value is set to zero if it is less than svd_tol.
Default: 1e-8

use_spam Should the package spam (sparse matrices) be used for matrix calculations? If TRUE spam is always used. If FALSE, it is never used. If NA its use is determined by the size and the sparsity of the matrix.
Default: NA.

20. special: Options for specific methods
multicopies Only used by \texttt{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 \texttt{multicopies} products of random fields is calculated.

21. spectral: Options for the spectral (turning bands) method

\texttt{ergodic} In case of an additive model and \texttt{ergodic=FALSE}, the additive component are chosen proportional to their variance. In total lines are simulated. If \texttt{ergodic=TRUE}, the components are simulated separately and then added.

Default: \texttt{FALSE}.

\texttt{prop\_factor} see \texttt{RPspectral}

\texttt{sigma} see \texttt{RPspectral}

\texttt{sp\_grid} see \texttt{RPspectral}

\texttt{sp\_lines} see \texttt{RPspectral}

22. tbm: Options for the turning bands method

center Scalar or vector. If not NA, the center is used as the center of the turning bands for \texttt{TBM2} and \texttt{TBM3}. Otherwise the center is determined automatically such that the line length is minimal. See also \texttt{points} and the examples below.

Default: NA.

\texttt{fulldim} positiv integer. The dimension of the space into which the simulated field is embedded. So, the value \texttt{fulldim} must be at least the dimension of the field.

Default: 3.

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

This option is used by both \texttt{RPspectral} and \texttt{RPTbm}, the latter only when the dimension is 2.

Default: \texttt{TRUE}.

\texttt{layers} Logical or integer. If \texttt{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 \texttt{FALSE} then turning layers may never be used.

Default: \texttt{TRUE}.

\texttt{lines} Number of lines used.

Default: 60.

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

Default: 0.0.

\texttt{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 \texttt{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}.

Default: 2.0.
points integer. If greater than 0, points gives the number of points simulated on the TBM line, hence must be greater than the minimal number of points given by the size of the simulated field and the two parameters TBMx.linesimufactor and TBMx.linesimustep. If points is not positive the number of points is determined automatically. The use of center and points is highlighted in an example below.
Default: 0.

reduceddim if positiv integer, then the value itself. If negativ, then the value is substracted from fulldim.
Default: -2.

**23. internal:** Internal options mostly for warnings and messages

All these options should not be changed by the user unless he/she really known what he/she is doing.

Most of the options below change their value in a session without the user’s notice.

do_tests Internal variable. Do not use it. Default: FALSE.

examples_reduced non-negative integer. If positive, then the design of any simulation in RandomFields is internally reduced in size (roughly down to the given value in each direction). Warnings report this behaviour. This option is necessary to run the examples of RandomFields under the time constraint of CRAN.

stored.init internally used logical argument. This option is closely related to storing which controls whether intermediate calculations should be stored to have faster repeated simulations.

This user option is internally overwritten if the user calls several simulations at once. This current value is stored in stored.init.
Default: FALSE.

warn_ambiguous internally used logical argument. Usually, the argument grid in RFsimulate, for instance, can or should be given. If not given, the system takes a default definition. Additionally a message is displayed in this case if ambiguous=TRUE.
Default: FALSE.

warn_aspect_ratio internally used logical argument. if TRUE then a warning is given not a standard graphical device is used and the package plots try to keep a certain aspect ratio.
Default: TRUE.

warn_colour_palette internally used logical argument. If none of the packages RColorBrewer and colorspace are available and graphics are displayed, a message is displayed.
Default: TRUE.

warn_constant The definition of RMconstant has changed. A warning is displayed if the command is used. warn_constant will become obsolete in future versions.
Default: TRUE.

warn_coordinates internally used logical argument. If TRUE then a transformation from earth coordinates to cartesian coordinates is reported.
Default: TRUE.

warn_missing_zenit Only for Earth systems: a missing zenith 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 argu-
  ment spConform=FALSE oldstyle return values are obtained instead of S4 objects.
  Default: TRUE.

warn_normal_mode internally used logical argument. if TRUE then the function Rffit displays the
  message that other values for the option modus_operandi are available.
  Default: TRUE.

warn_oldstyle internally used logical argument. If TRUE a warning is given if an obsolete function
  from Version 2 is used.
  Default: TRUE.

warn_on_grid internally used logical argument. If a (one-dimensional) grid is given, but the argu-
  ment grid=FALSE, e.g. in Rfsimulate, this contraction is reported if warn_on_grid=TRUE
  Default: TRUE.

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

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

Value

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

if no.readonly=FALSE then additionally, a list called readonly is included containing
  * covmaxchar: the maximum length of a model name
  * covnr: number of currently implemented variogram/covariance models (-1 means that none of the functions like Rfsimulate, Rffit, etc., have been called yet.)
  * distrmaxchar: max. name length for a distribution
  * distrnr: number of currently implemented distributions
  * maxdim: maximum number of dimensions for a random field
  * maxmodels: maximum number of elementary models in definition of a complex covariance model
  * methodmaxchar: max. name length for methods
  * methodnr: number of currently implemented simulation methods

Author(s)

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

• General

• rectangular distribution; eps_zhou

• shape_power

See Also

`RFsimulate, RFoptionsAdvanced, RandomFields, and RFgetMethodNames.`

Examples

```r
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 <- RHgauss()

for (exactness in c(NA, FALSE, TRUE)) {
  readline(paste("\n\nexactness: ", exactness, "; press return\n"))
  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.

Examples

```r
# 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

len <- 10

x <- seq(0, 1, len=len)
y <- seq(0, 1, len=len)
grid.size <- c(length(x), length(y))
meth <- RPcirculant
model <- RMexp(var=1.1, Aniso=matrix(nc=2, c(2,0.1,1.5,1)))
m <- 5
n <- 100

c1 <- numeric(m)
time <- unix.time{
  for (i in 1:m){
    cat("", i, " out of", m, "\n")
    z <- RFsimulate(meth(model), x, y, n=n, pch="",
                  dependent=FALSE, spConform=FALSE, trials=5, force=TRUE)
    c[i] <- cov(z[1, dim(z)[2]], z[dim(z)[1], 1,])
  }
}

true.cov <- RFCov(model, t(y[c(1, length(y))]), t(x[c(length(x), 1)]))
print(time)
Print(true.cov, mean(c1), sd(c1), empty.lines=1)# true mean is zero
```
# using local.dependent=TRUE ...
c2 <- numeric(m)
time <- unix.time(
  for (i in 1:m) {
    cat("", i)
    z <- RFsimulate(meth(model), x, y, n2 * n, pch="",
                    dependent=TRUE, spConform=FALSE, trials=5, force=TRUE)
    c2[i] <- cov(z[1, dim(z)[2]], z[2, 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.

################################################################################
## EXAMPLE 2  ##
## This example shows that the same realisation can be        ##
## obtained on different grid geometries (or point        ##
## configurations, i.e. grid, non-grid) using TBM             ##
################################################################################

step <- 1
x1 <- seq(-150, 150, step)
y1 <- seq(-15, 15, step)
x2 <- seq(-0.5, 0.5, step)
model <- RPtbm(RMexp(scale=10))
RFoptions(storing=TRUE)
m <- c(2, 2, 2, 0.1, 0.1)
points <- 700

## simulation of a random field on long thin stripe
z1 <- RFsimulate(model, x1, y1, center=0, seed=0,
                 points=points, storing=TRUE, spConform=FALSE)
ScreenDevice(height=1.55, width=12)
par(mar=m)
image(x1, y1, z1, col=rainbow(100))
polygon(range(x2)[c(1,2,2,1)], range(y1)[c(1,1,2,2)],
        border="red", lwd=3)

## definition of a random field on a square of shorter diagonal
z2 <- RFsimulate(model, x2, x2, register=1, seed=0,
                 center=0, points=points, spConform=FALSE)
RFpar

Graphical parameters for plots

Description

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

Usage

RFpar(...)  

Arguments

... see par

Value

If RFpar is called without arguments, the current list is returned.
If RFpar is called with NULL only, the current list is deleted.
Otherwise the arguments are stored for global use in RandomFields.

Author(s)

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

See Also

plot-method
RFpointsDataFrame-class

Class RFpointsDataFrame

Description

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

Creating Objects

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

Slots

data: object of class `data.frame`, containing attribute data
coords: n-times-1 matrix of coordinates (each row is a point)
.RFparams: list of 2; .RFparams$n is the number of repetitions of the random field contained in the data slot, .RFparams$vdime 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`.

Examples

```r
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

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

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

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

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

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

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

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

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

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

Details

Methods summary and dimensions are defined for the “parent”-class RFsp.

Author(s)

Alexander Malinowski <Alexander.Malinowski@web.de>

See Also

RFspatialPointsDataFrame, which is for point locations in higher dimensional spaces, RFpointsDataFrame-class which is for one-dimensional locations on a grid, RFsp

Examples

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

x <- runif(100)
f <- RFsimulate(model=RMexp(), x=x, n=3)

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

plot(f, nmax=2)
RFratiotest  
LIKELIHOOD RATIO TEST

Description

The function performs an approximate $\chi^2$ test or a Monte Carlo likelihood ratio test based on fitgauss. Currently it only works for Gaussian random fields.

Usage

RFratiotest(nullmodel, alternative, x, y = NULL, z = NULL, T = NULL, grid=NULL, data,
alpha, n = 5 / alpha, seed = 0,
lower = NULL, upper = NULL, methods,
sub.methods, optim.control = NULL, users.guess = NULL,
distances = NULL, dim, transform = NULL, ...)

Arguments

nullmodel, alternative
See Details.

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 ROptions()$general$seed if the latter is not NA.

x, y, z, T, grid, data, lower, upper, methods, sub.methods, optim.control, users.guess, distances, ...  
see RFit

Details

nullmodel (and the alternative) can be

• a covariance model, see RMmodel or type RFgetModelNames(type="variogram") to get all options.

Depending weather the RFOptions ratiotest_approx is TRUE the the chisq approximation is performed. Otherwise a Monte Carlo ratio test is performed.

• Rfit or RMmodelFit

Here, a chisq approximative test is always performed on the already fitted models.

RFratiotest tries to detect whether nullmodel is a submodel of alternative. If it fails,

• a message is printed that says that an automatic detection has not been possible;
• it is not guaranteed anymore that the alternative model returns a (log) likelihood that is at least as large as that of the nullmodel, even if nullmodel is a submodel of alternative. This is due to numerical optimisation which is never perfect.

Otherwise it is guaranteed that the alternative model has a (log) likelihood that is at least as large as that of the nullmodel.

Value

The test returns a message whether the null hypothesis, i.e. the smaller model is accepted. Invisibly, a list that also contains

• p, the p-value
• n
• data.ratio the log ratio for the data
• simu.ratio the log ratio for the simulations
• data.fit the models fitted to the data
• msg the message that is also directly returned

It has S3 class "RFratiotest".

Methods

print prints the summary
summary gives a summary

Note

An important RFoptions is ratiotest.approx.

Note

Note that the likelihood ratio test may take a huge amount of time.

Note

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

Author(s)

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

See Also

RFfit RMmodel, RandomFields, weather.
Examples

RFsimulate

Simulation of Random Fields

Description

This function simulates unconditional random fields:

- univariate and multivariate, spatial and spatio-temporal Gaussian random fields
- fields based on Gaussian fields such as Chi2 fields or Binary fields, see RP.
- stationary Poisson fields
- stationary max-stable random fields.

It also simulates conditional random fields for

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

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

Usage

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

Arguments

model object of class RMmodel, RFformula or formula: specifies the model to be simulated; the best is to consider the examples below, first.

- if of class RMmodel, model specifies a covariance or variogram model of a Gaussian random field; type RFgetModelNames(type="variogram") for a list of available models; see also RMmodel
- if of class RFformula or formula, submodel specifies a linear mixed model where random effects can be modelled by Gaussian random fields; see RFformula for details on model specification.
- for (many) more options see RFsimulateAdvanced.

x vector of x coordinates, or object of class GridTopology or raster; For more options see RFsimulateAdvanced.

y optional vector of y coordinates

z optional vector of z coordinates
`RFsimulate` 149

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

grid logical; RandomFields can find itself the correct value in nearly all cases, so that usually grid need not be given. See also `RFsimulateAdvanced`.

distances another alternative to pass the (relative) coordinates, see `RFsimulateAdvanced`.

dim Only used if distances are given.

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

Matrix, data.frame or object of class `RFsp`; coordinates and response values of measurements in case that conditional simulation is to be performed; If given is not given and data is a matrix or data is a data.frame, then the first columns are interpreted as coordinate vectors, and the last column(s) as (multiple) measurement(s) of the field; if the argument x is missing, data may contain NAs, which are then replaced by conditionally simulated values (random imputing); for details on matching of variable names see Details; if of class `RFsp` 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.

given If given, provided, data must be a matrix or an array containing the data only.

err.model For conditional simulation and random imputing only. Usually `err.model=RMnugget(var=var)`, or not given at all (error-free measurements).

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

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

Details

By default, all Gaussian random fields have zero mean. Simulating with trend can be done by including `Rmtrend` in the model, see the examples below.

If data is passed, conditional simulation based on simple kriging is performed:

- if of class `RFsp`, `ncol(data@coords)` must equal the dimension of the index space. If `data@data` contains only a single variable, variable names are optional. If `data@data` contains more than one variable, variables must be named and model must be given in the tilde notation `resp ~ ...` (see `Rfformula`) and "resp" must be contained in `names(data@data)`.
- If data is a matrix or a data.frame, either `ncol(data)` equals \((\text{dimension of index space} + 1)\) and the order of the columns is (x, y, z, T, response) or, if data contains more than one response variable (i.e. `ncol(data) > (\text{dimension of index space} + 1)`), `colnames(data)` must contain `colnames(x)` or those of "x", "y", "z", "T" that are not missing. The response variable name is matched with model, which must be given in the tilde notation. If "x", "y", "z", "T" are missing and data contains NAs, `colnames(data)` must contain an element which starts with 'data'; the corresponding column and those behind it are interpreted as the given data and those before the corresponding column are interpreted as the coordinates.
• if x is missing, \texttt{RFsimulate} searches for NAs in the data and performs a conditional simulation for them.

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

\textbf{Value}

By default, an object of the virtual class \texttt{RFsp}; result is of class \texttt{RMmodel}.

• \texttt{RFspatialGridDataFrame} if the space-time dimension is greater than 1 and the coordinates are on a grid,
• \texttt{RFgridDataFrame} if the space-time dimension equals 1 and the coordinates are on a grid,
• \texttt{RFspatialPointsDataFrame} if the space-time dimension is greater than 1 and the coordinates are not on a grid,
• \texttt{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 \texttt{RFsimulateAdvanced} for additional options.

\textbf{Note}

Several advanced options can be found in sections ‘General options’ and ‘coords’ of \texttt{RFoptions}. In particular, option \texttt{spConform=FALSE} leads to a simpler (and faster!) output, see \texttt{RFoptions} for details.

\textbf{Author(s)}

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

\textbf{References}


See \texttt{RFsimulateAdvanced} for more specific literature.

\textbf{See Also}

\texttt{RFempiricalvariogram,RFfit,RFgetModelInfo,RFgui,RMmodel,RFoptions,RFsimulateAdvanced,RFsimulate.more.examples}
RFsimulate

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 a
# 100 random locations:
n <- 100
x <- runif(n, min=-1, max=1)
y <- runif(n, min=-1, max=1)
data <- RFsimulate(model = RMexp(), x=x, y=y, grid=FALSE)
plot(data)

# let simulate a field conditional on the above data
x.seq.cond <- y.seq.cond <- seq(-1.5, 1.5, length=n)
model <- RMexp()
cond <- RFsimulate(model, x=x.seq.cond, y=y.seq.cond, data=data)
plot(cond, data)

---

**RFsimulate.more.examples**

*Further Examples for the Simulation of Random Fields*

---

**Description**

This man page will give a collection of basic examples for the use of **RFsimulate**.

For other kind of random fields (binary, max-stable, etc.) or more sophisticated approaches see **RFsimulateAdvanced**.

See **RFsimulate.sophisticated.examples** for further examples.

**See Also**

**RFsimulate, RFsimulateAdvanced**

**RFsimulate.more.examples**

*Further Examples for the Simulation of Random Fields*

---

**Examples**

RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
##
RFoptions(seed=NA) to make them all random again
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 function in the list are 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=NULL, x_ij is the i-th coordinate in the j-th dimension; otherwise, if grid=TRUE, the columns of x are interpreted as gridtripples (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

gid logical; determines whether the vectors x, y, and z or the columns of x should be interpreted as a grid definition (see Details). If grid=TRUE, either x, y, and z must be equidistant vectors in ascending order or the columns of x must be given in the gridtriple format: c(from, stepsize, len).

Note: if grid is not given, RFsimulate tries to guess what is meant.

c(from, stepsize, len) (see Details)

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

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

distances object of class dist representing the upper 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
RFsimulateAdvanced

n
number of realizations to generate

... further options and control arguments for the simulation that are passed to and processed by \texttt{RfOptions}

Details

\texttt{RFsimulate} simulates different classes of random fields, controlled by the wrapping model.

If the wrapping function of the \texttt{model} argument is a covariance or variogram model (i.e., one of list obtained by \texttt{RFgetModelsNames(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 \texttt{method specification}.

If other than Gaussian fields are to be simulated, the \texttt{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 \texttt{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 \texttt{cbind(x, y, z)}.

If \(T\) is not missing, the field is simulated for all combinations \((x[i], T[k])\) or \((x[i], y[i], z[i], T[k])\), \(i = 1, \ldots, \text{rown}(x), k = 1, \ldots, \text{length}(T)\), even if \texttt{model} is not explicitly a space-time model.

If \texttt{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 \texttt{expand.grid(x, y, z, T)}.

There, “grid” means “equidistant in each direction”, i.e. all vectors must be equidistant and in ascending order. In case of more than 3 space dimensions, the coordinates must be given in matrix notations. To enable different grid lengths for each direction in combination with the matrix notation, the “gridtriple” notation \(c(from, stepsize, len)\) is used: If \(x, y, z, T\) or the columns of \(x\) are of length 3, they are internally replaced by \texttt{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 ~ ...} (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 \(\text{dimension of index space} + 1\) and the order of the columns is \((x, y, z, T, \text{response})\) or, if \texttt{data} contains more than one response variable (i.e. \texttt{ncol(data)} > \(\text{dimension of index space} + 1\)), \texttt{colnames(data)} must contain \texttt{colnames(x)} or those of "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 "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{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 $err_{\text{model}}$ must be given separately. For sake of generality, any model (and not only the nugget effect) is allowed. Consequently, $err_{\text{model}}$ is ignored when unconditional simulation is performed.

**Value**

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

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

if $vdim > 1$ the $vdim$-variate vector makes the first dimension

if grid=TRUE an array of the dimension of the random field makes the next dimensions. Here, the dimensions are ordered in the sequence x, y, z, T (if given).

Else if no time component is given, then the values are passed as a single vector. Else if the time component is given the next 2 dimensions give the space and the time, respectively.

if $n > 1$ the repetitions make the last dimension

Note: Conversion between the sp format and the conventional format can be done using the method RFspDataFrame2conventional and the function conventional2RFspDataFrame. InitRFsimulate returns 0 if no error has occurred and a positive value if failed.

**Note**

Advanced options are

- spConform (suppressed return of S4 objects)
- practicalrange (forces range of covariances to be one)
- exactness (chooses the simulation method by precision)
- seed (sets .Random.seed locally or globally)

See RFoptions for further options.

**Author(s)**

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

**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:

• Random coins:

See Also

*RFoptions, RMmodel, RFgui, methods for simulating Gaussian random fields, RFFit, RFempiricalvariogram, RFsimulate.more.examples, RFsimulate.sophisticated.examples, RPgau,**
Examples

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

---

**RFsp-class**  
*Class RFsp*

**Description**

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

The first two class subclasses are summarized in "RFspatialDataFrame" whilst the latter two are summarized in "RFgridDataFrame".

**Objects from the Class**

are never to be generated; only derived classes can be meaningful

**Methods**

- **summary** signature(obj = "RFsp"): returns a summary of the object; uses or imitates summary method of class Spatial from the sp-package
- **dimensions** signature(obj = "RFsp"): retrieves the number of spatial or spatio-temporal dimensions spanned
- **isGridded** signature(obj = "RFsp"): logical, tells whether the data is on a regular spatial grid
- **[** signature(obj = "RFsp"): selects columns of the data-slot, while all other slots are kept unmodified
- **[<-** signature(obj = "RFsp"): replaces columns of the data-slot, while all other slots are kept unmodified
- **variance** signature(object = "RFsp"): returns the kriging variance if available

**Warning**

this class is not useful in itself, but the above mentioned classes in this package derive from it

**Author(s)**

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

RFspatialGridDataFrame, RFspatialPointsDataFrame, RFgridDataFrame, RFpointsDataFrame, sp2RF

Examples

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

RFSp2conventional    coerce RFsp-class to conventional format

Description

c coerce RFsp-class to conventional format

Usage

RFSpDataFrame2conventional(obj)

RFSpDataFrame2dataArray(obj)

Arguments

obj

object of class RFsp.

Value

RFSpDataFrame2conventional returns a list; RFSpDataFrame2dataArray returns an array containing the data-slot of obj;

Author(s)

Alexander Malinowski <Alexander.Malinowski@web.de>

See Also

RFspatialGridDataFrame, RFspatialPointsDataFrame
Examples

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

n <- 3
x <- GridTopology(cellcentre.offset=c(0, 0),
cells.size=c(1, 0.2),
cells.dim=c(10, 30))
f <- RFsimulate(model=RMexp(), x=x, n=n)
str(f)
str(RFspDataFrame2conventional(f))
Methods

contour code signature(obj = "RFspatialGridDataFrame"): generates contour plots

plot signature(obj = "RFspatialGridDataFrame"): generates nice image plots of the random field; if \(space - time - dim\), 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 code signature(obj = "RFspatialGridDataFrame"): generates persp plots

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

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

RFspDataFrame2conventional signature(obj = "RFspatialGridDataFrame"): conversion to a list of non-sp-package based objects; the data-slot is converted to an array of dimension \([1*(vdim > 1) + space - time - dimension + 1*(n > 1)]\); the grid-slot is converted to a 3-row matrix; the grid definition of a possible time-dimension becomes a separate list element

RFspDataFrame2dataArray signature(obj = "RFspatialGridDataFrame"): conversion of the data-slot to an array of dimension \([space - time - dimension + 2]\), where the space-time-dimensions run fastest, and \(vdim\) and \(n\) are the last two dimensions

coordinates signature(x = "RFspatialGridDataFrame"): calculates the coordinates from grid definition

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

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

as signature(x = "RFspatialGridDataFrame"): converts into other formats, only implemented for target class RFspatialPointsDataFrame

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

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

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

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

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

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

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

Details

Note that in the data-slot, each column is ordered according to the ordering of coordinates(grid), the first dimension runs fastest and for all BUT the second dimension, coordinate values are in ascending order. In the second dimension, coordinate values run from high to low. Hence, when converting to conventional formats using RFspDataFrame2conventional or RFspDataFrame2dataArray, the data array is re-ordered such that all dimensions are in ascending order. as.matrix does not perform re-ordering.

Methods summary, dimensions and isGridded are defined for the “parent”-class RFsp.
Author(s)

Alexander Malinowski <Alexander_Malinowski@web.de>, Martin Schlather, schlather@math.uni-mannheim.de

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

See Also

RFspatialPointsDataFrame-class, which is for point locations that are not on a grid, RFgridDataFrame-class which is for one-dimensional locations, RFsp, sp2RF

Examples

ROptions(seed=0) ## ANY* simulation will have the random seed 0; set
##            ROptions(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(RFspDataFrame$conventional(f))
str(RFspDataFrame$dataArray(f))
coordinates(f)[1:25,]
str(f[2]) ## selects second column of data-slot
all.equal(f, cbind(f,f)[1:3]) ## TRUE
str(as(f, "RFspatialPointsDataFrame"))

plot(f, nmax=2)
steps <- 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, seed=0)

f.old <- RFSimulate(model=RMexp(), x=x, n=n, spConform=FALSE, seed=0)
all.equal(RFspDataFrame$conventional(f.sp)$data, f.old) ## TRUE
RFspatialPointsDataFrame-class
Class "RFspatialPointsDataFrame"

Description
Class for spatial attributes that have spatial or spatio-temporal locations (at least of dimension 2) that are not on a grid. Direct extension of class SpatialPointsDataFrame from the sp-package. See sp2RF for an explicite transformation.

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

Slots
.RFparams: list of 2; .RFparams$n is the number of repetitions of the random field contained in the data slot, .RFparams$vdim gives the dimension of the values of the random field, equals 1 in most cases
data: object of class data.frame, containing attribute data
coords.nrs: See SpatialPointsDataFrame
coords: matrix of coordinates (each row is a point); in case of SpatialPointsDataFrame an object of class SpatialPoints is also allowed see SpatialPoints
bbox: matrix specifying the bounding box
proj4string: object of class CRS; projection

Extends
Class SpatialPointsDataFrame, directly. Class SpatialPoints, by class SpatialPointsDataFrame. Class Spatial, by class SpatialPoints.

Methods
plot signature(obj = "RFspatialPointsDataFrame"); generates nice plots of the random field; if space−time−dim2, a two-dimensional subspace can be selected using the argument MARGIN; to get different slices in a third direction, the argument MARGIN.slices can be used; for more details see plot-method or type method?plot("RFspatialPointsDataFrame")
show signature(x = "RFspatialPointsDataFrame"); uses the show-method for class SpatialPointsDataFrame.
print signature(x = "RFspatialPointsDataFrame"): identical to show-method
RFspDataFrame2conventional signature(obj = "RFspatialPointsDataFrame"); conversion to a list of non-sp-package based objects; the data-slot is converted to an array of dimension $[1 \times (vdim > 1) + space−time−dimension + 1 \times (n > 1)]$
coordinates signature(x = "RFspatialPointsDataFrame"): returns the coordinates
[ signature(x = "RFspatialPointsDataFrame"): selects columns of data-slot; returns an object of class RFspatialPointsDataFrame.

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

as signature(x = "RFspatialPointsDataFrame"): converts into other formats, only implemented for target class RFspatialGridDataFrame

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

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

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

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

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

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

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

Details

Note that in the data-slot, each column is ordered according to the ordering of coordinates(grid), the first dimension runs fastest and for all BUT the second dimension, coordinate values are in ascending order. In the second dimension, coordinate values run from high to low. Hence, when converting to conventional formats using RFspDataFrame2conventional or RFspDataFrame2dataArray, the data array is re-ordered such that all dimensions are in ascending order. as.matrix does not perform re-ordering.

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

Author(s)

Alexander Malinowski <Alexander.Malinowski@web.de>

See Also

RFspatialGridDataFrame-class, which is for point locations that are on a grid, RFpointsDataFrame-class which is for one-dimensional locations, RFsp, sp2RF

Examples

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

x <- cbind(runif(50), runif(50))
f <- RFsimulate(model=RMexp(), x=x, n=3)

str(f)
str(RFspDataFrame2conventional(f))
coordinates(f)[1:25,]
str(f[2]) ## selects second column of data-slot
all.equal(f, cbind(f,f)[1:3]) ## TRUE
try(as(f, "RFspatialGridDataFrame")) # yields error

plot(f, nmax=2)

f2 <- Rfsimulate(model=RMwhittle(nu=1.2, scale=10), x=cbind(x,x), n=4)
plot(f2, MARGIN=c(3,4), nmax=2)

f.sp <- Rfsimulate(model=RMexp(), x=x, n=3, seed=0)
f.old <- Rfsimulate(model=RMexp(), x=x, n=3, spConform=FALSE, seed=0)
all.equal(RFspDataFrame2conventional(f.sp)$data, f.old) ## TRUE

---

**RMangle**

*Anisotropy matrix given by angle*

**Description**

RMangle delivers an anisotropy matrix for the argument Aniso in RMmodel in two dimensions. RMangle requires one two streching values, passed by ratio or diag, and an angle.

In two dimensions and with angle equal to \( \alpha \) and diag equal to \((d_1, d_2)\) the anisotropy matrix \( A \) is

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

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

\[
A = \text{diag}(d_1, d_2, d_3) \text{ \%\% matrix}(\text{ncol}=3, \text{c}(\cos(\alpha) \times \cos(L), \sin(\alpha) \times \cos(L), \sin(L), -\sin(\alpha), \cos(\alpha) \times \sin(L), -\sin(L), \cos(L)))
\]

i.e. \( Ax \) turns a vector \( x \) first in \( x-z \) plane, then in the \( x-y \) plane.

**Usage**

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

**Arguments**

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

**Value**

RMangle returns an object of class RMmodel.

**Author(s)**

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

RMtrafo, RMmodel

Examples

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

    model <- RMexp(Aniso=RMangle(angle=pi/4, ratio=3))
    plot(model, dim=2)

    x <- seq(0, 2, 0.05)
    z <- RFSimulate(x, x, model=model)
    plot(z)

    model <- RMexp(Aniso=RMangle(angle=pi/4, lat.angle=pi/8, diag=c(1,2,3)))
    x <- seq(0, 2, 0.2)
    z <- RFSimulate(x, x, x, model=model)
    plot(z, MARGIN.slices=3)

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

RMaskey

Askey model

Description

Askey’s model

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

Usage

RMaskey(alpha, var, scale, Aniso, proj)
RMtento(var, scale, Aniso, proj)
Arguments

alpha    a numerical value in the interval [0,1]
var, scale, Aniso, proj
optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above
covariance function remains unmodified.

Details

This covariance function is valid for dimension $d$ if $\alpha \geq (d + 1)/2$. For $\alpha = 1$ we get the well-
known triangle (or tent) model, which is valid on the real line, only.

Value

\texttt{RMaskey} returns an object of class \texttt{RMmodel}

Author(s)

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

References

Covariance function


Applications as covariance function


Tail correlation function (for $\alpha \geq [d/2] + 1$)


See Also

\texttt{RMmodel}, \texttt{RMbigneiting}, \texttt{RMgengneiting}, \texttt{RMgneiting}, \texttt{RFsimulate}, \texttt{RFit}.  

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

\[
C(h, u) = |E + 2Ah^tA|^{-1/2}\phi(\sqrt{||h||^2/2 + (z^t h + u)^2}(1 - 2h^t A(E + 2Ah^tA)^{-1} Ah))
\]

where \(E\) is the identity matrix. The spatial dimension is \(d - 1\) and \(h\) is real-valued.

Usage

RMave(phi, A, z, spacetime, var, scale, Aniso, proj)

Arguments

- \(\phi\): a covariance model which is a normal mixture, that means an RMmodel whose monotone property equals 'normal mixture', see RFgetModelNames(monotone="normal mixture")
- \(A\): a symmetric \(d - 1 \times d - 1\)-matrix if the corresponding random field is in the \(d\)-dimensional space
- \(z\): a \(d - 1\) dimensional vector if the corresponding random field is on \(d\)-dimensional space
- \(\text{spacetime}\): logical. If FALSE then the model is interpreted as if \(h = 0\), i.e. the spatial dimension is \(d\). Default is TRUE
- \(\text{var, scale, Aniso, proj}\): optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

See Schlather, M. (2010), Example 13 with l=1)
Value

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

Author(s)

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

References


See Also

`RFFit`, `RFsimulate`, `RMmodel`, `RMstp`

Examples

```r
ROptions(seed=0) #*ANY* simulation will have the random seed 0; set
ROptions(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+B))^(1/2) ) *
phi( sqrt( sum(h*h)/2 + (t(z) %*% h)^2 )
( 1-2*t(h) %*% A %*% solve(E+B) %*% A %*% h ) )
# stopifnot(abs(z1-z2)<1e-12, abs(z2-z3)<1e-12)
```
Description

RMbcw refers to the indicator function of a ball with radius 1.

Usage

RMbcw(var, scale, Aniso, proj)

Arguments

var, scale, Aniso, proj

optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Author(s)

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

See Also

RMpolygon, RMSpheric, RFSimulate, RMmodel.

Examples

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

Description

RMbcw is a variogram model that bridges between some intrinsically stationary isotropic processes and some stationary ones. It reunifies the RMgenfbm, RMdewijsian and RMgencauchy.

The corresponding centered semi-variogram only depends on the distance $r \geq 0$ between two points and is given by

$$\gamma(r) = \frac{(r^\alpha + 1)^{\beta/\alpha} - 1}{2^{\beta/\alpha} - 1}$$

where $\alpha \in (0,2]$ and $\beta \leq 2$. 
Usage

RMbcw(alpha, beta, var, scale, Aniso, proj)

Arguments

alpha a numerical value; should be in the interval (0,2].
beta a numerical value; should be in the interval (-infty,2].
var, scale, Aniso, proj optional arguments; same meaning for any Rmmodel. If not passed, the above variogram remains unmodified.

Details

For beta > 0, beta < 0, beta = 0 we have the generalised fractal Brownian motion RMgenfbm, the generalised Cauchy model RMgencauchy, and the de Wisjian model Rmdewijsian, respectively.
Hence its two arguments alpha and beta allow for modelling the smoothness and a wide range of tail behaviour, respectively.

Value

RMbcw returns an object of class Rmmodel

Author(s)

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

References


See Also

RMgenfbm, RMgencauchy, Rmdewijsian, Rmmodel, Rfsimulate, Rffit.

Examples

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

model <- RMbcw(alpha=1, beta=0.5)
x <- seq(0, 10, 0.02)
plot(model)
plot(Rfsimulate(model, x=x))
Mbernoulli

Covariance Model for binary field based on a Gaussian field

Description

Mbernoulli gives the centered correlation function of a binary field, obtained by thresholding a Gaussian field.

Usage

Mbernoulli(phi, threshold, correlation, centred, var, scale, Aniso, proj)

Arguments

phi   covariance function of class RMmodel.
threshold   real valued threshold, see RPbernoulli. Currently only threshold=0.0 is possible.
Default: 0.
correlation   logical. If FALSE the corresponding covariance function is returned
Default: TRUE.
centred   logical. If FALSE the uncentred covariance is returned.
Default: TRUE.
var, scale, Aniso, proj   optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model yields the covariance function of the field that is returned by RPbernoulli

Value

Mbernoulli returns an object of class RMmodel.

Note

Previous to version 3.0.33 the covariance function was returned, not the correlation function

Author(s)

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

References

Ballani, Schlather
**See Also**

`rmbessel`, `rmbernoulli`, `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")
```

---

**RMbessel**  

**Bessel Family Covariance Model**

**Description**

`RMbessel` is a stationary isotropic covariance model belonging to the Bessel family. The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

\[
C(r) = 2^\nu \Gamma(\nu + 1)r^{-\nu}J_\nu(r)
\]

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

**Usage**

`RMbessel(nu, var, scale, Aniso, proj)`

**Arguments**

- `nu`  
  a numerical value; should be equal to or greater than \( \frac{d-2}{2} \) to provide a valid covariance function for a random field of dimension \( d \).

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


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 \texttt{RMdampedcos} for $\lambda = 0$, there.

A second important case is $\nu = 0.5$ with covariance function

$$C(r) = \frac{\sin(r)}{r}$$

and which is valid for $d \leq 3$. This coincides with \texttt{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

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

Author(s)

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

References

• \url{http://homepage.tudelft.nl/11r49/documents/wi4006/bessel.pdf}

See Also

\texttt{RMdampedcos}, \texttt{RMwave}, \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 <- RMBessel(nu=1, scale=0.1)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFSimulate(model, x=x))
\end{verbatim}
**Description**

`RMBigneiting` is a bivariate stationary isotropic covariance model family whose elements are specified by seven parameters.

Let

\[ \delta_{ij} = \mu + \gamma_{ij} + 1. \]

Then,

\[ C_n(h) = c_{ij}(C_{n,\delta}(h/s_{ij})), i,j=1,2 \]

and \( C_{n,\delta} \) is the generalised Gneiting model with parameters \( n \) and \( \delta \), see `Rmgengneiting`, i.e.,

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

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

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

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

**Usage**

`RMBigneiting(kappa, mu, s, sred12, gamma, cdiag, rhored, c, var, scale, Aniso, proj)`

**Arguments**

- `kappa` argument that chooses between the four different covariance models and may take values 0, ..., 3. The model is \( k \) times differentiable.
- `mu` \( \mu \) has to be greater than or equal to \( d^2/2 \) 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 off diagonals is given by \( s_{12} = s_{21} = sred12 * \min\{s_{11}, s_{22}\}\).
- `gamma` a vector of length 3 of numerical values; each entry is positive. The vector `gamma` equals \( (\gamma_{11}, \gamma_{21}, \gamma_{22}) \). Note that \( \gamma_{12} = \gamma_{21} \).
- `cdiag` a vector of length 2 of numerical values; each entry positive; the vector \( (c_{11}, c_{22}) \)
- `c` a vector of length 3 of numerical values; the vector \( (c_{11}, c_{21}, c_{22}) \). Note that \( c_{12} = c_{21} \).
- Either `rhored` and `cdiag` or `c` must be given.
- `rhored` value in \([-1, 1]\). See also the Details for the corresponding value of \( c_{12} = c_{21} \).
- `var`, `scale`, `Aniso`, `proj` optional arguments; same meaning for any `RModel`. If not passed, the above covariance function remains unmodified.
Details

A sufficient condition for the constant $c_{ij}$ is

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

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

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

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

Let

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

If $d \geq 0$ and $t_j \notin (0, s_{12})$ then $m_j = \infty$ else

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

In the function \texttt{RMbigneiting}, either $c$ is passed, then the above condition is checked, or $\text{rhored}$ is passed then $c_{12}$ is calculated by the above formula.

Value

\texttt{RMbigneiting} returns an object of class \texttt{RMmodel}

Author(s)

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

References

- Bevilacqua, M., Daley, D.J., Porcu, E., Schlather, M. (2012) Classes of compactly supported correlation functions for multivariate random fields. Technical report. \texttt{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 \texttt{RMbigneiting}, in particular, neither referring to this original work nor to \texttt{RandomFields}, which has included \texttt{RMbigneiting} since version 3.0.5 (05 Dec 2013).
**RMbiwm**

### See Also

RMaskey, RMbiwm, RMgengneiting, RMgneiting, 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 <- 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))
```

---

**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 $W_{\nu}$ is the covariance of the **RMwhittle** model. For constraints on the constants see details.

### Usage

```r
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 $0.5(\nu_{11} + \nu_{22}) * \nu_{red}$.
- **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 \texttt{(RMwhittle)} model applies. If logical then the formula for the \texttt{RMMatern} model applies. See there for details.

\texttt{var, scale, Aniso, proj}

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

\section*{Details}

Constraints on the constants: For the diagonal elements we have

$$\nu_{ii}, s_{ii}, c_{ii} > 0.$$  

For the offdiagonal elements we have

$$s_{12} = s_{21} > 0,$$

$$\nu_{12} = \nu_{21} = \nu_{red} = 0.5(\nu_{11} + \nu_{22}) \nu_{red}$$

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

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

for some constant $\rho_{red}$ in $[-1, 1]$.

The constants $f$ and $m$ in the last equation are given as follows:

$$f = \frac{\Gamma(\nu_{11} + d/2)\Gamma(\nu_{22} + d/2)}{\Gamma(\nu_{11})\Gamma(\nu_{22})} \times \frac{(\Gamma(\nu_{12})/\Gamma(\nu_{12} + d/2))^{2} \times (c_{12}^{2} / (s_{11}^{\nu_{12}} s_{22}^{\nu_{22}}))^{2}}$$

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

$$g(t) = (1/s_{12}^{2} + t^{2})^{2\nu_{12} + d} \times (1/s_{11}^{2} + t^{2})^{-\nu_{11} - d/2} \times (1/s_{22}^{2} + t^{2})^{-\nu_{22} - d/2}$$

(cf. Gneiting, T., Kleiber, W., Schlather, M. (2010), Full Bivariate Matern Model (Section 2.2))

\section*{Value}

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

\section*{Author(s)}

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

\section*{References}


\section*{See Also}

\texttt{Rmparswm, 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

x <- y <- seq(-10, 10, 0.2)
model <-RMbr2bg(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))
```

---

**Transformation from Brown-Resnick to Bernoulli**

Description

This function can be used to model a max-stable process based on a binary field, with the same extremal correlation function as a Brown-Resnick process

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

Here, \( \Phi \) 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 `RMmodel`.

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

object of class `RMmodel`

Author(s)

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

References


See Also

`maxstableAdvanced`, `RMbr2eg`, `RMmodel`, `RMm2r`, `RPbernoulli`, `RPbrownresnick`, `RPschlather`.

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(var=1.62 / 2)
x <- seq(0, 10, 0.05)
z <- RFsimulate(RPschlather(RMbr2eg(model)), x, x)
plot(z)
```

---

**RMbr2eg**

Transformation from Brown-Resnick to Gauss

Description

This function can be used to model a max-stable process based on the a binary field, with the same extremal correlation function as a Brown-Resnick process

\[
C_{eg}(h) = 1 - 2(1 - 2\Phi(\sqrt{\gamma(h)/2}))^2
\]

Here, \(\Phi\) 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 \texttt{RMmodel}.

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

Details

\texttt{RMbr2eg}

The extremal Gaussian model \texttt{RPschlather} simulated with \texttt{RMbr2eg(RMmodel())} has tail correlation function that equals 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/de/publications/software}

References


See Also

\texttt{maxstableAdvanced, RMbr2bg, RMmodel, RMM2r, RPbernoulli, RPBrownresnick, RPschlather},

Examples

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

\texttt{model} <- \texttt{RMexp(var=1.62 / 2)}
\texttt{binary.model} <- \texttt{RPbernoulli(RMbr2bg(model))}
\texttt{x} <- \texttt{seq(0, 10, 0.05)}
\texttt{z} <- \texttt{RFsimulate(RPschlather(binary.model), x, x)}
\texttt{plot(z)}
RMBrownresnick (Tail correlation function of the Brown-Resnick process)

Description

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

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

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

Usage

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

Arguments

phi
variogram of class RMMmodel.

var, scale, Aniso, proj
optional arguments; same meaning for any RMMmodel. If not passed, the above covariance function remains unmodified.

Details

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

Value

object of class RMMmodel

Note

In the paper Kabluchko et al (2009) the variogram instead of the semi-variogram is considered, so the formulae differ slightly.

In Version 3.0.33 a typo has been corrected.

Here, a definition is used that is consistend with the rest of the package.

Author(s)

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

References


See Also

* RFsimulate, RMin2r, RMin3b, RMmps, RModel.

Examples

```r
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
RModel <- RMfbm(alpha=1.5, scale=0.2)
plot(RMBrownresnick(RModel))

#simulate and plot corresponding Gaussian random field
x <- seq(-5, 5, 0.05)
z <- RFsimulate(RMBrownresnick(RModel), x=x, y=x)
plot(z)
```

---

**RMcauchy**

*Cauchy Family Covariance Model*

**Description**

*RMcauchy* is a stationary isotropic covariance model belonging to the Cauchy family. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (1 + r^2)^{-\gamma}$$

where $\gamma > 0$. See also *Rmgencauchy*.

**Usage**

```r
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 RModel`. 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 \texttt{rmgencauchy}) includes this family for the choice \( \alpha = 2 \) and \( \beta = 2\gamma \). The generalized Hyperbolic Family (see \texttt{rmhyperbolic}) includes this family for the choice \( \xi = 0 \) and \( \gamma = -\nu/2 \); in this case \( \text{scale} = \delta \).

Value

\texttt{RMcauchy} returns an object of class \texttt{RMmodel}

Author(s)

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

References


See Also

\texttt{RMcauchytbm, RMgencauchy, 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 <- RMcauchy(gamma=1)
x <- seq(0, 10, 0.02)
plot(model, xlim=c(-3, 3))
plot(RFsimulate(model, x=x, n=4))
\end{verbatim}

\textbf{RMcauchytbm}  Modifications of the Cauchy Family Covariance Model

Description

\texttt{RMcauchytbm}() is a shortcut of \texttt{RMtbm(RMgencauchy())} and is given here for downwards compatibility.
Usage

RMcauchytbm(alpha, beta, gamma, var, scale, Aniso, proj)

Arguments

alpha, beta  see RMgencauchy.
gamma  is the same as fulldim in RMTbm.
var, scale, Aniso, proj
  optional arguments; same meaning for any Rmmodel. If not passed, the above
covariance function remains unmodified.

Value

RMcauchytbm returns an object of class Rmmodel

Author(s)

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

References

- Gneiting, T. and Schlather, M. (2004) Stochastic models which separate fractal dimension and
  Hurst effect. SIAM review 46, 269–282.

See Also

RMcauchy, RMgencauchy, Rmmodel, RFSimulate, RFfit.

Examples

RFoptions(seed=0) # set random number generator seed to 0
RFoptions(seed=NA) # for reproducability

model <- RMcauchytbm(alpha=1, beta=1, gamma=3)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFSimulate(model, x=x))
**Description**

`RMchoquet` is a 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

$$
\psi(\theta) = \sum_{n=0}^{\infty} b_{n,2}/(n + 1) \cdot P_n(\cos(\theta)),
$$

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 realvalued funçions on $[0, \pi]$, with $\psi(0) = 1$ and such that the associated isotropic function

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

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

is (strict) positive definit 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 famiy (see `RMMultiquad`) and the model of the sine power funçion (see `RMSinepower`).

**Value**

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

**Author(s)**

Christoph Berreth, <cberreth@mail.uni-mannheim.de>
**References**


**See Also**

`RMmodel`, `RFSimulate`, `RFFit`, `spherical models`, `RMmultiquad`, `RMsinepower`

**Examples**

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

#b =
#model <- RMchoquet(b=b)
#x <- seq(0, 10, 0.02)
#plot(model)
#plot(RFSimulate(model, x=x))
```

---

**RMcircular**  
**Circular Covariance Model**

**Description**

`RMcircular` is a stationary isotropic covariance model which is only valid for dimensions $d \leq 2$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

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

**Usage**

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

**Arguments**

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

  optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.
Details

The model is only valid for dimensions $d \leq 2$. It is a covariance function with compact support (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 82).

Value

RMcircular returns an object of class RMmodel

Author(s)

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

References


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))

Covariance Matrix Constant in Space

RMconstant defines a spatially constant covariance function

Usage

RMconstant(M, var)

Arguments

M a numerical matrix defining the user-defined covariance for a random field; The matrix should be positive definite, symmetric and its dimension should be equal to the length of observation or simulation vector.

var variance
RMcoord

Description

This function is used to define, in the (rare) case, coordinates that differ from the original coordinates to define a covariance matrix for a random effect model
~ RMmodel() + Z @ RMcoord(coord=X, RMmodel2())

Usage

RMcoord(C0, coord, dist)

Arguments

C0  covariance function of class RMmodel.
coord, dist  either coordinates or a the lower matrix of a distance matrix can be passed

Value

RMcoord returns an object of class RMmodel.

Author(s)

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

Model for covariates

Description

The model makes covariates available.

Usage

RMcovariate(c, x, y=NULL, z=NULL, T=NULL, grid, var, scale, Aniso, proj, raw, norm, addNA)

Arguments

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

c
  vector or matrix of data

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.

Details

The functions interpolates (nearest neighbour) between the values.
**Value**

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

**Note**

- `c`, `x`, also accept lists of data. However, its use is not in an advanced stage yet.

**Author(s)**

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

**See Also**

`RMfixcov`, `RMmodel`, `RMtrend`

**Examples**

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

z <- 0.2 + (1:10)
RFfctn(RMcovariate(z), 1:10)
RFfctn(RMcovariate(z, 1:10), c(2, 2.1, 2.5, 3))
```

---

**RMcoxisham**

*Cox Isham Covariance Model*

**Description**

`RMcoxisham` is a stationary covariance model which depends on a univariate stationary isotropic covariance model $C_0$, which is a normal scale mixture.

The corresponding covariance function only depends on the difference $(h,t) \in \mathbb{R}^{d+1} = \mathbb{R}^d \times \mathbb{R}$ between two points in $d + 1$-dimensional space and is given by

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

Here $\mu \in \mathbb{R}^d$ 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**

```r
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(monotone="normal mixture")}
and whose \texttt{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

\texttt{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. See Schlather, M. (2010), Example 9).

Value

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

Author(s)

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

References


See Also

\texttt{RMmodel, RFsimulate, RFit}.

Examples

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

model <- RMcoxisham(RMgauss(), mu = 1, D = 1)
x <- seq(0, 10, 0.3)
plot(model, dim = 2)
```
**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**

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

**Arguments**

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

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

**Details**

The model is only valid for dimensions $d \leq 3$. It is a 2 times differentiable covariance function with compact support (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 84).

**Value**

**RMcubic** returns an object of class **RMmodel**

**Author(s)**

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

**References**


**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

\[
C(h) = (-\nabla_h (\nabla_h)^T) C_0(h)
\]

**Usage**

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

**Arguments**

- `phi` a univariate stationary covariance model (2 or 3 dimensional).
- `var, scale, Aniso, proj` optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

**Details**

The model returns the potential field in the first component, the corresponding curlfree field and field of sources and sinks in the last component.

See also the models `RMdivfree` and `RMvector`.

**Value**

`RMcurlfree` returns an object of class `RMmodel`

**Author(s)**

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

References


See Also

RMdivfree, RMvector, 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 <- 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

\[
C(r) = \phi(r), 0 \leq r \leq d \\
C(r) = b_0((dR)^a - r^a)^{2a}, d \leq r \leq dR \\
C(r) = 0, dR \leq r
\]

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

Usage

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

phi

da 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 Rmmodel. If not passed, the above
covariance function remains unmodified.

Details

The algorithm that checks the given parameters knows only about some few necessary conditions.
Hence it is not ensured that the cutoff-model is a valid covariance function for any choice of $\phi$ 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 Rmmodel

Author(s)

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

References

Graph. Statist. 11, 587–599

See Also

Rmmodel, RFsimulate, RFfit.

Examples

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

model <- Rmexp()
plot(model, model.cutoff=RMcutoff(model, diameter=1), xlim=c(0, 4))
Description

`RMdagum` is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

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

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

Usage

`RMdagum(beta, gamma, var, scale, Aniso, proj)`

Arguments

- `beta`: numeric in \((0, 1]\)
- `gamma`: numeric in \((0, 1)\)
- `var, scale, Aniso, proj`: optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

Details

Like the generalized Cauchy model the Dagum family can be used to model fractal dimension and Hurst effect. For a comparison of these see Berg, C. and Mateau, J. and Porcu, E. (2008). This paper also establishes valid parameter choices for the Dagum family, but be careful because therein the model is parameterized differently.

Value

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

Author(s)

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

References

See Also

RMmodel, RFsimulate, RFFit.

Examples

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

model <- RMdagum(beta=0.5, gamma=0.5, scale=0.2)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))

---

Description

**RMdampedcos** is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

\[
C(r) = \exp(-\lambda r) \cos(r)
\]

Usage

RMdampedcos(lambda, var, scale, Aniso, proj)

Arguments

- lambda numeric. The range depends on the dimension of the random field (see details)
- var, scale, Aniso, proj optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

The model is valid for any dimension \( d \). However, depending on the dimension of the random field the following bound for the arguments \( \lambda \) 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

**RMdampedcos** returns an object of class **RMmodel**

Author(s)

Martin Schlather, <schlather@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 <- RMdampedcos(lambda=0.3, scale=0.1)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

**RMdelay**

---

**Bivariate Delay Effect**

---

**Description**

**RMdelay** is a \((m + 1)\)-variate stationary covariance model, which depends on a univariate stationary covariance model \(C_0\).

The corresponding covariance function only depends on the difference \(h \in \mathbb{R}^d\) between two points in \(d\)-dimensional space and is given by

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

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 single univariate random field, by shifting it by fixed value.

Value

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

Author(s)

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

References


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")
```
RMdewijsian

**Modified DeWijsian Variogram Model**

**Description**

The modified RMdewijsian model is an intrinsically stationary isotropic variogram model. The corresponding centered semi-variogram only depends on the distance \( r \geq 0 \) between two points and is given by

\[
\gamma(r) = \log(r^\alpha + 1)
\]

where \( \alpha \in (0, 2] \).

**Usage**

`RMdewijsian(alpha, var, scale, Aniso, proj)`

**Arguments**

- **alpha** a numerical value; in the interval \((0, 2]\).
- **var, scale, Aniso, proj**
  - optional arguments; same meaning for any `Rmodel`. If not passed, the above variogram remains unmodified.

**Details**

Originally, the logarithmic model \( \gamma(r) = \log(r) \) was named after de Wijs and reflects a principle of similarity (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 90). But note that \( \gamma(r) = \log(r) \) is not a valid variogram (\( \gamma(0) \) does not vanish) and can only be understood as a characteristic of a generalised random field.

The modified RMdewijsian model \( \gamma(r) = \log(r^\alpha + 1) \) is a valid variogram model (cf. Wackernagel, H. (2003), p. 336).

**Value**

`RMdewijsian` returns an object of class `Rmodel`

**Note**

Note that the (non-modified) de Wijsian model equals \( \gamma(r) = \log(r) \).

**Author(s)**

Martin Schlather, <schlather@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))

RMdivfree Divfree Covariance Model

Description

RMdivfree is a multivariate covariance model which depends on a univariate stationary covariance
model where the covariance function phi(h) is twice differentiable.

The corresponding matrix-valued covariance function C of the model only depends on the difference
h between two points and it is given by

\[ C(h) = (-\Delta E + \nabla \nabla^T)C_0(h) \]

Usage

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

Arguments

phi a univariate stationary covariance model (in 2 or 3 dimensions).
var, scale, Aniso, proj
optional arguments; same meaning for any RMmodel. If not passed, the above
covariance function remains unmodified.

Details

The model returns the potential field in the first component, the corresponding divfree field and the
field of curl strength in the last component.

See also the models RMcurlfree and RMvector.

Value

RMdivfree returns an object of class RMmodel
**Author(s)**
Martin Schlather, <schlather@math.uni-mannheim.de>

**References**

**See Also**
RMcurlfree, RMvector, 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 <- RMdivfree(RMgauss(), scale=4)
plot(model, dim=2)

x.seq <- y.seq <- seq(-10, 10, 0.2)
simulated <- RFsimulate(model=model, x=x.seq, y=y.seq)

plot(simulated)
plot(simulated, select.variables=1)
plot(simulated, select.variables=2:3)
plot(simulated, select.variables=list(2:3))
plot(simulated, select.variables=list(1, 2:3, 4))
plot(simulated, select.variables=list(1, c(1, 2:3), 4))
```

---

**RMeaxxa**

**Special models for rotation like fields**

**Description**

*RMeaxxa* and *RMetaxxa* define the auxiliary functions

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

and

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

, respectively.

**Usage**

*RMeaxxa*(E, A)
*RMetaxxa*(E, A, alpha)
Arguments

- \( E \): m-variate vector of positive values
- \( A \): \( m \times k \) matrix
- \( \alpha \): angle for the rotation matrix \( R \)

Details

- \( R\text{meaxxa} \) is defined in space and returns an m-variate model.
- \( R\text{metaxxa} \) 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

\( R\text{meaxxa} \) and \( R\text{metaxxa} \) return an object of class \( R\text{Mmodel} \).

Author(s)

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

References


See Also

- \( R\text{Mmodel}, S10 \)

Examples

```r
# see S10
```

Description

\( R\text{mepscauchy} \) is a stationary isotropic covariance model belonging to the generalized Cauchy family. **In contrast to most other models it is not a correlation function.** The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

\[
C(r) = (\varepsilon + r^\alpha)^{(\frac{1}{\alpha} - \frac{\beta}{\alpha})}
\]

where \( \varepsilon > 0, \alpha \in (0, 2] \) and \( \beta > 0 \). See also \( R\text{mcauchy} \).
Usage

RMepscauchy(alpha, beta, eps, var, scale, Aniso, proj)

Arguments

alpha
  a numerical value; should be in the interval (0,2] to provide a valid covariance function for a random field of any dimension.

beta
  a numerical value; should be positive to provide a valid covariance function for a random field of any dimension.

eps
  a positive value

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

Details

This model has a smoothness parameter $\alpha$ 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>

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))
```

---

**RMexp**

*Exponential Covariance Model*

**Description**

**RMexp** is a stationary isotropic covariance model whose corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

\[
C(r) = e^{-r}.
\]

**Usage**

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

**Arguments**

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

**Details**

This model is a special case of the Whittle covariance model (see **RMwhittle**) if \( \nu = \frac{1}{2} \) and of the symmetric stable family (see **RMstable**) if \( \nu = 1 \). Moreover, it is the continuous-time analog of the first order autoregressive time series covariance structure.

The exponential covariance function is a normal scale mixture.

**Value**

**RMexp** returns an object of class **RMmodel**.

**Author(s)**

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

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

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 `Rmodel`; either a variogram model or a covariance model
n   integer, see formula above. Default is -1.; if the multivariate dimension of the
     submodel is greater than 1 then only the default value is valid.
standardised  logical. If TRUE then the above formula holds. If FALSE then only
              the nominator of the above formula is returned. Default value is TRUE.
var, scale, Aniso, proj  optional arguments; same meaning for any `Rmodel`. If not passed, the
                         above covariance function remains unmodified.

Details

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

Value

`RMexponential` returns an object of class `Rmodel`

Author(s)

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

References

See, for instance,

  fields. In Porcu, E., Montero, J. M., Schlather, M. *Advances and Challenges in Space-

See Also

`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 <- RMexponential(RMfbm(alpha=1))  # identical to RExp()
plot(RMexp(), model=model, type=c("p", "l"), pch=20)
```
**Description**

RMfbm is an intrinsically stationary isotropic variogram model. The corresponding centered semi-
variogram only depends on the distance \( r \geq 0 \) between two points and is given by

\[
\gamma(r) = r^\alpha
\]

where \( \alpha \in (0, 2] \).

By now, the model is implemented for dimensions up to 3.

For a generalized model see also RMgenfbm.

**Usage**

```r
RMfbm(alpha, var, scale, Aniso, proj)
```

**Arguments**

- `alpha` numeric in \( (0, 2] \); refers to the fractal dimension of the process
- `var, scale, Aniso, proj` optional arguments; same meaning for any RMmodel. If not passed, the above variogram remains unmodified.

**Details**

The variogram is unbounded and belongs to a non-stationary process with stationary increments.

For \( \alpha = 1 \) and `scale=2` we get a variogram corresponding to a standard Brownian Motion.

For \( \alpha \in (0, 2) \) the quantity \( H = \frac{\alpha}{2} \) is called Hurst index and determines the fractal dimension \( D \) of the corresponding Gaussian sample paths

\[
D = d + 1 - H
\]

where \( d \) is the dimension of the random field (see Chiles and Delfiner, 1999, p. 89).

**Value**

RMfbm returns an object of class RMmodel.

**Author(s)**

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

**References**

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 <- RMfbm(alpha=1)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFSimulate(model, x=x))
```

---

**RMfixcov**  
Fixed Covariance Matrix

Description

**RMfixcov** is a user-defined covariance according to the given covariance matrix. It extends to the space through a Voronoi tessellation.

Usage

```R
RMfixcov(M, x, y=NULL, z=NULL, T=NULL, grid, var, scale, Aniso, proj,
        raw, norm)
```

Arguments

- `scale, Aniso, proj, var`: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.
- `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
- `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 (outside mixed models)
- `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>

References

ianola, D., Schlather, M., Mackay, T.F.C., Simianer, H. (2012): Using Whole Genome Se-
quence Data to Predict Quantitative Trait Phenotypes in Drosophila melanogaster. PLoS
Genet 8(5): e1002685.

See Also

RMcovariate, RMmodel, Rfsimulate, Rffit, RMuser

Examples

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

## Example 1 showing that the covariance structure is correctly implemented
n <- 10
z <- matrix(runif(n^2), nc=n)
(z <- z %*% t(z))
Rfcovmatrix(RMfixcov(z), 1:n)

## Example 2 showing that the covariance structure is interpolated
Rfcovmatrix(RMfixcov(z, 1:n), c(2, 2.1, 2.5, 3))

## Example 3 showing the use in a separable space-time model
model <- RMfixcov(z, 1:n, proj="space") * RMexp(s=40, proj="time")
(z <- Rfsimulate(model, x = seq(0,12, 0.5), T=1:100))
plot(z)
**RMfixed**

**Fixed Effect Model**

**Description**

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

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

that specifies the Linear Mixed Model.

Important Remark: $\text{RMfixed}$ is NOT a function although the parantheses notation is used to specify the vector of coefficients.

The matrix $X$ is the design matrix and $\beta$ 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 \text{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>

**See Also**

\text{RMmodel}, \text{RFformula}, \text{Rfsimulate},

**Examples**

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

---

**RMflatpower**

**Variogram Model Similar to Fractal Brownian Motion**

**Description**

\text{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

$$\gamma(r) = \frac{r^2}{(1 + r^2)^\alpha}$$

where $\alpha \in (0, 1]$.

For related models see \text{RMgenfbm}. 
**Usage**

\`RMflatpower(alpha, var, scale, Aniso, proj)\`

**Arguments**

- `alpha` numeric in \((0, 1]\); refers to the fractal dimension of the process
- `var, scale, Aniso, proj` optional arguments; same meaning for any \`RMmodel\`. If not passed, the above variogram remains unmodified.

**Details**

The model is always smooth at the origin.

The parameter \(\alpha\) 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>

**References**


**See Also**

\`RMgenfbm, RMmodel, RFSimulate, RFfit\`.

**Examples**

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

\`model <- RMflatpower(alpha=0.5)
\`x <- seq(0, 10, 0.1)

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

---

*Note: This text is automatically generated and may not reflect the exact original content.*
RMfracdiff 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

RMfracdiff(a, var, scale, Aniso, proj)

Arguments

- \( a \) \(-1 \leq a < 1\)
- \( \text{var}, \text{scale}, \text{Aniso}, \text{proj} \)
  - 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

RMfracdiff returns an object of class RMmodel.

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

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**

```r
RMfractgauss(alpha, var, scale, Aniso, proj)
```

**Arguments**

- `alpha` \( 0 < \alpha \leq 2 \)
- `var, scale, Aniso, proj` optional arguments; same meaning for any **RMmodel**. If not passed, the above covariance function remains unmodified.

**Details**

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

**Value**

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

**Author(s)**

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


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 <- Rmfractgauss(alpha=0.5, scale=0.2)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

### Rmgauss

**Gaussian Covariance Model**

Description

*Rmgauss* is a stationary isotropic covariance model. The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

\[
C(r) = e^{-r^2}
\]

Usage

*Rmgauss*(var, scale, Aniso, proj)

Arguments

var, scale, Aniso, proj

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

Details

This model is called Gaussian because of the functional similarity of the spectral density of a process with that covariance function to the Gaussian probability density function.

The Gaussian model has an infinitely differentiable covariance function. This smoothness is artificial. Furthermore, this often leads to singular matrices and therefore numerically instable procedures (cf. Stein, M. L. (1999), p. 29).

The Gaussian model is included in the symmetric stable class (see *Rmstable*) for the choice \( \alpha = 2 \).
Value

**RMgauss** returns an object of class **RMmodel**

Note

The use of **RMgauss** is questionable from both a theoretical (analytical paths) and a practical point of view (e.g., speed of algorithms). Instead, **RMgneiting** should be used.

Author(s)

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

References


See Also

**RMstable** and **RMmatern** for generalisations;
**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))
```

---

**RMgencauchy**

*Generalized Cauchy Family Covariance Model*

Description

**RMgencauchy** is a stationary isotropic covariance model belonging to the generalized Cauchy family. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = (1 + r^\alpha)^{\frac{\beta}{\alpha}}$$

where $\alpha \in (0, 2]$ and $\beta > 0$. See also **RMcauchy**.
Usage

\texttt{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 \texttt{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 \texttt{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 \texttt{RMcauchy}) is included in this family for the choice $\alpha = 2$ and $\beta = 2\gamma$.

Value

\texttt{RMgencauchy} returns an object of class \texttt{RMmodel}

Author(s)

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

References

Covariance function


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

See Also

RMcauchy, RMcauchytbm, RMmodel, RFsimulate, RFFit.

Examples

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

model <- RMgencauchy(alpha=1.5, beta=1.5, scale=0.3)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))

RMgenfbm

Generalized Fractal Brownian Motion Variogram Model

Description

RMgenfbm is an intrinsically stationary isotropic variogram model. The corresponding centered
semi-variogram only depends on the distance \( r \geq 0 \) between two points and is given by

\[
\gamma(r) = (r^\alpha + 1)^{\beta/\alpha} - 1
\]

where \( \alpha \in (0, 2] \) and \( \beta \in (0, 2] \).

See also RMfbm.

Usage

RMgenfbm(alpha, beta, var, scale, Aniso, proj)

Arguments

alpha a numerical value; should be in the interval (0,2].

beta a numerical value; should be in the interval (0,2].

var, scale, Aniso, proj optional arguments; same meaning for any RMmodel. If not passed, the above
variogram remains unmodified.

Details

Here the variogram of RMfbm is modified by the transformation \( (\gamma + 1)^{\delta/\alpha} - 1 \) 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.
Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>

References


See Also

RMbcw RMfbm, RMmodel, RMflatpower, RFsimulate, RFit.

Examples

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

t <- RMgenfbm(alpha=1, beta=0.5)
x <- seq(0, 10, 0.02)
plot(t)
plot(RFsimulate(t, 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 $\kappa$ 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**,

$$C(r) = (1 - r)^\beta 1_{[0,1]}(r), \quad \beta = \mu + 1 = \mu + 2\kappa + 1.$$  

For $\kappa = 1$ the Gneiting model is given by

$$C(r) = (1 + \beta r)(1 - r)^\beta 1_{[0,1]}(r), \quad \beta = \mu + 2\kappa + 1.$$  

If $\kappa = 2$

$$C(r) = \left(1 + \beta r + \frac{2\beta^2 - 1}{3}r^2\right)(1 - r)^\beta 1_{[0,1]}(r), \quad \beta = \mu + 2\kappa + 1.$$  

In the case $\kappa = 3$

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

A special case of this model is **RMgneiting**. 

Usage

```r
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 `RMgengneiting` (with \( s = 1 \) there) for the choice \( \kappa = 3, \mu = 3/2 \).

Value

`RMgengneiting` returns an object of class `RMmodel`

Author(s)

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

References


See Also

`RMaskey`, `RMBigneiting`, `RMgneiting`, `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 <- RMgengneiting(kappa=1, mu=1.5)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))

## same models:
model2 <- RMgengneiting(kappa=3, mu=1.5, scale= 1 / 0.3011874658255)
```
RMgennsst

Non-Separable Space-Time model

Description

RMgennsst is a univariate stationary spaceisotropic covariance model whose corresponding covariance is given by

\[ C(h, u) = (\psi(u) + 1)^{-\delta/2} \phi(h/\sqrt{\psi(u) + 1}) \]

Usage

RMgennsst(phi, psi, var, scale, Aniso, proj)

Arguments

phi is normal mixture RMmodel, cf.
RFgetModelNames(monotone="normal mixture")
psi is a variogram RMmodel.
var, scale, Aniso, proj
optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

This model is used for space-time modelling where the spatial component is isotropic.

Value

RMgennsst returns an object of class RMmodel.

Author(s)

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

References


See Also

RMnsst, 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
```

---

**RMgneiting**:  
*Gneiting Covariance Model*

**Description**

`RMgneiting` is a stationary isotropic covariance model which is only valid up to dimension 3, or 5 (see the argument `orig`). The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

\[
C(r) = (1 + 8sr + 25s^2r^2 + 32s^3r^3)(1 - sr)^8
\]

if \( 0 \geq r \geq \frac{1}{s} \) and

\[
C(r) = 0
\]

otherwise. Here, \( s = 0.301187465825 \). For a generalized model see also `RMgengneiting`.

**Usage**

```r
RMgneiting(Orig, var, scale, Aniso, proj)
```

**Arguments**

- `var, scale, Aniso, proj`: optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.
- `Orig`: logical. if `TRUE` the above model is used. Otherwise the `RMgengneiting` model \( C(sr) \) with `kappa` 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 \texttt{RMgengneiting} for the choice $\kappa = 3, \mu = 1.5$.

Note that, in the original work by Gneiting (1999), a numerical value slightly deviating from the optimal one was used for $\mu = 1.5$: $s = \frac{10\sqrt{2}}{47}$.

**Value**

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

**Author(s)**

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

**References**

For the original version


For the version (\texttt{orig=FALSE})

- this package \texttt{RandomFields}

**See Also**

\texttt{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(\texttt{orig=FALSE}), model3=RMgauss(),
     xlim=c(-3,3), maxchar=100)
plot(RMgneiting(), model2=RMgneiting(\texttt{orig=FALSE}), model3=RMgauss(),
     xlim=c(1.5,2.5), maxchar=100)

model <- RMgneiting(\texttt{orig=FALSE}, scale=0.4)
x <- \texttt{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)
```
**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>

**References**


**See Also**

`RMbigneiting,RMgneiting,RMgengneiting,RMgauss,RMmodel,RMwhittle,RFsimate,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 <- RMgneitingdiff(nu=2, taper.scale=1, scale=0.2)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

---

**RMhyperbolic**

*Generalized Hyperbolic Covariance Model*

**Description**

**RMhyperbolic** is a stationary isotropic covariance model called “generalized hyperbolic”. The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points and is given by

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

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

**Usage**

```r
RMhyperbolic(nu, lambda, delta, var, scale, Aniso, proj)
```

**Arguments**

- `nu, lambda, delta`
  - numerical values; should either satisfy
    - \( \delta \geq 0, \lambda > 0 \) and \( \nu > 0 \), or
    - \( \delta > 0, \lambda > 0 \) and \( \nu = 0 \), or
    - \( \delta > 0, \lambda \geq 0 \) and \( \nu < 0 \).
- `var, scale, Aniso, proj`
  - optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

**Details**

This class is over-parametrized, i.e. it can be reparametrized by replacing the three parameters \( \lambda, \delta \) 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 Cauchy covariance function (see `RMcauchy`) with \( \gamma = -\nu \) and scale=\( \delta \); the choice \( \delta = 0 \) yields a covariance model of type `RMwhittle` with smoothness parameter \( \nu \) and scale parameter \( \lambda^{-1} \).
Value

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

Author(s)

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

References


See Also

`RMcauchy`, `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

model <- RMhyperbolic(nu=1, lambda=2, delta=0.2)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

RMIaco

**Iaco-Cesare model**

Description

The space-time covariance function is

\[ C(r,t) = (1.0 + r^\nu + t^\lambda)^\delta \]

Usage

`RMIaco(nu, lambda, delta, var, scale, Aniso, proj)`
Arguments
nu, lambda  number in (0, 2]
delta  positive number
var, scale, Aniso, proj
optional arguments; same meaning for any RMmodel. If not passed, the above
covariance function remains unmodified.

Value
RMiaco returns an object of class RMmodel

Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>

References
• de Cesare, L., Myers, D.E., and Posa, D. (2002) FORPRAN programs for space-time model-
ing. Computers & Geosciences 28, 205-212.
some parameteric families. Math. Geol. 34, 23-42.

See Also
RMmodel.

Examples
RFoptions(seed=0) # ANY simulation will have the random seed 0; set
# RFoptions(seed=NA) to make them all random again
model <- RMIaco(nu=1, lambda=1.5, delta=0.5)
plot(model, dim=2)
x <- seq(0, 10, 0.1)
plot(RFsimulate(model, x=x, y=x))
Arguments

- **phi**: covariance function of class \texttt{RMmodel}.
- **vdim**: for internal purposes
- **var, scale, Aniso, proj**: optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.

Value

\texttt{RMid} returns an object of class \texttt{RMmodel}

Author(s)

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

See Also

\texttt{RMmodel},

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 <- RMexp()
x <- 0:10
z <- RFsimulate(model, x)

t <- RFoptions()
model2 <- RMid(model)
z2 <- RFsimulate(model, x)
s <- RFoptions()

sum(abs(as.vector(z) - as.vector(z2))) == 0 # TRUE
\end{verbatim}

Description

Internal models or model names that may appear in feedbacks from 'RandomFields'. Those ending by ‘Intern’ should appear only in very rare cases.

Details

The following and many more internal models exist

- \texttt{RF\_Name\_} : internal representation of certain functions \texttt{RF\_name\_}
- \texttt{RO\#} : model for transforming coordinates within the cartesian system
- \texttt{RO>\#} : model for transforming earth coordinates to cartesian coordinates
• RMmissing: for error messages only
• RMmixed: internal representation of a mixed model
• RMselect: will be obsolete in future
• RMsetparam, RMptsGivenShape, RMstandardShape, RMstatShape: for max-stable processes and Poisson processes: models that combine shape functions with corresponding point processes
• RP__name__Intern: internal representations of some processes
• RPS, RPplusp, etc.: specific processes for RMS and RMplus etc. (For those covariance models that have specific simulation processes programmed.)
• RMS: internal representation of the modifying arguments var, scale, Aniso, proj

**Author(s)**

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

**Examples**

```r
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(RFspecific(RMexp() + RMnugget()), x)
RFgetModelInfo(which="internal", level=0)
```

---

**RMintexp**  
*Integral exponential operator*

**Description**

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

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

**Usage**

```r
RMintexp(phi, var, scale, Aniso, proj)
```
Arguments

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

Value

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

Author(s)

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

References

• Schlather, M. (2012) Construction of covariance functions and unconditional simulation of

See Also

\texttt{RMmodel,RFsimulate,RFFit}.

Examples

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

model <- RMintexp(RMbmd(alpha=1.5, scale=0.2))
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))

---

\textbf{RMintrinsc}  \hspace{1cm} \textit{Intrinsic Embedding Covariance Model}

Description

\textbf{RMintrinsc} 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

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

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

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

\texttt{RMintrinsic(phi, diameter, rawR, var, scale, Aniso, proj)}

Arguments

\begin{itemize}
  \item \texttt{phi} \hspace{2cm} an \texttt{RMmodel}; has to be stationary and isotropic
  \item \texttt{diameter} \hspace{2cm} a numerical value; positive; should be the diameter of the domain on which simulation is done
  \item \texttt{rawR} \hspace{2cm} a numerical value; greater or equal to 1
  \item \texttt{var, scale, Aniso, proj} \hspace{2cm} optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.
\end{itemize}

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 \texttt{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

\texttt{RMintrinsic} returns an object of class \texttt{RMmodel}

Author(s)

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

References

\begin{itemize}
\end{itemize}

See Also

\texttt{RPintrinsic, RMmodel, RFsimulate, RFFit}.
Examples

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

x.max <- 10
model <- RM intrinsic(RMfbm(alpha=1), diameter=x.max)
x <- seq(0, x.max, 0.02)
plot(model)
plot(RFsimulate(model, x=x))

---

**RMkolmogorov**

**Identical Model**

Description

RMkolmogorov corresponds to a vector-valued random fields with covariance function

\[ \gamma_{ij}(h) = \|h\|^{2/3} \left( \frac{4}{3} \delta_{ij} - \frac{1}{3} \frac{h_i h_j}{\|h\|^2} \right) \]

Usage

RMkolmogorov(var, scale, Aniso, proj)

Arguments

var, scale, Aniso, proj

optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value

RMkolmogorov returns an object of class RMmodel.

Author(s)

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

References

The above formula is from eq. (6.32) of section 6.2 in


See Also

RMmodel, RMcurlfree, RMdivfree, RMvector.
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(-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"))
```

**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**

```r
RMLgd(alpha, beta, var, scale, Aniso, proj)
```

**Arguments**

- **alpha** argument whose range depends on the dimension of the random field: \( 0 < \alpha \leq (3 - d)/2 \).
- **beta** \( \beta > 0 \).
- **var, scale, Aniso, proj** optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

**Details**

The model is only valid for dimension \( d = 1, 2 \).

This model admits simulating random fields where fractal dimension \( D \) of the Gaussian sample and Hurst coefficient \( H \) can be chosen independently (compare also `RMscauchy`):

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>}

References


See Also

\texttt{RMmodel, RFsimulate, RFfit}.

Examples

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

model <- RMLgd(alpha=0.7, beta=4, scale=0.5)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))
```

\textbf{RMma} \hspace{1cm} \textit{Ma operator}

Description

\textbf{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

\textbf{RMma}(phi, alpha, theta, var, scale, Aniso, proj)

Arguments

- \textbf{phi} \hspace{1cm} a stationary covariance \texttt{RMmodel}.
- \textbf{alpha} \hspace{1cm} a numerical value; positive
- \textbf{theta} \hspace{1cm} a numerical value; in the interval \((0, 1)\)
- \textbf{var}, \textbf{scale}, \textbf{Aniso}, \textbf{proj} \hspace{1cm} optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.
Value

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

Author(s)

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

References


See Also

\textbf{RMmodel}, \textbf{RFSimulate}, \textbf{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 <- RMma(RMgauss(), alpha=4, theta=0.5)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFSimulate(model, x=x))
```

\textbf{RMmastein}  \hspace{1cm} \textit{Ma-Stein operator}

Description

\textbf{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


Instead of the velocity parameter \( V \) in the original model description, a preceeding anisotropy matrix is chosen appropriately:

\[
\begin{pmatrix}
A & -V \\
0 & 1
\end{pmatrix}
\]

A is a spatial transformation matrix. (I.e. \((x,t)\) is multiplied from left on the above matrix and the first elements of the obtained vector are 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, \( \nu + \phi + \delta \) may not exceed the value 80.0. If exceeded the algorithm fails.

Value

**RMmastein** returns an object of class **RMmodel**

Author(s)

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

References


See Also

**RMwhittle**, **RMmodel**, **RFSimulate**, **RFFit**.
**Examples**

```r
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))
```

---

**RMmatri**

**Matrix operator**

**Description**

**RMmatri** is a multivariate covariance model depending on a multivariate covariance model \( \phi \). The corresponding covariance function is given by

\[
C(h) = M \phi(h) M^t
\]

**Usage**

```
RMmatri(phi, M, var, scale, Aniso, proj)
```

**Arguments**

- **phi**: a \( k \)-variate covariance **RMmodel**.
- **M**: a \( k \) times \( k \) matrix
- **var, scale, Aniso, proj**: optional arguments; same meaning for any **RMmodel**. If not passed, the above covariance function remains unmodified.

**Value**

**RMmatri** returns an object of class **RMmodel**

**Author(s)**

Martin Schlather, <schlather@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
# bivariate Linear Model of Coregionalisation
model <- RMMatrix(M = c(0.9, 0.43), RMwhittle(nu = 0.3)) +
    RMMatrix(M = c(0.6, 0.8), RMwhittle(nu = 2))
x <- y <- seq(-10, 10, 0.2)
simu <- RFSimulate(model, x, y)
plot(simu)
Variogram models (stationary increments/intrinsically stationary)

- `RMfbm` fractal Brownian motion

Basic Operations

- `RMMult`, `*` product of covariance models
- `RMPlus`, `+` sum of covariance models or variograms

Basic models for mixed effect modelling

- `RMfixcov` constant pre-defined covariance
- `RMfixed` fixed or trend effects; Caution: `RMfixed` is not a function and can be used only in formula notation

Others

- `RMtrend` trend
- `RMangle` defines a 2x2 anisotropy matrix by rotation and stretch arguments.

Author(s)

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

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

References


See Also

*RM* for an overview over more advanced classes of models

*RC, RF, RP, RR, R., Rfcov, Rfformula, RMmodelsAdvanced, RMmodelsAuxiliary, trend modelling*

Examples

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

## an example of a simple model
model <- RMexp(var=1.6, scale=0.5) + RMnugget(var=0) # exponential + nugget
plot(model)
```

---

**RMmodel-class**

*Class* RMmodel

**Description**

Class for *RandomField*’s representation of explicit covariance models

**Usage**

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

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

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

- **x**: object of class RFsp or RFempVario or RFfit or RMmodel; in the latter case, x can be any sophisticated model but it must be either stationary or a variogram model.
- **y**: ignored in most methods.
- **MARGIN**: vector of two; two integer values giving the coordinate dimensions w.r.t. which the field or the covariance model is to be plotted; in all other directions, the first index is taken.
- **fixed.MARGIN**: only for class(x)="RMmodel" and if dim > 2; a vector of length dim-2 with distance values for the coordinates that are not displayed.
- **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 'RMmodel', see Details.
- **dim**: must equal 1 or 2; only for class(x)="RMmodel"; the covariance function and the variogram are plotted as a function of \(R^{\text{dim}}\).
- **n.points**: integer; only for class(x)="RMmodel"; the number of points at which the model evaluated (in each dimension); defaults to 200.
- **fct.type**: character; only for class(x)="RMmodel"; must equal NULL, "Cov" or "Variogram"; controls whether the covariance (fct.type="Cov") or the variogram (fct.type="Variogram") is plotted; NULL implies automatic choice, where "Cov" is chosen whenever the model is stationary.

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 = "RMmodel")**: allows to sum up covariance models; internally calls RMplus.
- **signature(x = "RMmodel")**: allows to substract covariance models; internally calls R.minus.
- **signature(x = "RMmodel")**: allows to multiply covariance models; internally calls R.minus.
- **signature(x = "RMmodel")**: allows to divide covariance models; internally calls R.div.
- **signature(x = "RMmodel")**: concatenates covariance functions or variogram models.
RMmodel-class

plot signature(x = "RMmodel"): gives a plot of the covariance function or of the variogram model, for more details see plot-method.

points signature(x = "RMmodel"): adds a covariance plot to an existing plot, for more details see plot-method.

lines signature(x = "RMmodel"): adds a covariance plot to an existing plot, for more details see plot-method.

str signature(x = "RMmodel"): as the usual str-method for S4 objects but where only those entries of the 'par.general'-slot are shown that contain values different from 'RFdefault'

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

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

[ signature(x = "RMmodel"): enables accessing the slots via the"["-operator, e.g. x["par.general"]

[< signature(x = "RMmodel"): enables replacing the slots via the"["-operator

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

Author(s)

Alexander Malinowski <Alexander.Malinowski@web.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
Description

Extension of Class \texttt{RMmodel} which additionally contains the likelihood of the data w.r.t. the covariance model represented by the "RMmodel" part, the estimated trend of the data if it is a constant trend, and the residuals of the data w.r.t. the model. Objects of this class only occur as slots in the output of "RFfit".

Creating Objects

Objects are only ment to be created by the function \texttt{RFfit}

Slots

\begin{itemize}
\item \texttt{aic}: the AIC value for the ml estimation
\item \texttt{aicc}: the corrected AIC value for the ml estimation
\item \texttt{bic}: the BIC value for the ml estimation
\item \texttt{call}: see \texttt{RMmodel}.
\item \texttt{likelihood}: numeric; the likelihood of the data w.r.t. the covariance model
\item \texttt{name}: see \texttt{RMmodel}.
\item \texttt{par.model}: see \texttt{RMmodel}.
\item \texttt{par.general}: see \texttt{RMmodel}.
\item \texttt{param}: vector of estimated parameters
\item \texttt{residuals}: array or of class \texttt{RFsp}; residuals of the data w.r.t. the trend model
\item \texttt{submodels}: see \texttt{RMmodel}.
\item \texttt{trend}: numeric; the estimated mean of the data (if a constant mean was specified in the model)
\item \texttt{variab}: vector of estimated variables. Variables are used in the internal representation and can be a subset of the parameters.
\end{itemize}

Extends

Class "RMmodel", directly.

Methods

\begin{itemize}
\item \texttt{[ signature(x = "RMmodelFit")}: enables accessing the slots via the "["-operator, e.g. x["likelihood"]
\item \texttt{[< signature(x = "RMmodelFit")}: enables replacing the slots via the "["-operator
\item \texttt{show signature(x = "RFfit")}: returns the structure of x
\item \texttt{print signature(x = "RFfit")}: identical with show-method
\item \texttt{anova} performs a likelihood ratio test base on a chisq approximation
\item \texttt{summary} gives a summary
\end{itemize}
Author(s)

Alexander Malinowski \textless{}Alexander.Malinowski@web.de\textgreater{}, Martin Schlather, \textless{}schlather@math.uni-mannheim.de\textgreater{}

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

See Also

\texttt{RMmodel Rffit}

Examples

\#

\texttt{see Rffit}

\begin{verbatim}
RMmodelgenerator-class

Class \texttt{RMmodelgenerator}

Description

Class for all functions of this package with prefix \texttt{RM}, i.e. all functions that generate objects of class \texttt{RMmodel}; direct extension of class \texttt{function}

Creating Objects

Objects should not be created by the user!

Slots

\texttt{.Data:} function; the genuine function that generates an objects of class \texttt{RMmodel}
\texttt{type:} character string; specifies the category of \texttt{RMmodel-function}, see Details
\texttt{domain:} character string; specifies whether the corresponding function(s) depend on 1 or 2 variables, see Details
\texttt{isotropy:} character string; specifies the type of isotropy of the corresponding covariance model, see Details
\texttt{operator:} logical; specifies whether the underlying covariance model is an operator, see Details
\texttt{monotone:} character string; specifies the kind of monotonicity of the model
\texttt{finterange:} logical; specifies whether the underlying covariance model has finite range, see Details
\texttt{simpleArguments:} logical. If \texttt{TRUE} than all the parameters are real valued (or integer valued).
\texttt{maxdim:} numeric; the maximal dimension, in which the corresponding model is a valid covariance model, see Details
\texttt{vdim:} numeric; dimension of the value of the random fields at a single fixed location, equals 1 in most cases, see Details
\end{verbatim}
Extends

Class \texttt{function}, directly.

Methods

\texttt{show} signature(\(x = \text{"RMmodel"}\)): returns the structure of \(x\)

\texttt{print} signature(\(x = \text{"RMmodel"}\)): identical with \texttt{show}-method

\texttt{[} signature(\(x = \text{"RMmodelgenerator"}\)): enables accessing the slots via the \texttt{[}-operator, e.g. \(x[\text{"maxdim"]}\)

\texttt{[<-} signature(\(x = \text{"RMmodelgenerator"}\)): enables replacing the slots via the \texttt{[}-operator

Details

type: can be one of the following strings:

\begin{itemize}
  \item \texttt{tail correlation function'}: indicates that the function returns a tail correlation function (a subclass of the set of positive definite functions)
  \item \texttt{positive definite'}: indicates that the function returns a covariance function (positive definite function)
  \item \texttt{negative definite'}: indicates that the function returns a variogram model (negative definite function)
  \item \texttt{process'}: functions of that type determine the class of processes to be simulated
  \item \texttt{method for Gauss processes'}: methods to simulate Gaussian random fields
  \item \texttt{method for Brown-Resnick processes'}: methods to simulate Brown-Resnick fields
  \item \texttt{point-shape function'}: functions of that type determine the distribution of points in space
  \item \texttt{distribution family'}: e.g. (multivariate) uniform distribution, normal distribution, etc., defined in \texttt{RandomFields}. See \texttt{RR} for a complete list.
  \item \texttt{shape function'}: functions used in, e.g., M3 processes (\texttt{RPsmith})
  \item \texttt{trend'}: \texttt{RMtrend} or a \texttt{mixed model}
  \item \texttt{interface'}: indicates internal models which are usually not visible for the users. These functions are the internal representations of \texttt{RFsimulate}, \texttt{RFcov}, etc.. See \texttt{RF} for a complete list.
  \item \texttt{undefined'}: some models can take different types, depending on the parameter values and/or the submodels
  \item \texttt{other type'}: very very special internal functions, not belonging to any of the above types.
\end{itemize}

domain: can be one of the following strings:

\begin{itemize}
  \item \texttt{single variable'}: Function depending on a single variable
  \item \texttt{kernel'}: model refers to a kernel, e.g., an non-stationary covariance function
  \item \texttt{framework dependent'}: domain depends on the calling model
  \item \texttt{mismatch'}: this option is used only internally and should never appear
\end{itemize}

isotropy: can be one of the following strings:

\begin{itemize}
  \item \texttt{isotropic'}: indicates that the model is isotropic
  \item \texttt{space-isotropic'}: indicates that the spatial part of a spatio-temporal model is isotropic
'zero-space-isotropic': this property refers to space-time models; the model is called zerospaceisotropic if it is isotropic as soon as the time-component is zero
'vector-isotropic': multivariate vector model (flow fields) have a different notion of isotropy
'symmetric': the most basic property of any covariance function or variogram model
'cartesian system', 'earth system', 'spherical system', 'cylinder system': different coordinate systems
(non-dimension-reducing): the property $f(x) = f(-x)^T$ does not hold
'parameter dependent': indicates that the type of isotropy of the model depends on the parameters passed to the model; in particular parameters may be submodels if an operator model is considered
'
'mismatch': this option is used only internally and should never appear
operator: if TRUE, the model requires at least one submodel
monotone: 'mismatch in monotonicity': used if a statement on the monotonicity does not make sense, e.g. for 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 not known
'monotone': isotone or antitone
'Gneiting-Schaback class': function belonging to Euclid’s hat in Gneiting’s 1999 paper
'normal mixture': scale mixture of the Gaussian model
'completely monotone': completely monotone function
'Bernstein': Bernstein function
Note that
• 'not monotone' includes 'monotone' and 'Bernstein'
• 'monotone' includes 'Gneiting-Schaback class'
• 'Gneiting-Schaback class' includes 'normal mixture'
• 'normal mixture' includes 'completely monotone'
finitrange: if TRUE, the covariance of the model has finite range
maxdim: if a positive integer, maxdim gives the maximum dimension in which the model is a valid covariance model, can be Inf; vdim=-1 means that the actual maxdim depends on the parameters; vdim=-2 means that the actual maxdim depends on the submodel(s)
vdim: if a positive integer, vdim gives the dimension of the random field, i.e. univariate, bi-variate, ...; vdim=-1 means that the actual vdim depends on the parameters; vdim=-2 means that the actual vdim depends on the submodel(s)

Author(s)

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

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

Author(s)

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

See Also

RC, RR RF, R.

Examples

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

RFgetModelNames(type="positive definite", domain="single variable",
    isotropy="isotropic", operator=!FALSE) ## Rmodel.Rd

RMmodelsAdvanced

Advanced features of the models

Description

Here, further models and advanced comments for RMmodel are given. See also RFgetModelNames.

Details

Further stationary and isotropic models

RMaskey               Askey model (generalized test or triangle model)
RMbessel              Bessel family
RMcircular            circular model
RMconstant            spatially constant model
RMcubic               cubic model (see Chiles \& Delfiner)
RMDagum               Dagum model
RMDampedcos           exponentially damped cosine
RMqexp                Variant of the exponential model
RMfracdiff            fractionally differenced process
RMfracgauss           fractional Gaussian noise
RMGengneiting         generalized Gneiting model
RMGneitingdiff        Gneiting model for tapering
RMhyperbolic          generalised hyperbolic model
RM1gd                 Gneiting’s local-global distinguisher
RMma                  one of Ma’s model
RMpenta               penta model (see Chiles \& Delfiner)
RMpower               Golubov’s model
RMwave                cardinal sine

Variogram models (stationary increments/intrinsically stationary)
**RMdewijsian**  generalised version of the DeWijssian model

**RMgenfbm**  generalized fractal Brownian motion

**RMflatpower**  similar to fractal Brownian motion but always smooth at the origin

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

<table>
<thead>
<tr>
<th>Operator</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>RMexponential</td>
<td>exponential of a covariance model</td>
</tr>
<tr>
<td>RMintexp</td>
<td>integrated exponential of a covariance model (INCLUDES ma2)</td>
</tr>
<tr>
<td>RMp</td>
<td>powered variograms</td>
</tr>
<tr>
<td>RMqam</td>
<td>Porcu’s quasi-arithmetic-mean model</td>
</tr>
<tr>
<td>RMS</td>
<td>details on the optional transformation arguments (var, scale, Aniso, proj).</td>
</tr>
</tbody>
</table>

**Stationary and isotropic composed models (operators)**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMcutoff</td>
<td>Gneiting’s modification towards finite range</td>
</tr>
<tr>
<td>RMintrinsic</td>
<td>Stein’s modification towards finite range</td>
</tr>
<tr>
<td>RMnatsc</td>
<td>practical range</td>
</tr>
<tr>
<td>RMstein</td>
<td>Stein’s modification towards finite range</td>
</tr>
<tr>
<td>RMTbm</td>
<td>Turning bands operator</td>
</tr>
</tbody>
</table>

**Stationary space-time models**
See **RMmodelsSpaceTime**

**Non-stationary models**
See **RMmodelsNonstationary**

**Negative definite models that are not variograms**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMsum</td>
<td>a non-stationary variogram model</td>
</tr>
</tbody>
</table>

**Models related to max-stable random fields (tail correlation functions)**
See **RMmodelsTailCorrelation.**

**Other covariance models**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMuser</td>
<td>User defined model</td>
</tr>
<tr>
<td>RMfixcov</td>
<td>User defined covariance structure</td>
</tr>
</tbody>
</table>

**Trend models**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aniso</td>
<td>for space transformation (not really trend, but similar)</td>
</tr>
<tr>
<td>RMcovariate</td>
<td>spatial covariates</td>
</tr>
<tr>
<td>RMprod</td>
<td>to model variability of the variance</td>
</tr>
</tbody>
</table>
**RMmodelsAdvanced**

**RMpolynome** easy modelling of polynomial trends
**RMTrend** for explicite 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 \textsc{null} the argument is not set (but must have a valid name).
- \textsc{aniso} can be given also by \texttt{RManglE} or any other \texttt{RMmodel} instead by a matrix
- Note also that a completely different possibility exists to define a model, namely by a list. This format allows for easy flexible models and modifications (and some few more options, as well as some abbreviations to the model names, see \texttt{PrintModellist()}). Here, the argument \texttt{var}, \texttt{scale}, \texttt{aniso} and \texttt{proj} must be passed by the model \texttt{RMS}. For instance,

  ```r
  model <- RMexp(scale=2, var=5)
  is equivalent to
  model <- list("RMS", scale=2, var=5, list("RMexp"))
  The latter definition can be also obtained by
  print(RMexp(scale=2, var=5))
  ```

  ```r
  model <- RMnsst(phi=RMgauss(var=7), psi=RMfbm(alpha=1.5), scale=2, var=5)
  is equivalent to
  model <- list("RMS", scale=2, var=5,
  list("RMnsst", phi=list("RMS", var=7, list("RMgauss")),
  psi=list("RMfbm", alpha=1.5)))
  ```

  All models have secondary names that stem from \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, <malinowski@math.uni-mannheim.de>

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

---

**References**


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
## of the differentiabilities
x <- seq(-0.6, 0.6, len=50)
model <- RMwhittleCnu=0.8 + 1.5 * R.isCp(new="isotropic"), "<=", 0.5)C
z <- RFsimulateCmodel=model, x, x, n=4
plot(z)
```

Description

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

Details

**Covariance models**

- `RMbiwm`  full bivariate Whittle-Matern model (stationary and isotropic)
- `RMBigneiting`  bivariate Gneiting model (stationary and isotropic)
- `RMCurlfree`  curlfree (spatial) vector-valued field (stationary and anisotropic)
- `RMDelay`  bivariate delay effect model (stationary)
- `RMDivfree`  divergence free (spatial) vector valued field, (stationary and anisotropic)
- `RMExponential`  functional returning $e^C$
- `RMKolmogorov`  Kolmogorov’s model of turbulence
### Trend models

- **RMtrend**: for explicite trend modelling
- **R.models**: for implicite trend modelling
- **R.c**: binding univariate trend models into a vector

### Author(s)

Martin Schlather, <schlather@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)

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

## generation of artificial data
```
Here, nonstationary covariance models are presented.

Details

Covariance models

RMnonstwm
RMprod
Aniso

scale, cf. RMS, can be any non-negative function for any scale mixture model, such as the whittle-matern-classes, the powered exponential family, and the

Trend models see RMmodelsTrend

Author(s)

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

See Also

RFFormula, RMmodels, RM, RMmodelsAdvanced

‘nonstationary’, a vignette for nonstationary 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 take also 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
- **RMcoxivisham** 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
- **Rmiaco** non-separable space-time model
- **RMmastein** Ma-Stein model
- **RMsnsst** Gneiting’s non-separable space-time model
- **RMstein** Stein’s non-separable space-time model
- **RMsst** Single temporal process
- **RMtbm** Turning bands operator

## Author(s)

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

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

## References

See Also

\texttt{Rfformula, RM, RMmodels, RMmodelsAdvanced}

Examples

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

## separable model with exponential model in space and gaussian in time
model <- RMexp(proj = "space") * RMgauss(proj = "time")
xt <- seq(0, 10, 0.1)
z <- RFsimulate(model, x=xt, T=xt)
plot(z)
\end{verbatim}

\textit{RMmpplus} \hspace{1cm} \textit{Mixture of shape functions}

Description

\textit{RMmpplus} is a multivariate covariance model which depends on up to 10 submodels $C_0, C_1, \ldots, C_9$. It is used to gather with \texttt{RPsmit} it allowed for mixed moving maxima with a finite number of shape functions.

Usage

\begin{verbatim}
RMmpplus(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, p)
\end{verbatim}

Arguments

- \texttt{C0} \hspace{1cm} an \texttt{RMmodel}.
- \texttt{C1, C2, C3, C4, C5, C6, C7, C8, C9} \hspace{1cm} optional; each an \texttt{RMmodel}.
- \texttt{p} \hspace{1cm} 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

\textit{RMmpplus} returns an object of class \texttt{RMmodel}

Author(s)

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

See Also

\texttt{RMplus, RMmodel, RFsimulate, RFit, RPsmit}.
Examples

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 RMmodel
theta is a vector of values in $[0, 1]$, summing up to 1.
var, scale, Aniso, proj optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Note that $\psi(\cdot) := \phi(\sqrt{\cdot})$ is completely monotone if and only if $\phi$ is a valid covariance function for all dimensions, e.g. RMstable, RMgauss, RMexpontential.

Warning: RandomForest cannot check whether the combination of $\phi$ and $C_i$ is valid.
Value

`rmqam` returns an object of class `rmmodel`.

Author(s)

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

References


See Also

`rmqam`, `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
```

---

**RMmult**

*Multiplication of Random Field Models*

Description

`RMmult` is a multivariate covariance model which depends on up to 10 submodels $C_0, C_1, ..., C_10$. In general, realizations of the created `rmmodel` are pointwise product of independent realizations of the submodels.

In particular, if all submodels are given through a covariance function, the resulting model is defined through its covariance function, which is the product of the submodels' covariances.

Usage

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

Arguments

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

**RMmodel**s can also be multiplied via the \(*\)-operator, e.g.: \(C0 \ast C1\)

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

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

equals

\[
\text{RMexp}(\text{Aniso}=A2 \%\% A1) \ast \text{RMspheric}(\text{Aniso}=A3 \%\% A1)
\]

In case that all submodels are given through a covariance function, the global argument var of **RMmult** is multiplied to the product covariance of **RMmult**.

Value

**RMmult** returns an object of class **RMmodel**

Author(s)

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

See Also

**RMplus**, **RMmodel**, **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)
```

**RMmultiquad**

*The Multiquadric Family Covariance Model on the Sphere*

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

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

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

RMmultiquad(delta, tau, var, scale, Aniso, proj)

Arguments

delta a numerical value in (0, 1)
tau a numerical value greater than 0
var, scale, Aniso, proj optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Details

Special cases (cf. Gneiting, T. (2013), p.1333) are known for fixed parameter $\tau = 0.5$ which leads to the covariance function called 'inverse multiquadric'

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

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

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

For a more general form, see RMchoquet.

Value

RMmultiquad returns an object of class RMmodel

Author(s)

Christoph Berreth, <cberreth@mail.uni-mannheim.de>, Martin Schlather

References


See Also

RMmodel, RFsimulate, RFfit, RMchoquet, spherical models

Examples

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

RFoptions(coord_system="sphere")
model <- RMmultiquad(delta=0.5, tau=1)
plot(model, dim=2)
RMnatsc

### the following two pictures are the same

```r
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")
```

---

<table>
<thead>
<tr>
<th>RMnatsc</th>
<th>Natural scale</th>
</tr>
</thead>
</table>

**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

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)

Description

The non-stationary Whittle-Matern model $C$ is given by

$$C(x, y) = \Gamma(\mu)\Gamma(\nu(x))^{-1/2}\Gamma(\nu(y))^{-1/2}W_\mu(|x - y|)$$

where $\mu = [\nu(x) + \nu(y)]/2$, and $\nu$ must a positive function. $W_\mu$ is the covariance of the RMwhittle model or the RMmatern model and

Details

The non-stationary Whittle-Matern models are obtained by the respective stationary model, replacing the real-valued argument for $nu$ by a non-negative function.

Note

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

Author(s)

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

References

**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**

\[
\text{RMnsst}(\text{phi, psi, delta, var, scale, Aniso, proj})
\]

**Arguments**

- **phi** is normal mixture \texttt{RMmodel}, cf. \texttt{RFgetModelNames(monotone="normal mixture")}
- **psi** is a variogram \texttt{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 \texttt{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**

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

**Examples**

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

x & \leftarrow \text{seq(-1.2, 1.2, len=50)} \\
\text{model} & \leftarrow \text{RMwhittle(mu=RMgauss())}
\end{align*}
\]

\[
\begin{align*}
z & \leftarrow \text{RFsimulate(model=model, x, x, n=4)} \\
\text{plot}(z)
\end{align*}
\]
Author(s)

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

References


See Also

`RMgennsst`, `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 <- RMnugget(tol=1, vdim=5, var=1, scale=1, aniso=1, proj=1)
x <- seq(0, 6, 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 $1, ..., vdim$ 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, scale, Aniso, proj)`
Arguments

tol Only for advanced users. See \texttt{RPNugget}.
vdim Only for advanced users. See \texttt{RPNugget}.
var, scale, Aniso, proj
optional arguments; same meaning for any \texttt{RMmodel}. If not passed, the above covariance function remains unmodified.

Details

Note that the argument scale does not affect the covariance model; Aniso has an effect in case of zonal anisotropy.

The nugget effect belongs to Gaussian white noise and is often used for modeling measurement errors.

The locations at a distance less than or equal to \texttt{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 \texttt{nugget.tol}=0.0. However, if the anisotropy matrix does not have full rank and \texttt{nugget.tol}=0.0 then, the simulations are likely to be odd. The value of \texttt{nugget.tol} should be of order $10^{-15}$.

Value

\texttt{RMnugget} returns an object of class \texttt{RMmodel}

Author(s)

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

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 <- RMnugget(Aniso=matrix(1, nr=2, nc=2))
x <- seq(0, 10, 0.02)
plot(RFsimulate(model, x=x, y=x, tol=1e-10))
\end{verbatim}
Description

\texttt{RMrparswm} 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 \texttt{RMwhittle} model.

\texttt{RMrparswmX} is defined as

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

where \( \rho_{ij} \) is any covariance matrix.

Usage

\texttt{RMrparswm(nudiag, var, scale, Aniso, proj)}

\texttt{RMrparswmX(nudiag, rho, var, scale, Aniso, proj)}

Arguments

- \texttt{nudiag} a vector of arbitrary length of positive values; each entry positive; the vector \((\nu_{11}, \nu_{22}, \ldots)\). The offdiagonal elements \( \nu_{ij} \) are calculated as \(0.5(\nu_{ii} + \nu_{jj})\).
- \texttt{rho} any positive definite \( m \times m \) matrix; here \( m \) equals \texttt{length(nudiag)} For the calculation of \( c_{ij} \) see Details.
- \texttt{var, scale, Aniso, proj} optional arguments; same meaning for any \texttt{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 \texttt{RMrparswmX} is \texttt{RMschur(M=rho, RMrparswm(nudiag, var, scale, Aniso, proj))}.

Value

\texttt{RMrparswm} returns an object of class \texttt{RMmodel}.
**RMPenta**

**Author(s)**

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

**References**


**See Also**

`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 <- RMParswmX(nudiag=c(1.3, 0.7, 2), rho=rho)
x.seq <- y.seq <- seq(-10, 10, 0.1)
z <- RFsimulate(model = model, x=x.seq, y=y.seq)
plot(z)
```

---

**RMPenta**  
*Penta Covariance Model*

**Description**

`RMPenta` is a stationary isotropic covariance model, which is valid only for dimensions $d \leq 3$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

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

**Usage**

`RMPenta(var, scale, Aniso, proj)`

**Arguments**

`var, scale, Aniso, proj`  
optional arguments; same meaning for any `RModel`. If not passed, the above covariance function remains unmodified.
Details

The model is only valid for dimension \( d \leq 3 \).

It has a 4 times differentiable covariance function with compact support (cf. Chiles, J.-P. and

Value

\texttt{RMpenta} returns an object of class \texttt{RMmodel}

Author(s)

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

References

  Wiley.

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 <- RMpenta()
x <- seq(0, 10, 0.02)
plot(model)
plot(RFSimulate(model, x=x))
\end{verbatim}

Description

\texttt{RMplus} is an additive covariance model which depends on up to 10 submodels \( C_0, C_1, \ldots, C_{10} \). In

general, realizations of the created \texttt{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

\begin{verbatim}
RMplus(C0, C1, C2, C3, C4, C5, C6, C7, C8, C9, var, scale, Aniso, proj)
\end{verbatim}
Arguments

- `C0` a `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

`RMmodel`s can also be summed up via the `+`-operator, e.g.: `C0 + C1`

The global arguments `var`, `scale`, `Aniso`, `proj` of `RMplus` are multiplied to the corresponding arguments of the submodels (from the right side).

Value

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

Author(s)

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

See Also

`RMmult`, `RMmodel`, `RMsum`, `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 <- RMplus(RMgauss(), RMnugget(var=0.1))
model2<- RMgauss() + RMnugget(var=0.1)
plot(model, "+model1", model2, type=c("p", "l"), pch=20, xlim=c(0,3)) # the same
```

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 hyperplan 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/de/publications/software

References

Poisson polygons / Poisson hyperplane tessellation


Poisson storm process


Mixed Poisson storm process


See Also

RMball, RMSpheric, RFsimulate, RMmodel.

Examples

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

<table>
<thead>
<tr>
<th>RMpolynome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating polynomial models</td>
</tr>
</tbody>
</table>

Description

Polynomial, mainly used in trend models, can be created easily with this function.

Usage

RMpolynome(degree, dim, value=NA, varnames = c("x", "y", "z", "T"), proj=1:4)
Arguments

- **degree**: degree of the polynome
- **dim**: number of variables in the polynome
- **value**: values of the coefficients. See Details
- **varnames**: the names of the variables
- **proj**: the projection to certain dimensions.

Details

If the length of `value` is smaller than the number of mononomes, the remaining terms are filled with NAs. If the length is larger, the vector is cut.

Value

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

Author(s)

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

See Also

`RMtrend`, `RFit`.

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)
```

**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>

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

This model defines

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

References


See Also

**RMsum, RMmodel, RMMult**
Examples

RFoptions(seed=0) #* ANY* simulation will have the random seed 0; set
# RFoptions(seed=NA) to make them all random again
RFcov(RMprod(RMexp()), as.matrix(1:10), as.matrix(1:10), grid=FALSE)

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 **RMmodel** that is a normal scale mixture. See, for instance, RFgetModelNames(monotone="normal mixture")
- **C1, C2, C3, C4, C5, C6, C7, C8, C9**: optional further univariate stationary **RMmodel**.
- **theta**: a vector with positive entries
- **var, scale, Aniso, proj**: optional arguments; same meaning for any **RMmodel**. If not passed, the above covariance function remains unmodified.

Details

Note that \(\psi(\cdot) := \phi(\sqrt{\cdot})\) is completely monotone if and only if \(\phi\) is a valid covariance function for all dimensions, e.g. **RMstable, RMgauss, RMeponential**.

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

Value

**RMqam** returns an object of class **RMmodel**.
RMqexp

Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>

References

See Also
RMmqam, RMmodel, RFsimulate, RFfit.

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

model <- RMqam(phi=RMgauss(), RMexp(), RMgauss(),
    theta=c(0.3, 0.7), scale=0.5)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))

RMqexp

Variant of the exponential model

Description
The covariance function is

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

Usage
RMqexp(alpha, var, scale, Aniso, proj)

Arguments
alpha value in [0, 1]
var,scale,Aniso,proj optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

Value
RMqexp returns an object of class RMmodel
Author(s)
Martin Schlather, <schlather@math.uni-mannheim.de>

References
• ?

See Also
RMmodel

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

model <- RMqexp(alpha=0.95, scale=0.2)
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x))

---

RMrational  Rational function

Description
Defines a simple rational function.

\[ f(h) = \frac{a_1 + a_2 z(h)}{1 + z(h)} \]

where

\[ z(h) = h^\top A A^\top h \]

Usage
RMrational(A, a)

Arguments
A  a \( d \times d \) matrix
a  a vector of one or two components; the second component has default value zero.

Value
RMrational returns an object of class RMmodel


**Author(s)**

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

**See Also**

RMmodel, S10

**Examples**

# see S10

---

**RMrotat**

*Rotation matrices*

**Description**

RMrotat and RMrotation are auxiliary space-time functions that create some rotation

\[
 f(h,t) = s(\cos(\phi t)h_1 + \sin(\phi t)h_2) / ||h||
\]

and

\[
 f(h,t) = (\cos(\phi t)h_1 + \sin(\phi t)h_2, -\sin(\phi t)h_1 + \cos(\phi t)h_2, t)
\]

respectively

**Usage**

RMrotat(speed, phi)
RMrotation(phi)

**Arguments**

- **speed**: real value
- **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>
**See Also**

RMmodel, S10

**Examples**

```r
# see S10
```

---

**RMS**  
*Scaling operator*

**Description**

**RMS** is an operator that modifies the variance and the coordinates or distances of a submodel $\phi$ by

$$C(h) = v \star \phi(A \ast h/s).$$

Most users will never call **RMS** directly, see the details.

**Usage**

```r
RMS(phi, var, scale, Aniso, proj, anisoT)
```

**Arguments**

- **phi**: submodel
- **var**: is the optional variance parameter $v$, It can be also an arbitrary non-negative function.
- **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 $x$).
- **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 $\text{phi}(..., \text{var}, \text{scale}, \text{anisoT}, \text{Aniso}, \text{proj})$, where $\text{phi}$ 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.

Author(s)
Martin Schlather, schlather@math.uni-mannheim.de

See Also
RMmodel, Rmprod

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

RMschlather

Covariance Model for binary field based on Gaussian field

Description
RMschlather gives the tail correlation function of the extremal Gaussian process, i.e.

\[ C(h) = 1 - \sqrt{1 - \phi(h)/\phi(0)})/2 \]

where \( \phi \) 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 \texttt{RPschlather}.

Value

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

Author(s)

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

See Also

\texttt{RPschlather RMmodel,RFsimulate}.

Examples

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

## This examples considers an extremal Gaussian random field
## with Gneiting's correlation function.
## first consider the covriance model and its corresponding tail
## correlation function
model <- Rmgneiting()
plot(model, model.tail.corr.fct=RMschlather(model), xlim=c(0, 5))

## the extremal Gaussian field with the above underlying
## correlation function that has the above tail correlation function tcf
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 corrrlation
## function equal to the above tail correlation function.
z <- RFsimulate(RMschlather(model), x)
plot(z)
\end{verbatim}
**Description**

The covariance function is

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

where ‘\( \ast \)’ denotes the Schur product, i.e. elementwise multiplication

**Usage**

```r
RMschur(phi, M, diag, rhored, var, scale, Aniso, proj)
```

**Arguments**

- `phi` covariance function of class `RMmodel`.
- `M` constant \( n \times n \) covariance matrix of the same size as multivariate model `phi`.
- `diag`, `rhored` alternative way of passing `M`: `diag` is a vector of variances, `rhored` is a vector containing the correlations of lower triangle of the `M`.
- `var`, `scale`, `Aniso`, `proj` optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

**Value**

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

**Author(s)**

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

**References**

- ?

**See Also**

`RMmodel`, `RMmatrix`, `RMmatrix`
Examples

```r
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))
```

RMsing  
Random sign

Description

RMsing defines a random sign. It can be used as part of the model definition of a Poisson field.

Usage

```r
RMsing(phi, p)
```

Arguments

- `phi` shape function of class `RMmodel`.
- `p` probability of keeping the sign

Details

RMsing changes the sign of the shape function phi with probability 1-p and keeps it otherwise.

Value

RMsing returns an object of class `RMmodel`

Note

Random univariate or multivariate objects are usually start with `RR` not with `RM`. This is an exception here, as it operates on shape functions.

Author(s)

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

See Also

`RMmodel RR`. 
Examples

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

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

Description

\texttt{RM\textit{sinepower}} is a 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

\texttt{RM\textit{sinepower}(alpha, var, scale, Aniso, proj)}

Arguments

\begin{itemize}
  \item \texttt{alpha} a numerical value in $(0, 2]$
  \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}

Details

For the sine power function of Soubeyrand, Enjalbert and Sache, see Gneiting, T. (2013) equation (17). For a more general form see \texttt{RMchoquet}.

Value

\texttt{RM\textit{sinepower}} returns an object of class \texttt{RMmodel}

Author(s)

Christoph Berreth, <cberreth@mail.uni-mannheim.de>, Martin Schlather
References


See Also

`RMmodel`, `RFsimulate`, `RFFit`, `spherical models`, `RMchoquet`

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 <- RMsinepower(alpha=1.7)
plot(model, dim=2)

## the following two pictures are the same
x <- seq(0, 0.4, 0.01)
z1 <- RFsimulate(model, x=x, y=x)
plot(z1)

x2 <- x * 180 / pi
z2 <- RFsimulate(model, x=x2, y=x2, coord_system="earth")
plot(z2)

stopifnot(all.equal(as.array(z1), as.array(z2)))
RFoptions(coord_system="auto")
```

---

**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

`RMspheric(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

This covariance model is valid only for dimensions less than or equal to 3.
The covariance function has a finite range.

Value

\texttt{RMsphe}ric returns an object of class \texttt{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>

References


See Also

\texttt{RMmodel, RFsimulate, RFFit}, linkspherical models

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 <- RMsphe()  # (without var, scale, Aniso, proj these defaults are
x <- seq(0, 10, 0.02)  # used, see help(RMmodel)
plot(model)
plot(RFsimulate(model, x=x))
\end{verbatim}
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

\texttt{RMstable(alpha, var, scale, Aniso, proj)}

\texttt{RMpoweredexp(alpha, var, scale, Aniso, proj)}

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}) $\alpha = 1$ 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, <schlather@math.uni-mannheim.de>
**RMstein**

**Stein nonseparable space-time model**

**Description**

RMstein is a univariate stationary covariance model whose corresponding covariance function only depends on the difference $h$ between two points and is given by

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

Here $W_{\nu}$ is the covariance of the RMcwhittle 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 RMmodel. If not passed, the above covariance function remains unmodified.

Details


Value

RMstein returns an object of class RMmodel

Author(s)

Martin Schlather, <schlather@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 <- RMstein(nu=1.5, z=0.9)
x <- seq(0, 10, 0.05)
plot(RFsimulate(model, x=x, y=x))
**Description**

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

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

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

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

$$m = h^t Mh$$

$$h = x - y$$

**Usage**

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

**Arguments**

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

**Details**


**Value**

`RMstp` returns an object of class `RMmodel`

**Author(s)**

Martin Schlather, <schlather@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=RMexxa(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))
```

---

### Description

`RMsum` is given by

\[
C(x, y) = \phi(x) + \phi(y)
\]

It is a negative definite function although not a variogram.

### Usage

```r
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/de/publications/software

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 stationary isotropic covariance model in dimension `reduceddim` which depends on a univariate 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 `reduceddim=1` and `fulldim=3`. If only one of the arguments are given, then the difference of the two arguments equals 2.

For \( d = n + 2 \), where \( n = \) `reduceddim` and \( d = \) `fulldim` the original dimension, we have
\[ C(r) = \phi(r) + r\phi'(r)/n \]

which, for \( n=1 \) reduced to the standard TBM operator

\[ C(r) = \frac{d}{dr} r\phi(r) \]

For \( d == 2 \) \&\& \( n == 1 \) we have

\[ C(r) = \frac{d}{dr} \int_0^r \frac{u\phi(u)}{\sqrt{r^2 - u^2}} du \]

‘Turning layers’ is a generalization of the turning bands method, see Schlather (2011).

**Value**

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

**Author(s)**

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

**References**

Turning bands


Turning layers


**See Also**

\texttt{RPtbn}, \texttt{RFsimulate}. 
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 <- RMspheric()
plot(model, model.on.the.line=RMtbm(RMspheric()), xlim=c(-1.5, 1.5))

z <- RFsimulate(RPtbm(model), x, x)
plot(z)
```

**RMtrafo**

Transformation of coordinate systems

**Description**

The functions transform a coordinate system into another. Currently, essentially only from the earth system to cartesian.

RMtrafo is the internal basic function that also allows to reduce vectors to their norm.

**Usage**

```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_CARTESIANCOORD`, `RC_GNOMONICPROJ`, `RCORTHOGRAPHICPROJ`, `RC_ISONAMES`. Or the corresponding `RC_ISONAMES`. Note that RMtrafo only allows for integer values. Default: `RC_CARTESIANCOORD`

- `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 zenit must be given explicitly for projection onto a plane. See the examples below.

Author(s)

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

References

For calculating the earth coordinates as ellipsoid:

- en.wikipedia.org/wiki/Geographic Coordinate System
- nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

See Also

linkconstants, RMangle

Examples

data(weather)
(coord <- weather[1:5, 3:4])

(z <- RFFctn(RMtrafo(new=RC_CARTESIANCOORD), coord))
(zl <- RFearth2cartesian(coord)) ## equals z
zl - z ## 0, i.e., zl 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))
RMtrend

\texttt{RFearth2cartesian(coord, system="gnomonic")}
\texttt{RFearth2cartesian(coord, system="orthographic")}

\begin{tabular}{l l}
\textbf{RMtrend} & \textit{Trend Model} \\
\end{tabular}

\textbf{Description}

\texttt{RMtrend} is a pure trend model with covariance 0.

\textbf{Usage}

\texttt{RMtrend(mean)}

\textbf{Arguments}

\begin{itemize}
\item \texttt{mean} numeric or \texttt{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 \texttt{mean} is interpreted as constant mean of \(Z_i(\cdot)\).
\end{itemize}

\textbf{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 \texttt{Rfformula}.

\textbf{Value}

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

\textbf{Note}

Using uncapsulated substraction to build up a covariance function is ambiguous, see the examples below. Best to define the trend separately, or to use \texttt{R.minus}.

\textbf{Author(s)}

Marco Oesting, \texttt{oesting@math.uni-mannheim.de}

Martin Schlather, \texttt{schlather@math.uni-mannheim.de}  \texttt{http://ms.math.uni-mannheim.de/de/publications/software}
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
model <- RMexp(var=2, scale=1) + trend
data <- RFSimulate(model, 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=data))

## 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=data))

## IMPORTANT: substraction is not a way to combine definite models
## with trends

## (model0 <- RMexp(var=0.4) + trend) ## exponential covariance with mean -1
## (model1 <- RMexp(var=0.4) - 1) ## same as model0
## (model2 <- RMexp(var=0.4) + RMtrend(-1)) ## same as model0
## (model3 <- RMexp(var=0.4) - 1) ## this is a purely deterministic model
##    ## with exponential trend

plot(RFSimulate(model0, x=x, y=x)) ## exponential covariance
##    ## and mean -1
plot(RFSimulate(model1, x=x, y=x)) ## dito
plot(RFSimulate(model2, x=x, y=x)) ## dito
plot(RFSimulate(model3, x=x, y=x)) ## purely deterministic model!
Description

It may be used to truncate the support of a shape function when Poisson fields or M3 processes are created.

Usage

RMtruncsupport(phi, radius)

Arguments

phi function of class RMmodel.
radius truncation at radius

Value

RMtruncsupport returns an object of class RMmodel.

Author(s)

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

References


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
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
RMuser(type, domain, isotropy, vdim, beta,
       varnames = 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.

- **varnames**
  Just the names of the variables. More variable names might be given here than used in the function. See Detail for the interpretation of variables.

- **fctn, fst, snd**
  a user defined function and its first, second and third derivative, given as `quote(myfunction(x))` or as `quote(myfunction(x, y))`, see Details 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 \texttt{varnames}.

A kernel should depend on the first two arguments given by \texttt{varnames}.

Value

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

Note

- The use of \texttt{RMuser} is completely on the risk of the user. There is no way to check whether the expressions of the user are correct in any sense.
- Note that \( x, y, z \) and \( T \) are reserved argument names that define solely the coordinates. Hence, none of these names might be used for other arguments within these functions.
- In the user-defined functions the models of \texttt{RandomFields} are not recognised, so they cannot be included in the function definitions.
- \texttt{RMuser} may not be used in connection with obsolete commands of \texttt{RandomFields}.

Author(s)

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

See Also

\texttt{RMcovariate}, \texttt{RMfixcov}, \texttt{RFFit}, \texttt{RMmodelgenerator} \texttt{RMmodel}, \texttt{RFsimulate}, \texttt{RCISONAMES}, \texttt{RC_DOMAIN_NAMES}

Examples

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

## Alternatively to 'model <- RMexp()' one may define the following
## (which is, however, much slower and cannot use all features of
## RandomFields)

## user-defined exponential covariance model
model <- RMuser(type="positive definite", domain="single variable",
iso="isotropic", fctn=exp(-x))
x <- y <- seq(1, 10, len=100)
plot(model)
z <- RFsimulate(model, x=x, y=y)
plot(z)

## the kernel, which is the scalar product (not programmed (yet) in
\end{verbatim}
## RandomFields %to do
model <- RMnugget(var=1e-5) +
    RMuser(type="positive definite", domain="kernel",
    iso="symmetric", fctn=sum(x * y))
x <- y <- seq(1, 10, len=35)
z <- RFsimulate(model, x=x, y=y, n=6, svdtol=1e-9)
plot(z)

## Vector Covariance Model

### Description

**RMvector** is a multivariate covariance model which depends on a univariate covariance model that is stationary in the first \( D_{space} \) 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

\[
C(h, t) = \left( -0.5 \ast (a + 1) \Delta + a \nabla \nabla^T \right) C_0(h, t)
\]

where the operator is applied to the first component \( h \) only.

### Usage

```r
RMvector(phi, a, Dspace, var, scale, Aniso, proj)
```

### Arguments

- **phi**: an RMmodel; has two components \( h \) (2 or 3 dimensional and stationary) and \( t \) (arbitrary dimension)
- **a**: a numerical value; should be in the interval \([-1, 1]\).
- **Dspace**: an integer; either 2 or 3; the first \( D_{space} \) coordinates give the first component \( h \)
- **var**, **scale**, **Aniso**, **proj**: optional arguments; same meaning for any RMmodel. If not passed, the above covariance function remains unmodified.

### Details

\( C_0 \) is either a spatiotemporal model (then \( t \) is the time component) or it is an isotropic model. Then, the first \( D_{space} \) coordinates are considered as \( h \) coordinates and the remaining ones as \( t \) coordinates. By default, \( D_{space} \) equals the dimension of the field (and there is no \( t \) component). If \( a = -1 \) then the field is curl free; if \( a = 1 \) then the field is divergence free.

### Value

**RMvector** returns an object of class RMmodel.
**Author(s)**

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

**References**


**See Also**

`RMcurlfree`, `RMdivfree`, `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 <- RMvector(RMgauss(), scale=0.3)
x <- seq(0, 10, 0.4)
plot(RFsimulate(model, x=x, y=x, z=0), select.variables=list(1:2))
```

---

**RMwave**

**Wave Covariance Model / Cardinal Sine**

**Description**

`RMwave` is a stationary isotropic covariance model, which is valid only for dimension $d \leq 3$. The corresponding covariance function only depends on the distance $r \geq 0$ between two points and is given by

$$C(r) = \frac{\sin(r)}{r} \begin{cases} \quad 1_{r > 0} + 1_{r=0}. 
\end{cases}$$

It is a special case of `RMBessel`.

**Usage**

```r
RMwave(var, scale, Aniso, proj)
RMcardinalsine(var, scale, Aniso, proj)
```

**Arguments**

`var, scale, Aniso, proj`

optional arguments; same meaning for any `RMmodel`. If not passed, the above covariance function remains unmodified.

**Details**

The model is valid only for dimension $d \leq 3$. It is a special case of `RMBessel` for $\nu = 0.5$. This covariance models a hole effect (cf. Chiles, J.-P. and Delfiner, P. (1999), p. 92).
Value

`rmwave` returns an object of class `rmmodel`.

Author(s)

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

References


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))
```

---

**RMwhittlematern**

**Whittle-Matern Covariance Model**

**Description**

`rmMatern` is a stationary isotropic covariance model belonging to the Matern family. The corresponding covariance function only depends on the distance \( r \geq 0 \) between two points.

The Whittle model is given by

\[
C(r) = W_\nu(r) = 2^{1-\nu}\Gamma(\nu)^{-1}r^\nu K_\nu(r)
\]

where \( \nu > 0 \) and \( K_\nu \) is the modified Bessel function of second kind.

The Matern model is given by

\[
C(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}(\sqrt{2\nu r})^\nu K_\nu(\sqrt{2\nu r})
\]

**Usage**

`RMwhittle(nu, notinvnu, var, scale, Aniso, proj)`

`RMmatern(nu, notinvnu, var, scale, Aniso, proj)`
Arguments

nu
   a numerical value called "smoothness parameter"; should be greater than 0.
notinvnu
   logical, if not given the model is defined as above. (default).
See the Notes.
var, scale, Aniso, proj
   optional arguments; same meaning for any RModel. If not passed, the above
covariance function remains unmodified.

Details

RMwhittle and RMmatern are two alternative parametrizations of the same covariance function.
The Matern model should be preferred as this model separates the effects of scaling parameter and
the shape parameter.

This 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 appears as limit of the Matern model.
For generalisations see section ‘seealso’.

Value

The function return an object of class RModel.

Note

The Matern model called by \(C(r \sqrt{2})\) equals the Handcock-Wallis (1994) parametrisation.
The model allows further to be reparameterized by substituting \(\nu\) for \(\nu^{-1}\) setting the argument
notinvnu=TRUE. Note that the inversion of \(\nu\) does not really make sense for the Whittle model. Due
to this fact, if the argument notinvnu is given, the Whittle model changes its definition and becomes
identical to the Matern model.

The Whittle-Matern model is a normal scale mixture.

Author(s)

Martin Schlather, <schlather@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 \texttt{RMmodel}. Either a model for a process or a covariance model must be specified. In the latter case, a Gaussian process \texttt{RPGauss} is tacitely assumed.

stationary_only optional arguments; same meaning as for \texttt{RPGauss}. It is ignored if the submodel is a process definition.

threshold real valued. \texttt{RPbernoulli} returns 1 if value of the random field given by \texttt{phi} is equal to or larger than the value of \texttt{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. \texttt{threshold} defaults value is 0.

Details

\texttt{RPbernoulli} can applied to any field. If only a covariance model is given, a Gaussian field is simulated as underlying field.

Value

The function returns an object of class \texttt{RMmodel}.

Author(s)

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

See Also

Auxiliary \texttt{RMmodels}, \texttt{RP}, \texttt{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)
```

---

**RPchi2**

*Simulation of a Chi2 Random Fields*

**Description**

`RPchi2` defines a chi2 fields.

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

**See Also**

Auxiliary `RMmodels,RP,RPgauss`
Examples

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

model <- RPchi2(RMexp(), f=2)
x <- seq(0, 10, 0.1)
z <- RFsimulate(model=model, x, x, n=4)
plot(z)

Description

This function is used to specify a Gaussian random field that is to be simulated or estimated. Returns
an object of class \texttt{RMmodel}.

Usage

\texttt{RPgauss(phi, boxcox, stationary\_only)}

Arguments

\begin{itemize}
  \item \texttt{phi} the \texttt{RMmodel}.
  \item \texttt{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.
  \item \texttt{stationary\_only} Logical or NA. Used for the automatic choice of methods.
    \begin{itemize}
      \item \texttt{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.
      \item \texttt{FALSE}: intrinsic embedding is always allowed, actually it’s the first one considered in
        the automatic selection algorithm.
      \item \texttt{NA}: the simulation of the Brownian motion allowed, but intrinsic embedding is not used for
        translation invariant (“stationary”) covariance models.
    \end{itemize}
    Default: NA
\end{itemize}

Value

The function returns an object of class \texttt{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.

By default, all Gaussian random fields have zero mean. Simulating with trend can be done by including Rmtrend in the model.

Rpgauss allows to simulate different classes of random fields, controlled by the wrapping model:

If the submodel is a pure covariance or variogram model, i.e. of class RMmodel, a corresponding centered Gaussian field is simulated. Not only stationary fields but also non-stationary and anisotropic models can be used, e.g. zonal anisotropy, geometrical anisotropy, separable models, non-separable space-time smodels, multiplicative or nested models; see RMmodel for a list of all available models.

Author(s)

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

See Also

RP, Gaussian, RMmodel, RFoptions, Rfbrownresnick, RPchi2, RPopitz, RPt, RPschlather.

Do not mix up with RMgauss or RRgauss.

Examples

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

model <- RMexp()
x <- seq(0, 10, 0.02)
plot(model)
plot(RFsimulate(model, x=x, seed=0))
plot(RFsimulate(Rpgauss(model), x=x, seed=0), col=2) ## the same

RPoisson Simulation of Random Fields

Description

Shot noise model, which is also called moving average model, trigger process, dilution random field, and by several other names.
RPprocess

Models for classes of random fields (RP commands)

Description

Here, all the classes of random fields are described that can be simulated.
Implemented processes

- Gaussian Random fields see Gaussian
- Max-stable Random Fields see Maxstable
- Other Random Fields
  - Binary field
  - chi2 field
  - composed Poisson (shot noise, random coin)
  - t field

Author(s)

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

See Also

RC, RR, RM, RF, 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
```

---

**RPt**

*Simulation of a T Random Fields*

Description

**RPt** defines a t fields.
**Usage**

```r
RPt(phi, boxcox, nu)
```

**Arguments**

- `phi`: the `RMmodel`. If a model for the distribution is not specified, `RPgauss` is used as default and a covariance model is expected.
- `boxcox`: the one or two parameters of the box cox transformation. If not given, the globally defined parameters are used. see `RFboxcox` for Details.
- `nu`: non-negative number. Degree of freedom.

**Value**

The function returns an object of class `RMmodel`.

**Author(s)**

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

**References**

Related to the extremal t process


**See Also**

Auxiliary `RMmodels`, `RP`, `RPgauss`

**Examples**

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

model <- RPt(RMexp(), nu=2)
x <- seq(0, 10, 0.1)
z <- RFsimulate(model, x, x, n=4)
plot(z)
```
**RRdeterm**  
*Random scaling used with balls*

**Description**

`RRdeterm` refers to the distribution of a deterministic variable.

**Usage**

```r
call: RRdeterm(mean)
```

**Arguments**

- `mean`  
  the deterministic value

**Value**

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

**Author(s)**

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

**See Also**

`RMmodel`, `RRdistr`, `RRgauss`

**Examples**

```r
Rfoptions(seed=0)  # ANY* simulation will have the random seed 0; set
# Rfoptions(seed=NA) to make them all random again
x <- seq(-2, 2, 0.001)
p <- Rfpdistr(RRdeterm(mean=1), q=x)
plot(x, p, type="l")
```

---

**RRdistr**  
*RRdistr*

**Description**

`RRdistr` defines of distribution family given by `fct`. It is used to introduce **Random parameters** based on distributions defined on R.

**Usage**

```r
RRdistr(fct, nrow, ncol, envir)
```
Arguments

- `fct`: an arbitrary family of distribution. E.g. `norm()` for the family `dnorm`, `pnorm`, `qnorm`, `rnorm`.
- `nrow`, `ncol`: The matrix size (or vector if `ncol`=1) the family returns. Except for very advanced modelling we always have `nrow=ncol`=1, which is the default.
- `envir`: an environment; defaults to `new.env()`.

Details

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

Note

`RRdistr` is the generic model introduced automatically when distribution families in R are used in the model definition. See the examples below.

Note

See Bayesian Modelling for some 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 of obsolete commands of RandomFields.

Author(s)

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

See Also

`RMmodel`, `RR`, `RFSimulate`, `RFdistr`

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
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())
```
## rrRgauss

Random scaling used with balls

### Description

rrRgauss defines the d-dimensional vector of independent Gaussian random variables.

### Usage

```r
rrRgauss(mu, sd, log)
```

### Arguments

- `mu`, `sd`, `log`  
  See `Normal`. Here the components can be vectors, leading to multivariate distribution with independent components.

### Details

It has the same effect as `RRdistr(norm(mu=mu, sd=sd, log=log))`

### Value

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

### Author(s)

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

RRmodel, RRdistr, RRrunif.

Do not mix up RRgauss with RMgauss or Rpgauss.

Examples

RFoptions(seed=0) #*ANY* simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
r <- RRdistr(RRgauss(mu=c(1,5)), n=1000, dim=2)
plot(r[1,], r[2,])

RRloc

Random scaling used with balls

Description

RRloc modifies location and scale of a distribution.

Usage

RRloc(phi, mu, scale, pow)

Arguments

phi    distribution RRmodel.
mu     location shift
scale   scale modification
pow     argument for internal use only

Details

It has the same effect as RRdistr(norm(mu=mu, sd=sd, log=log))

Value

RRloc returns an object of class RRmodel

Author(s)

Martin Schlather, <schlather@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(RFrdistr(model, n=1000), 50)

## empirical density of the distribution 'RRspheric', shifted by 3
model <- RRloc(mu=3, RRspheric(balldim=2))
hist(RFrdistr(model, n=1000), 50)
```

**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-normalized value given by `phi` is returned. Default: FALSE
- `maxdensity` positive real number. The given density is truncated at the 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

**Details**

`RRmcmc` returns an object of class `RMmodel`.
Note
The use of RRMcmc is completely on the risk of the user. There is no way to check whether the integral of the modulus is finite.

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

See Also
RRmodel, RR, RRdistr, RRuser

Examples
RfOptions(seed=0)  ## ANY* simulation will have the random seed 0; set
##               RfOptions(seed=NA) to make them all random again
## here a model with random scale parameter

## not exponential, but the Laplace distribution as symmetry is assumed
z <- Rfrdist(RRMcmc(RMexp(), sigma=1), n=10000)
hist(z, 100, freq=FALSE)
curve(0.5 * exp(-abs(x)), add=TRUE, col="blue")  ## Laplace distribution

RRrectangular

Description
Approximates an isotropic decreasing density function by a density function that is isotropic with respect to the $l_1$ norm.

Usage
RRrectangular(phi, safety, minsteplen, maxsteps, parts, maxit,
innermin, outermax, mcmc_n, normed, approx, onesided)

Arguments
phi  a shape function; it is the user’s responsibility that it is non-negative. See details.
safety, minsteplen, maxsteps, parts, maxit, innermin, outermax, mcmc_n
   Technical arguments to run an algorithm to simulate from this distribution. See RfOptions for the default values.
normed  logical. If FALSE then the norming constant $c$ in the Details is set to 1. This affects the values the density function, the probability distribution and the quantile function, but not the simulation of random variables.
approx logical. Default TRUE. If TRUE the isotropic distribution with respect to the \( l_1 \) norm is returned. If FALSE then the exact isotropic distribution with respect to the \( l_2 \) norm is simulated. Neither the density function, nor the probability distribution, not the quantile function will be available if approx=TRUE.

onesided logical. Only for used for univariate distributions. if TRUE then the density is assumed to be non-negative only on the positive real axis. Otherwise the density is assumed to be symmetric.

Details

This models defines an isotropic density function \( f \) with respect to the \( l_1 \) norm. i.e. \( f(x) = c \phi(\|x\|_{l_1}) \) with some function \( \phi \). Here, \( s \) is norming constant so that the integral of \( f \) equals one.

In case \( \phi \) is monotonously decreasing then rejection sampling is used, else MCMC.

The function \( \phi \) might have a polynomial pole at the origin and asymptotically decreasing of the form \( x^\beta \exp(-x^\delta) \).

Value

RRRectangular returns an object of class RMmodel

Author(s)

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

See Also

RMmodel, RRdistr, RRgauss,

Examples

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

# simulation of Gaussian variables (in a not very straight forward way):
distr <- RRRectangular(RMgauss(), approx=FALSE)
z <- 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(RMspheric(), approx=FALSE)
z <- RRdistr(distr, n=1000000)
hist(z, 200, freq=TRUE)
x <- seq(-10, 10, 0.01)
lines(x, 4/3 * RFcov(RMspheric(), x))
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

```r
RRspheric(spacedim, balldim, R)
```

Arguments

- `spacedim` dimension of the hyperplane; defaults to 1.
- `balldim` the dimension of the ball
- `R` radius. Default: 1

Value

`RRspheric` returns an object of class `RMmodel`

Author(s)

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

See Also

- `RMmodel`, `RMbhall`

Examples

```r
RFOptions(seed=0) ## *ANY* simulation will have the random seed 0; set
## RFOptions(seed=NA) to make them all random again
hist(RFrdistr(RRspheric(balldim=2), n=1000), 50)
```
**Description**

The model refers to the d-dimensional univariate distribution on a rectangular window.

**Usage**

```r
RRunif(min, max, normed)
```

**Arguments**

- `min, max`: lower and upper corner of a rectangular window
- `normed`: logical with default value `TRUE`. Advanced. If `FALSE` then the indicator function for the window is not normed to get a probability distribution. Nonetheless random drawing from the distribution still works.

**Details**

In the one-dimensional case it has the same effect as `RRdistr(unif(min=min, max=max, log=log))`

**Value**

`RRunif` returns an object of class `RMmodel`

**Author(s)**

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

**See Also**

`RMmodel`, `RRdistr`, `RRgauss`, `RRspheric`

**Examples**

```r
RFoptions(seed=0)  ## *ANY* simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
## uniform distribution on [0,1] x [-2, -1]
RFrdistr(RRunif(c(0, -2), c(2, -1)), n=5, dim=2)
RFpdistr(RRunif(c(0, -2), c(2, -1)), q=c(1, -1.5), dim=2)
RFddistr(RRunif(c(0, -2), c(2, -1)), x=c(1, -1.5), dim=2)
```
Models for stationary max-stable random fields

Description
Here the code of the paper on 'Models for stationary max-stable random fields' is given.

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

References

Examples

```r
RFoptions(seed=0, xi=1)
## seed = 0 : *ANY* simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
## xi = 0.5: Frechet margins with alpha=2

## Due to change in the handling the seeds here are different from the
## seed in the paper.

x <- seq(0, 10, length=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/de/publications/software;

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)
T <- c(0, 0.02, 1275)
col <- c(topo.colors(300)[1:100], cm.colors(300)[c((1:50) * 2, 101:150)])

model <- RMcoxisham(mu=c(1, 1), D=matrix(nr=2, c(1, 0.5, 0.5, 1)),
                    RMwhittle(nu=1))
Z <- RFsimulate(model, x, y, T=T, sp_lines=1500, every=10)
plot(Z, MARGIN=slices=3, col=col)
plot(Z, MARGIN.movie=3) # add 'file="ci.avi"' to get it stored
```

---

*SBS14*  
*Systematic co-occurrence of tail correlation functions among max-stable processes*
Description

Here the code of the paper on ‘On some covariance models based on normal scale mixtures’ is given.

Author(s)

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

References


Examples
(Mixed) Moving Maxima

Description

*rpsmith* defines a moving maximum process or a mixed moving maximum process with finite number of shape functions.

Usage

*rpsmith*(shape, tcf, xi, mu, s)

Arguments

- **shape**: an *RModel* giving the spectral function
- **tcf**: an *RModel* 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 *Rm2r* construction of deterministic, monotone functions.
- **xi, mu, s**: the extreme value index, the location parameter and the scale parameter, respectively, of the generalized extreme value distribution. See Details.

Details

The argument *xi* is always a number, i.e. $\xi$ is constant in space. In contrast, $\mu$ and $s$ might be constant numerical value or given a *RModel*, in particular by a *Rm2r* model. The default values of *mu* and *s* are 1 and $\xi$, respectively.

It simulates max-stable processes $Z$ that are referred to as “Smith model”.

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

where $(W_i, X_i)$ are the points of a Poisson point process on $\mathbb{R}^d \times (0, \infty)$ with intensity $dw \ast c/x^2 dx$ and $Y_i \sim Y$ are iid measurable random functions with $E[\int \max(0, Y(x)) dx] < \infty$. The constant $c$ is chosen such that $Z$ has standard Frechet margins.

Note

IMPORTANT: for consistency reasons with the geostatistical definitions in this package the scale argument differs froms the original definition of the Smith model! See the example below.

*rpsmith* depends on *RRectangular* and its arguments.

Advanced options are **maxpoints** and **max_gauss**, see *RFoptions*. 
soil

Author(s)

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

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
x <- seq(0, 10, 0.1)
z <- RFsimulate(RPsmith(model, xi=0), x, x)
plot(z)

## original Smith model
x <- seq(0, 10, 0.05)
model <- RMgauss(scale = sqrt(2)) # !! cf. definition of RMgauss
z <- RFsimulate(RPsmith(model, xi=0), x, x)
plot(z)

## for some more sophisticated models see 'maxstableAdvanced'

---

soil

Soil data of North Bavaria, Germany

Description

Soil physical and chemical data collected on a field in the Weissenstaedter Becken, Germany
Usage

data(soil)

Format

This data frame contains the following columns:

- **x.coord**: x coordinates given in cm
- **y.coord**: y coordinates given in cm
- **nr**: number of the samples, which were taken in this order
- **moisture**: moisture content [Kg/Kg * 100%]
- **NO3.N**: nitrate nitrogen [mg/Kg]
- **Total.N**: total nitrogen [mg/Kg]
- **NH4.N**: ammonium nitrogen [mg/Kg]
- **DOC**: dissolved organic carbon [mg/Kg]
- **N20N**: nitrous oxide [mg/Kg dried substance]

Details

For technical reasons some of the data were obtained as differences of two measurements (which are not available anymore). Therefore, some of the data have negative values.

Source

The data were collected by Wolfgang Falk, Soil Physics Group, University of Bayreuth, Germany.

References


Examples

```r
RFoptions(seed=0) ## *ANY* simulation will have the random seed 0; set
## RFoptions(seed=NA) to make them all random again
```
```r
data(soil)
str(soil)
soil <- RFspatialPointsDataFrame(
  coords = soil[, c(\texttt{x.coord}, \texttt{y.coord})],
  RFparams=list(vdim=6, n=1)
)
data <- soil[\texttt{moisture}]

## plot the data first
colour <- rainbow(100)
plot(data, col=colour)

## fit by eye
gui.model <- RFgui(data)

## fit by ML
model <- -1 + RMwhittle(scale=NA, var=NA, nu=NA) + RMnugget(var=NA)
(fit <- RFfit(model, data=data))
plot(fit, method=c(\texttt{\textit{ml}}, \texttt{plain}, \texttt{sqrt.nr}, \texttt{sd.inv}),
  model = gui.model, col=1:8)

## Kriging ...
x <- seq(min(data@coords[, 1]), max(data@coords[, 1]), l=121)
k <- RFinterpolate(fit, x=x, y=x, data=data)
plot(x=k, col=colour)
plot(x=k, y=data, col=colour)

## what is the probability that at no point of the
## grid given by x and y the moisture is greater than 24 percent?
cs <- RFsimulate(model=fit, x=x, y=x, data=data, n=50)
plot(cs, col=colour)
plot(cs, y=data, col=colour)
print(mean(apply(as.array(cs) <= 24, 3, all))) ## about 40 percent ...
```

---

### Description

The function transforms an `sp` object to an `RFsp` object.

This explicite transformation is only necessary if several variables are repeated measurements are
given.
Usage

sp2RF(sp, param=list(n=1, vdim=1))

Arguments

sp an ‘sp’ object
param n number of repetitions; vdim the number of variables (multivariability)

Value

sp2RF returns an object of class RFsp.

Note

The two options varnames and coordnames, cf. Section ‘coords’ in RFoptions, might be useful.

Author(s)

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

See Also

RFsp

Examples

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

p <- 100
n <- 5
x <- runif(p, 0, 1)
y <- runif(p, 0, 1)
z <- RFsimulate(RMexp(), x=x, y=y, n=n)
z1 <- z2 <- as.data.frame(z)
coordinates(z2) <- ~coords.x1 + coords.x2

emp.var <- RFempiricalvariogram(data=z)
emp.var1 <- RFempiricalvariogram(data=z1)
emp.var2 <- RFempiricalvariogram(data=sp2RF(z2, param=list(n=n, vdim=1)))

## results are all equal, except for the calls:
emp.var@call <- emp.var1@call <- emp.var2@call <- NULL
dimnames(emp.var@emp.vario) <- dimnames(emp.var1@emp.vario) <-
dimnames(emp.var2@emp.vario) <- NULL
stopifnot(all.equal(emp.var, emp.var1))
stopifnot(all.equal(emp.var, emp.var2))
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

\texttt{RPspecific(phi, boxcox)}

Arguments

- \texttt{phi} object of class \texttt{RMmodel}; specifies the covariance model to be simulated.
- \texttt{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.

Details

\texttt{RPspecific} is used for specific algorithms or specific features for simulating certain covariance functions

- \texttt{RMplus} is able to simulate separately the fields given by its summands. This is necessary, e.g., when a \texttt{RMtrend} is involved.
- \texttt{RMMult} for Gaussian random fields only. \texttt{RMMult} simulates the random fields of all the components and multiplies them. This is repeated several times and averaged.
- \texttt{RMS} Then, for instance, sqrt(var) is multiplied onto the (Gaussian) random fields after the field has been simulated. Hence, when var is random, then, for each realisation of the Gaussian field (for n>1 in \texttt{RFSimulate}) a new realisation 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. \texttt{RPspecific(RMS())} is called internally when the user wants to simulate Anisotropic fields with isotropic methods, e.g. \texttt{RPtmb}.

- \texttt{RMppplus}
- \texttt{RMtrend}

Note that \texttt{RPspecific} applies only to the first model or operator in argument phi.

Value

\texttt{RPspecific} returns an object of class \texttt{RMmodel}

Author(s)

Martin Schlather, <schlather@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.
## In particular, it can be seen that RPspecific is applied to RMplus.
RFgetModelInfo(model, 0, which="internal")

## example for explicit use
model <- RPspecific(RMS(var=unif(min=0, max=10), RMgauss()))
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, n=6, seed=i), ylim=c(-5,5))
}
```

---

Spectral turning bands method

Description

The spectral turning bands method is a simulation method for stationary Gaussian random fields (Mantoglou and Wilson, 1982). It makes use of Bochner’s theorem and the corresponding spectral measure $\Xi$ for a given covariance function $C(h)$. For $x \in \mathbb{R}^d$, the field

$$Y(x) = \sqrt{2}\cos(\langle V, x \rangle + 2\pi U)$$

with $V^\ast \Xi$ and $U^\ast Uf_0((0,1))$ 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.
Spectral

Usage

RPspectral(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 kπ/sp_lines for k in 1:sp_lines, otherwise. This argument is only considered if the spectral measure, not the density is used.

Default: TRUE.

prop_factor positive real value. Sometimes, the spectral density must be samples by MCMC. Let p the average rejection rate. Then the chain is sampled every nth point where n = |log(p)|*prop_factor

Default: 50.

sigma real. Considered if the Metropolis algorithm is used. It gives the standard deviation of the multivariate normal distribution of the proposing distribution. If sigma is not positive thenRandomFields tries to find a good choice for sigma itself.

Default: 0.

Value

RPspectral returns an object of class RMmodel

Author(s)

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

References


See Also

Gaussian, RP, RPtbm.
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

model <- RPspectral(RMmatern(nu=1))
y <- x <- seq(0, 10, len=400)
z <- RFsimulate(model, x, y, n=2)
plot(z)
```

Spherical models  Covariance models valid on a sphere

Description

This page summarizes the covariance models that can be used for spherical coordinates (and earth coordinates).

Details

The following models are available

** Completely monotone function allowing for arbitrary scale**

- **RMbcw**: Model bridging stationary and intrinsically stationary processes for $\alpha \leq 1$ and $\beta < 0$
- **RMcubic**: cubic model
- **RMDagum**: Dagum model with $\beta < \gamma$ and $\gamma \leq 1$
- **RMexp**: exponential model
- **RMgencauchy**: generalized Cauchy family with $\alpha \leq 1$ (and arbitrary $\beta > 0$)
- **RMmatern**: Whittle-Matern model with $\nu \leq 1/2$
- **RMsph**: symmetric stable family or powered exponential model with $\alpha \leq 1$
- **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**

- **RMconstant**: spatially constant model
- **RMnugget**: nugget effect model

**Compactly supported covariance functions allowing for scales up $\pi$ (or 180 degree)**

- **RMaskey**: Askey’s model
- **RMcircular**: circular model
- **RMgengneiting**: Wendland-Gneiting model; differentiable models with compact support
- **RMsph**: differentiable model with compact support
- **RMspheric**: spherical model
Spherical models

Anisotropic models

none up to now.

Basic Operators

\[ \text{RMMult.} \ast \quad \text{product of covariance models} \]
\[ \text{RMplus.} + \quad \text{sum of covariance models or variograms} \]

See RMmodels for cartesian models.

Author(s)

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

See Also

cartesian systems, RMmodels, RMtrafo

Examples

\[
\text{RFoptions(seed=0)} \quad \text{## *ANY* simulation will have the random seed 0; set}
\quad \text{RFoptions(seed=NA) to make them all random again}
\]
\[
\text{RFgetModelNames(isotropy=c("spherical isotropic"))}
\]
\[
\text{## an example of a simple model valid on a sphere}
\text{model <- RMexp(var=1.6, scale=0.5) + RMnugget(var=0) \# exponential + nugget}
\text{plot(model)}
\]
\[
\text{## a simple simulation}
\text{l <- seq(0, 85, 1.2)}
\text{coord <- cbind(lon=l, lat=l)}
\]
\[
\text{z <- RFsimulate(RMwhittle(s=30, nu=0.45), coord, grid=TRUE) \# takes 1 min}
\text{plot(z)}
\]
\[
\text{z <- RFsimulate(RMwhittle(s=500, nu=0.5), coord, grid=TRUE,}
\quad \text{new_coord_sys="orthographic", zenith=c(25, 25))}
\text{plot(z)}
\]
\[
\text{z <- RFsimulate(RMwhittle(s=500, nu=0.5), coord, grid=TRUE,}
\quad \text{new_coord_sys="gnomonic", zenith=c(25, 25))}
\text{plot(z)}
\]
## Square roots

Methods relying on square roots of the covariance matrix

### Usage

```r
RPdirect(phi, boxcox, max_variab)
RPsequential(phi, boxcox, max_variables, back_steps, initial)
```

### 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.
- **max_variab**: integer less than 30000. If the number of variables to generate is greater than `max_variab`, neither SVD nor Cholesky decomposition are performed. If the given covariance structure has finite range and `use_spam` = `FALSE`, then `spam` is tried.
  
  It is important that this option is set conveniently to avoid great losses of time during the internal search of an appropriate method by `RPgauss`.
  
  Default: `8192`.
- **max_variables**: The maximum size of the conditional covariance matrix (default to 5000).
- **back_steps**: Number of previous instances on which the algorithm should condition. If less than one then the number of previous instances equals `max` / (number of spatial points).
  
  Default: `10`.

---

```r
## space-time modelling on the sphere
sigma <- 5 * sqrt((R.lat()-30)^2 + (R.lon()-20)^2)
model <- RMprod(sigma) * RMtrafo(RMexp(s=500, proj="space"), "cartesian") * RMspheric(proj="time")

z <- RFSimulate(model, 0:10, 10:20, T=seq(0, 1, 0.1),
            coord_system="earth", new_coordunits="km")
plot(z, MARGIN=0, slices=3)
```
**Square roots**

First, \( N = \text{(number of spatial points)} \times \text{back\_steps} \) number of points are simulated. Then, sequentially, all spatial points for the next time instance are simulated at once, based on the previous \text{back\_steps} instances. The distribution of the first \( N \) points is the correct distribution, but differs, in general, from the distribution of the sequentially simulated variables. We prefer here to have the same distribution all over (although only approximatively the correct one), hence do some initial sequential steps first. If initial is non-negative, then initial first steps are performed. If initial is negative, then \text{back\_steps - initial} initial steps are performed. The latter ensures that none of the very first \( N \) variables are returned.

Default: \(-10\).

**Details**

\text{RPdirect} is based on the well-known method for simulating any multivariate Gaussian distribution, using the square root of the covariance matrix. The method is pretty slow and limited to about 8000 points, i.e. a 20x20x20 grid in three dimensions. This implementation can use the Cholesky decomposition and the singular value decomposition. It allows for arbitrary points and arbitrary grids.

\text{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**

\text{RPsequential} returns an object of class \text{RMmodel}

**Author(s)**

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

**References**


**See Also**

\text{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 <- 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)
```
Strokorb’s Functions

Covariance Models for Random Vector Fields

Description

Here the code of the paper on ‘Covariance Models for Random Vector Fields’ is given.

Author(s)

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

References


Examples

```r
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

```r
RMn2r(phi)
RMn3b(phi)
RMmps(phi)
```
Strokorb’s Functions

Arguments

phi  a model for a tail correlation function belonging to the Gneiting class $H_d$

Details

RMM2r used with RPSmith defines a monotone shape function that corresponds to a tail correlation function belonging to Gneiting’s class $H_d$. Currently, the function is implemented for dimensions 1 and 3. Called as such it returns the corresponding monotone function.

RMM3b used with RPSmith defines balls with random radius that corresponds to a tail correlation function belonging to Gneiting’s class $H_d$. Currently, the function is implemented for dimensions 1 and 3. (Note that in Strokorb et al. (2014) the density function for twice the radius is considered.) Called as such it returns the corresponding density function for the radius of the balls.

RMMps used with RPSmith defines random hyperplane polygons that corresponds a tail correlation function belonging to Gneiting’s class $H_d$. It currently only allows for RMBrownresnick(RMfbm(alpha=1)) and dimension 2. Called as such it returns the tcf defined by the submodel – this definition may change in future.

Value

object of class RMMmodel

Author(s)

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

References


See Also

RFsimulate, RMMmodel.

Examples

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

model <- RMBrownresnick(RMfbm(alpha=1.5, s=0.2))
plot(RM2r(model))

x <- seq(0, 10, 0.005)
z <- RFsimulate(RPSmith(RM2r(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 function 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 $\gamma \leq 1$.
- **RMEexp**: Exponential model.
- **RMgencauchy**: Generalized Cauchy family with $\alpha \leq 1$ (and arbitrary $\beta > 0$).
- **RMMatern**: Whittle-Matern model with $\nu \leq 1/2$.
- **RMSstable**: 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**

- **RMMnugget**: 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.
Further Operators

- **RMbernoulli**: correlation of binary fields
- **RMbrownresnick**: tcf of a Brown-Resnick process
- **RMschlather**: tcf of a Schlather process

See **RMmodels** for cartesian models.

Author(s)

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

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)
```

**Description**

The Turning Bands method is a simulation method for stationary, isotropic random fields in any dimension and defined on arbitrary points or arbitrary grids. It performs a multidimensional simulation by superposing lower-dimensional fields. In fact, the Turning Bands method is called with the Turning Bands model, see **RMtbm**.

For details see **RMtbm**.
Usage

\texttt{RPrbm(phi, boxcox, fulldim, reduceddim, layers, lines, linessimufactor, linesimustep, center, points)}

Arguments

\begin{itemize}
\item \textbf{phi} \hspace{1cm} object of class \texttt{RMmodel}; specifies the covariance function to be simulated; a univariate stationary isotropic covariance model (see \texttt{RFgetModelNames(type="positive definite"), which is valid in dimension fulldim.}
\item \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.
\item \textbf{fulldim} \hspace{1cm} a positive integer. The dimension of the space of the random field to be simulated
\item \textbf{reduceddim} \hspace{1cm} a positive integer; less than fulldim. The dimension of the auxiliary hyperplane (most frequently a line, i.e. reduceddim=1 used in the simulation.
\item \textbf{layers} \hspace{1cm} a boolean value; for space time model. 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 \texttt{FALSE} then turning layers may never be used.
\end{itemize}
Default: \texttt{TRUE}.

\begin{itemize}
\item \textbf{lines} \hspace{1cm} Number of lines used. Default: \texttt{60}.
\item \textbf{linessimufactor} \hspace{1cm} linessimufactor or linesimustep must be non-negative; if linesimustep is positive then linesimufactor is ignored. If both arguments are naught then points is used (and must be positive). The grid on the line is linesimufactor-times finer than the smallest distance. See also linesimustep.
\end{itemize}
Default: \texttt{2.0}.

\begin{itemize}
\item \textbf{linesimustep} \hspace{1cm} If linesimustep is positive the grid on the line has lag linesimustep. See also linesimufactor.
\end{itemize}
Default: \texttt{0.0}.

\begin{itemize}
\item \textbf{center} \hspace{1cm} Scalar or vector. If not \texttt{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.
\end{itemize}
Default: \texttt{NA}.

\begin{itemize}
\item \textbf{points} \hspace{1cm} integer. If greater than 0, points gives the number of points simulated on the TBM line, hence must be greater than the minimal number of points given by the size of the simulated field and the two paramters 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.
\end{itemize}
Default: \texttt{0}.

Details

- 2-dimensional case
  It is generally difficult to use the turning bands method (RPrbm) directly in the 2-dimensional space. Instead, 2-dimensional random fields are frequently obtained by simulating a 3-dimensional
random field (using \texttt{Rptbm}) and taking a 2-dimensional cross-section. See also the arguments \texttt{fulldim} and \texttt{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 \texttt{RMnsst}, and multiplicate models with that separate the time component.

\textbf{Value}

\texttt{Rptbm} returns an object of class \texttt{RMmodel}

\textbf{Note}

Both the precision and the simulation time depend heavily on \texttt{linesimustep} and \texttt{linesimufactor}. For covariance models with larger values of the scale parameter, \texttt{linesimufactor}=2 is too small.

\textbf{Author(s)}

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

\textbf{References}

\textbf{Turning bands}


\textbf{Turning layers}


\textbf{See Also}

\texttt{Gaussian}, \texttt{RP}, \texttt{Rpspectral}

\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

# 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)
\end{verbatim}
Trend Modelling

Description

The coding of trend, 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>

See Also

RFcalc, RM, RMmodels, RMtrend, RMmodelsMultivariate

Examples

```r
require(geoR)
data(ca20) ## data set from geoR
car20.df <- as.data.frame(car20)
head(car20.df)
RFoptions(coordnames=c("east", "north"), varnames="data")

## covariance model with variance, scale and nugget to be estimated;
## just to abbreviate later on
```
weather

\[
M \leftarrow \text{RMexp(var=NA, scale=NA)} + \text{RMnugget(var=NA)}
\]

## short definition of a trend using the fact that ca20.df is a
data.frame
ca20.RFmod02 <- ~ 1 + altitude + M
(ca20.fit02.RF <- RFFit(ca20.RFmod02, data=ca20.df))

## long definition, which allows also for more general constructions
ca20.RFmod02 <- NA + NA*RMcovariate(ca20.df$altitude) + M
(ca20.fit02.RF <- RFFit(ca20.RFmod02, data=ca20.df))

## Note that the following also works.
## Here, the covariance model must be the first summand
ca20.RFmod02 <- M + NA + ca20.df$altitude
print(ca20.fit02.RF <- RFFit(ca20.RFmod02, data=ca20.df))

### The following does NOT work, as R assumes (NA + ca20.df$altitude) + M
(ca20.RFmod02 <- NA + ca20.df$altitude + M)
try(ca20.fit02.RF <- RFFit(ca20.RFmod02, data=ca20.df)) ### error ...

### factors:
ca20.RFmod03 <- ~ 1 + area + M ###
(ca20.fit03.RF <- RFFit(ca20.RFmod03, data=ca20.df))

---

**weather**

*Pressure and temperature forecast errors over the Pacific Northwest*

### Description

Meteorological dataset, which consists of difference between forecasts and observations (forecasts minus observations) of temperature and pressure at 157 locations in the North American Pacific Northwest.

### Usage

`data(weather)`

### Format

The data frame `weather` contains the following columns:

- **pressure** in units of Pascal
- **temperature** in units of degree Celcius
lon longitudinal coordinates of the locations
lat latitude coordinates of the locations

Furthermore, some results obtained from the data analysis in jss14 are delivered that are pars.model, pars, whole.model, whole.

Finally, the variable information contains packing information (the date and the version of RandomFields)

Details

The forecasts are from the GFS member of the University of Washington regional numerical weather prediction ensemble (UWME; Grimit and Mass 2002; Eckel and Mass 2005); they were valid on December 18, 2003 at 4 pm local time, at a forecast horizon of 48 hours.

Source

The data were obtained from Cliff Mass and Jeff Baars in the University of Washington Department of Atmospheric Sciences.

References


See Also

A reanalysis has been performed in Section 5 of the jss14 paper

Examples

```r
## see 'jss14'
```
Index

*Topic classes
  RFempVariog-class, 75
  RFFit-class, 81
  RFgridDataFrame-class, 104
  RFpointsDataFrame-class, 144
  RFsp-class, 158
  RFspatialGridDataFrame-class, 160
  RFspatialPointsDataFrame-class, 163
  RModel-class, 241
  RModelFit-class, 244
  RModelgenerator-class, 245

*Topic datasets
  soil, 325
  weather, 343

*Topic distribution
  Distribution Families, 24

*Topic hplot
  RFempVariog-class, 75
  RFFit-class, 81
  RFgridDataFrame-class, 104
  RFpointsDataFrame-class, 144
  RFspatialGridDataFrame-class, 160
  RFspatialPointsDataFrame-class, 163
  RModel-class, 241
  RModelFit-class, 244
  RModelgenerator-class, 245

*Topic htest
  RFsplit-class, 146

*Topic methods
  Brown-Resnick-Specific, 10
  Circulant Embedding, 14
  Coins, 18
  Hyperplane, 35
  Independent Variables, 37
  plot-method, 57
  Specific, 329
  Spectral, 330

Square roots, 334
Tbm, 339

*Topic models
  Mathematical C functions, 45
  RFempiricalvariogram, 72
  RFpar, 143
  RFsp2conventional, 159
  RMangle, 165
  RMaskey, 166
  RMave, 168
  RMBcw, 170
  RMBernoulli, 172
  RMBessel, 173
  RMBigneiting, 175
  RMBiwm, 177
  RMCauchy, 183
  RMCauchyTbm, 184
  RMCchoquet, 186
  RMCircular, 187
  RMCconstant, 188
  RMCcoord, 189
  RMCovariante, 190
  RMCoxisham, 191
  RMExponential, 193
  RMCurlfree, 194
  RMcutoff, 195
  RMCcutoff, 195
  RMDagum, 197
  RMDampedcos, 198
  RMDelay, 199
  RMDewijsian, 201
  RMDivfree, 202
  RMeaxx, 203
  RMEpscauchy, 204
  RMX, 206
  RMEponential, 207
  RMPFbm, 209
  RMFixcov, 210
  RMPfixed, 212
  RMPflatpower, 212

345
INDEX

rmfractdiff, 214
rmfractgauss, 215
rmgauss, 216
rmgencauchy, 217
rmgenfbm, 219
rmgengneiting, 220
rmgennsst, 222
rmgneiting, 223
rmgneitingdiff, 225
rmhyperbolic, 226
rmIaco, 227
rmid, 228
rminexp, 230
rmintrinsic, 231
rmkolmogorov, 233
rmLgd, 234
rmMa, 235
rmmasterstein, 236
rmMatrix, 238
rmmodel, 239
rmmppplus, 256
rmqam, 257
rmrational, 258
rmmultiquad, 259
rmnthsstwm, 262
rmssst, 263
rmnugget, 264
rpmarswm, 266
rpmlütt, 267
rmpolynome, 270
rmpower, 271
rmprod, 273
rmrational, 276
rmpolynomial, 270
rmpsum, 290
rmtbm, 291
rmt provisional, 293
rmtraj, 295
rmtruncsupport, 297
rmuser, 298
rmvector, 300
rmwave, 301
rmwhittlematern, 302
rmwhittlematern, 302
rrdetermin, 312
rrdistr, 312
rrgauss, 314
rrloc, 315
rrmcmc, 316
rrrectangular, 317
rrspheric, 319
rrunif, 320
Spherical models, 332
Tail Correlation Functions, 338
*Topic optimize
  fitgauss, 29
  Rfit, 77
  Rffitoptimiser, 84
*Topic print
  RFempVariog-class, 75
  RFfit-class, 81
  RFgridDataFrame-class, 104
  RFpointsDataFrame-class, 144
  RFSpatialGridDataFrame-class, 160
  RFSpatialPointsDataFrame-class, 163
  Rmmodel-class, 241
  RmmodelFit-class, 244
*Topic spatial
  BrownResnick, 12
  Constants, 19
  conventionalRFspDataFrame, 20
  Coordinate systems, 21
  Distribution Families, 24
  Extremal t, 26
  ExtremalGaussian, 27
  fitgauss, 29
  GaussianFields, 30
  GSPSJ06, 33
  Hierarchical Modelling, 34
  Internal functions, 38
  jss14, 40
  Mathematical C functions, 45
  Max-stable random fields, 49
  Max-stable random fields,
INDEX

advanced, 51
Obsolete Functions, 53
Others, 55
papers, 56
PrintModellist, 63
RandomFields-package, 6
RFboxcox, 64
RFcov, 66
RFcrossvalidate, 69
RFdistr, 71
RFempiricalvariogram, 72
RFfit, 77
RFfitoptimiser, 84
RFformula, 86
RFfractaldim, 89
RFfunction, 92
RFgetMethodNames, 93
RFgetModel, 97
RFgetModelInfo, 99
RFgetModelNames, 101
RFgui, 106
RFhurst, 108
RFinterpolate, 111
RFinearpart, 114
RFloglikelihood, 116
RFoldstyle, 119
RFoptions, 120
RFpar, 143
RFratiotest, 146
RFsimulate, 148
RFsimulateAdvanced, 153
RFsp2conventional, 159
RMangle, 165
RMaskey, 166
RMave, 168
RMBall, 170
RMBcw, 170
RMBernoulli, 172
RMBessel, 173
RMBigneiting, 175
RMBivm, 177
RMBbr2bg, 179
RMBbr2eg, 180
RMBrownresnick, 182
RMcauchy, 183
RMcauchytbm, 184
RMMchoquet, 186
RMCircular, 187
RMconstant, 188
RMCoord, 189
RCovariate, 190
RMcoxisham, 191
RCubic, 193
RCurlfree, 194
RCutoff, 195
RDDagum, 197
RDdampedcos, 198
RDdelay, 199
RDewijsian, 201
RDivfree, 202
RMeaxxa, 203
RMeaxxa, 204
RMepscauchy, 206
RMExp, 206
RMExponential, 207
RFbm, 209
RFfixcov, 210
RFflatpower, 212
RFFractdiff, 214
RFFractgauss, 215
RFGauss, 216
RMgencauchy, 217
RMgenfbm, 219
RMEngneiting, 220
RMEngneiting, 220
RMEngneiting, 222
RMEngneiting, 223
RMEngneitingdiff, 225
RMHyperbolic, 226
RMiaco, 227
RMinv, 228
RMintern, 229
RMinexp, 230
RMintrinsic, 231
RMMolgogorov, 233
RMLdg, 234
RMMma, 235
RMMaststein, 236
RMMatrix, 238
RMMmodel, 239
RMMmodels Overview, 248
RMMmodelsAdvanced, 249
RMMmodelsMultivariate, 252
RMMmodelsNonstationary, 254
RMMmodelsSpacetime, 255
RMMppplus, 256
RMMqam, 257
RMMult, 258
INDEX

RMMultiquad, 259
RMNonstwm, 262
RMMnst, 263
RMnugget, 264
RMParswm, 266
RMPenta, 267
Rmplus, 268
RMPolygon, 269
RMPolyvnom, 270
RMPower, 271
RMPod, 273
RMPam, 274
RMQexp, 275
RMRational, 276
RMRotorat, 277
RMS, 278
RMSchlather, 279
RMschur, 281
RMsigt, 282
RMsinepower, 283
RMSpheric, 284
RMSstatic, 286
RMStein, 287
RMTsp, 288
RMTsum, 290
RMTbm, 291
RMTtrafo, 293
RMTrend, 295
RMTriuncsustain, 297
RMuser, 298
RMvector, 300
RMIwave, 301
RMwhittlematern, 302
RPbernoulli, 305
RPhl2, 306
RPgauss, 307
RPoisson, 308
RPprocess, 309
Rplt, 310
RRrderm, 312
RRrdrst, 312
RRrgauss, 314
RRrloc, 315
RRmcnc, 316
RRrectangular, 317
RRrspheric, 319
RRrunif, 320
S02, 321
S10, 321
SBS14, 322
Smith, 324
sp2RF, 327
Spherical models, 332
SS12, 336
Strokrb's Functions, 336
Tail Correlation Functions, 338
Trend Modelling, 342
*, 255
*(RMMult), 258
*, RMMmodel, RMMmodel-method
(RMMmodel-class), 241
*, RMMmodel, logical-method
(RMMmodel-class), 241
*, RMMmodel, numeric-method
(RMMmodel-class), 241
*, logical, RMMmodel-method
(RMMmodel-class), 241
*, numeric, RMMmodel-method
(RMMmodel-class), 241
+, 87, 103, 255
+(RMPplus), 268
++ (RMPppplus), 256
+, RMMmodel, RMMmodel-method
(RMMmodel-class), 241
+, RMMmodel, logical-method
(RMMmodel-class), 241
+, RMMmodel, numeric-method
(RMMmodel-class), 241
+, logical, RMMmodel-method
(RMMmodel-class), 241
+, numeric, RMMmodel-method
(RMMmodel-class), 241
-(Mathematial C functions), 45
-, RMMmodel, RMMmodel-method
(RMMmodel-class), 241
-, RMMmodel, logical-method
(RMMmodel-class), 241
-, RMMmodel, numeric-method
(RMMmodel-class), 241
-, logical, RMMmodel-method
(RMMmodel-class), 241
-, numeric, RMMmodel-method
(RMMmodel-class), 241
.RF_fit (RFFit-class), 81
.RFFit (RFFit-class), 81
Random. seed, 146, 156
INDEX

(Mathematical C functions), 45

[][], rmmodel, ANY, ANY, ANY-method
(RMmodel-class), 241

[][], rmmodel, ANY, ANY-method
(RMmodel-class), 239

[][], rmmodel, ANY, ANY-method
(RMmodel-class), 241

[][], rmmodelFit, ANY, ANY, ANY-method
(RMmodelFit-class), 244

[][], rmmodelFit, ANY, ANY-method
(RMmodelFit-class), 244

[][], rmmodelFit-method (RMmodelFit-class), 241

[][], rmmodelgenerator, ANY, ANY, ANY-method
(RMmodelgenerator-class), 245

[][], rmmodelgenerator, ANY, ANY-method
(RMmodelgenerator-class), 245

[][], rmmodelgenerator-method
(RMmodelgenerator-class), 245

<=, RFgridDataFrame, ANY, ANY, ANY-method
(RFgridDataFrame-class), 104

<=, RFpointsDataFrame, ANY, ANY, ANY-method
(RFpointsDataFrame-class), 144

<=, RFsp, ANY, ANY, ANY-method
(RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158

<=, RFsp, ANY, ANY-method (RFsp-class), 158
abs (Mathematical C functions), 45
acosh (Mathematical C functions), 45
Advanced RM models, 103
Advanced RM models (RM models advanced), 249
AIC, 7
AIC, RFit-method (RFit-class), 81
AIC.RF_fit (RFfit-class), 81
AICC, 7
AICC.RF_fit (RFfit-class), 81
AICC.RFpointsFit (RFpointsFit-class), 81
Aniso, 250, 254
anova, 7, 130
anova, RFfit-method (RFfit-class), 81
anova, RMmodelfit-method (RMmodelfit-class), 244
anova.RF_fit (RFfit-class), 81
anova.RMmodelfit (RMmodelfit-class), 244
approx_step, 31
areamat, 124, 125
areamat=1.124
as.array.RFgridDataFrame (RFgridDataFrame-class), 104
as.array.RFpointsDataFrame (RFpointsDataFrame-class), 144
as.array.RFspatialGridDataFrame (RFspatialGridDataFrame-class), 160
as.array.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 163
as.data.frame.RFgridDataFrame
(RFgridDataFrame-class), 104
as.data.frame.RFpointsDataFrame
(RFpointsDataFrame-class), 144
as.data.frame.RFspatialGridDataFrame
(RFspatialGridDataFrame-class), 160
as.data.frame.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class), 163
as.matrix.RFgridDataFrame
(RFgridDataFrame-class), 104
as.matrix.RFpointsDataFrame
(RFpointsDataFrame-class), 144
as.matrix.RFspatialGridDataFrame
(RFspatialGridDataFrame-class), 160
as.matrix.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class), 163
as.vector.RFgridDataFrame
(RFgridDataFrame-class), 104
as.vector.RFpointsDataFrame
(RFpointsDataFrame-class), 144
as.vector.RFspatialGridDataFrame
(RFspatialGridDataFrame-class), 160
as.vector.RFspatialPointsDataFrame
(RFspatialPointsDataFrame-class), 163
asin (Mathematical C functions), 45
asinh (Mathematical C functions), 45
atan (Mathematical C functions), 45
atan2 (Mathematical C functions), 45
atanh (Mathematical C functions), 45
Auxiliary Models (Others), 55
Auxiliary RM models, 251
Auxiliary RM models (Others), 55
Auxiliary Models (Others), 55
Average (Coins), 18
back_steps, 31
Bayesian, 45, 65, 115, 117, 248
Bayesian (Hierarchical Modelling), 34
Bayesian Modelling, 6, 25, 313
Bayesian Modelling (Hierarchical Modelling), 34
BIC, 7
BIC, RFit-method (RFfit-class), 81
BIC.RF_fit (RFfit-class), 81
Binary field, 310
Binary fields, 148, 153
INDEX

binary processes, 45
BR methods (Brown-Resnick-Specific), 10
Brown-Resnick, 39
Brown-Resnick (BrownResnick), 12
Brown-Resnick process (BrownResnick), 12
Brown-Resnick-Specific, 10
BrownResnick, 12

c (Mathematical C functions), 45
coerce, RFemVarioGelist-method (RFemVarioG-class), 75
coerce, RFit, RFemVarioG-method (RFFit-class), 81
coerce, RFgridDataFrame, RFpointsDataFrame, ANY-method
(RFgridDataFrame-class), 104
coerce, RFgridDataFrame, RFpointsDataFrame-method
(RFgridDataFrame-class), 104
coerce, RFpointsDataFrame, RFgridDataFrame, ANY-method
(RFpointsDataFrame-class), 144
coerce, RFpointsDataFrame, RFgridDataFrame-method
(RFpointsDataFrame-class), 144
coerce, RFspatialGridDataFrame, data.frame-method
(RFspatialGridDataFrame-class), 160
coerce, RFspatialGridDataFrame, RFpointsDataFrame-method
(RFspatialGridDataFrame-class), 160
coerce, RFspatialPointsDataFrame, data.frame-method
(RFspatialPointsDataFrame-class), 163
coerce, RFspatialPointsDataFrame, RFspatialGridDataFrame, ANY-method
(RFspatialPointsDataFrame-class), 163
coerce, RFspatialPointsDataFrame, RFspatialGridDataFrame-method
(RFspatialPointsDataFrame-class), 163
coerce, SpatialGridDataFrame, RFspatialGridDataFrame-method
(RFspatialGridDataFrame-class), 160
coerce, SpatialPointsDataFrame, RFspatialPointsDataFrame-method
(RFspatialPointsDataFrame-class), 163

Coins, 18, 134

Composed Poisson, 310

compound Poisson processes, 45
CondSimu (Obsolete Functions), 53

Constants, 19
constants, 102, 104

contour, 161
contour (RFspatialDataFrame)
(RFspatialDataFrame-class), 160

conventionalRFspDataFrame, 20, 105, 144

coord_units (Changings), 13

coordinate system, 47
coordinate system (Coordinate systems), 21

Cos (Mathematical C functions), 45

Covariance (Obsolete Functions), 53

CovarianceFct (Obsolete Functions), 53
gaussian (gkr

CovMatrix (Obsolete Functions), 53  
CRS, 160, 163  
Cutoff (Circulant Embedding), 14  
data.frame, 105, 144, 160, 163  
DeleteAllRegisters (Obsolete Functions), 53  
DeleteRegister (Obsolete Functions), 53  
Dependencies (Internal functions), 38  
dimensions, RFDataFrame-method  
(RFsp-class), 158  
dimensions, RFsp-class, 158  
dimensions, RFsp-class, 158  
Direct (Square roots), 334  
dist, 73, 154, 293, 294  
Distribution Families, 24  
Distributions, 45  
DoSimulateRF (Obsolete Functions), 53  
Earth models (Spherical models), 332  
earth models (Spherical models), 332  
EmpiricalVariogram (Obsolete Functions), 53  
error function model (RMBrownresnick), 182  
exp (Mathematical C functions), 45  
expml (Mathematical C functions), 45  
extremal Gaussian (ExtremalGaussian), 27  
extremal Gaussian process  
(ExtremalGaussian), 27  
extremal t, 26  
extremal t (Extremal t), 26  
extremal t process (Extremal t), 26  
extremalGaussian, 27  
FinalizeExample (Internal functions), 38  
fitgauss, 29, 79, 146  
fitvario (Obsolete Functions), 53  
floor (Mathematical C functions), 45  
formula, 148, 153, 154  
formula notation, 45, 240  
fractal.dim (Obsolete Functions), 53  
function, 245, 246  
gamma (Mathematical C functions), 45  
Gaussian, 17, 19, 36, 38, 308, 310, 330, 331, 335, 341  
Gaussian (GaussianFields), 30  
Gaussian random fields, 69, 77, 79, 148  
GaussianFields, 30  
GaussRF (Obsolete Functions), 53  
GetModelList (PrintModelList), 63  
GetModelNames (PrintModelNames), 63  
GKS11, 56  
GKS11 (weather), 343  
GridTopology, 20, 73, 78, 105, 111, 148, 154, 160  
GridTopology2gridVectors, GridTopology-method  
(RFspatialGridDataFrame-class), 160  
GridTopology2gridVectors, matrix-method  
(RFspatialGridDataFrame-class), 160  
GSPSj06, 33, 56, 57  
Hierarchical (Hierarchical Modelling), 34  
Hierarchical Modelling, 34  
hist.RFgridDataFrame  
(RFgridDataFrame-class), 104  
hist.RFpointsDataFrame  
(RFpointsDataFrame-class), 144  
hist.RFspatialGridDataFrame  
(RFspatialGridDataFrame-class), 160  
hist.RFspatialPointsDataFrame  
(RFspatialPointsDataFrame-class), 163  
hurst (Obsolete Functions), 53  
Hyperplane, 35  
Hyperplanes (Hyperplane), 35  
image, 77  
Independent Variables, 37  
InitGaussRF (Obsolete Functions), 53  
InitMaxStableRF (Obsolete Functions), 53  
InitSimulateRF (Obsolete Functions), 53  
iinteractive, 131  
Internal functions, 38  
Intrinsic (Circulant Embedding), 14  
Inverse multiquadric (RMmultiquad), 259  
isGridded (RFsp-class), 158  
isGridded, RFsp-class, 158  
jpeg, 132  
JSS14, 40, 57, 344  
Kriging (Obsolete Functions), 53
kriging (RFinterpolate), 111

lambda, 124
lgamma (Mathematical C functions), 45
linear models, 69, 77, 79
lines.RModel (RModel-class), 241
lm, 91, 109, 110
log (Mathematical C functions), 45
log1p (Mathematical C functions), 45
log2 (Mathematical C functions), 45
logb (Mathematical C functions), 45
logLik, 7
logLik.RF_fit (RFfit-class), 81
logLik.Rfit (Rffit-class), 81

M2 (Smith), 324
M3 (Smith), 324
Major Revisions, 43
MajorRevisions (Major Revisions), 43
math.c (Mathematical C functions), 45
Mathematical C functions, 45
Mathematical functions, 248
matrix.methods, 31
max (Mathematical C functions), 45
Max-stable random fields, 49
Max-stable random fields, advanced, 51
Maxstable, 310
Maxstable (Max-stable random fields), 49
maxstable, 11, 13, 27, 28, 325
maxstable (Max-stable random fields), 49
maxstable processes, 45
maxstableAdvanced, 11, 13, 27, 28, 49, 50, 180, 181, 325
maxstableAdvanced (Max-stable random fields, advanced), 51
MaxStableRF (Obsolescent Functions), 53
method specification, 155
min (Mathematical C functions), 45
mixed model, 230, 246
mixed moving maxima (Smith), 324
moving maxima (Smith), 324
multiquadric family (RMmultiquad), 259
Multivariate and vector valued random fields, 45
Multivariate RMmodels
(RMmodelsMultivariate), 252

new.env, 313
new_coord_units (Changes), 13
non-stationary (RMmodelsNonstationary), 254
non-stationary RMmodels
(RMmodelsNonstationary), 254
Nonstationary RMmodels
(RMmodelsNonstationary), 254
norm, 314, 315
Normal, 314
Nugget (Independent Variables), 37

Obsolete Functions, 53
optim, 78, 79, 84, 128–130
options, 44
Other models (Others), 55
Others, 55

papers, 6, 56
par, 143
dfd, 132
persp, 77, 83, 161
persp, RFempVariog-method
(RFempVariog-class), 75
persp, Rfit-method (RFfit-class), 81
persp, RFspatialGridDataFrame-method
(plot-method), 57
plot, 6, 7, 131
plot, RFDataFrame, data.frame-method
(plot-method), 57
plot, RFDataFrame, matrix-method
(plot-method), 57
plot, RFDataFrame, missing-method
(plot-method), 57
plot, RFDataFrame, RFDataFrame-method
(plot-method), 57
plot, RFempVariog, missing-method
(RFempVariog-class), 75
plot, RFfit, missing-method
(RFFit-class), 81
plot, RFgridDataFrame, missing-method
(plot-method), 57
plot, RFpointsDataFrame, missing-method
(plot-method), 57
plot, RFspatialDataFrame, data.frame-method
(plot-method), 57
plot, RFspatialDataFrame, matrix-method
(plot-method), 57
plot(RFspatialDataFrame, missing-method)
   (plot-method), 57
plot(RFspatialDataFrame, RFspatialGridDataFrame-method)
   (plot-method), 57
plot(RFspatialDataFrame, RFspatialPointsDataFrame-method)
   (plot-method), 57
plot(RFspatialGridDataFrame, missing-method)
   (plot-method), 57
plot(RFspatialGridDataFrame, RFspatialPointsDataFrame-method)
   (plot-method), 57
plot(RFspatialPointsDataFrame, RFspatialPointsDataFrame-method)
   (plot-method), 57
plot(RFspatialPointsDataFrame, RFspatialGridDataFrame-method)
   (plot-method), 57
plot(RMmodel, missing-method)
   (RMmodel-class), 241
plot-method, 57, 143
plotWithCircles (Internal functions), 38
points.RMmodel (RMmodel-class), 241
Poisson spline (RMmultiquad), 259
powered error function
   (RMbrownresnick), 182
powered exponential, 254
powered exponential (RMstable), 286
print, 7
print, RFempVariog-method
   (RFempVariog-class), 75
print, RFFit-method (RFFit-class), 81
print, RMmodelFit-method
   (RMmodelFit-class), 244
print.crossvalidate (RFCrossvalidate), 69
print.RF_empVariog (RFempVariog-class), 75
print.RF_fit (RFFit-class), 81
print.RFFit (RFFit-class), 81
print.RFgridDataFrame
   (RFgridDataFrame-class), 104
print.RFpointsDataFrame
   (RFpointsDataFrame-class), 144
print.RFractionTest (RFractionTest), 146
print.RFspatialGridDataFrame
   (RFspatialGridDataFrame-class), 160
print.RFspatialPointsDataFrame
   (RFspatialPointsDataFrame-class),
INDEX

R.l1round (Mathematial C functions), 45
R.log (Mathematial C functions), 45
R.log1p (Mathematial C functions), 45
R.log2 (Mathematial C functions), 45
R.logb (Mathematial C functions), 45
R.ln (Mathematial C functions), 45
R.lrint (Mathematial C functions), 45
R.lround (Mathematial C functions), 45
R.minus, 242, 295
R.minus (Mathematial C functions), 45
R.models, 92, 251, 253, 254
R.models (Mathematial C functions), 45
R.multi (Mathematial C functions), 45
R.nearbyint (Mathematial C functions), 45
R.nextafter (Mathematial C functions), 45
R.nexttoward (Mathematial C functions), 45
R.p (Mathematial C functions), 45
R.plus (Mathematial C functions), 45
R.pow (Mathematial C functions), 45
R.remainder (Mathematial C functions), 45
R.rint (Mathematial C functions), 45
R.round (Mathematial C functions), 45
R.sin (Mathematial C functions), 45
R.sinh (Mathematial C functions), 45
R.sqrt (Mathematial C functions), 45
R.tan (Mathematial C functions), 45
R.tanh (Mathematial C functions), 45
R.tgamma (Mathematial C functions), 45
R.trunc (Mathematial C functions), 45
Random parameters, 312
RandomFields, 14, 70, 79, 88, 97, 104, 113, 140, 147
RandomFields (RandomFields-package), 6
RandomFields-package, 6
range.RFgridDataFrame (RFgridDataFrame-class), 104
range.RFpointsDataFrame (RFpointsDataFrame-class), 144
range.RFspatialGridDataFrame (RFspatialGridDataFrame-class), 160
range.RFspatialPointsDataFrame (RFspatialPointsDataFrame-class), 163
raster, 73, 78, 111, 148, 154
RC, 9, 25, 93, 241, 249, 310
RC (Constants), 19
RC_CARTESIAN_COORD, 293
RC_CARTESIAN_COORD (Constants), 19
RC_DOMAIN_NAMES, 104, 299
RC_DOMAIN_NAMES (Constants), 19
RC_EARTH_COORDS (Constants), 19
RC_GNOMONIC_PROJ, 293
RC_GNOMONIC_PROJ (Constants), 19
RC_ISOVALUES_NAMES, 104, 293, 299
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RC_ISOVALUES (Constants), 19
RF, 9, 20, 25, 44, 55, 100, 241, 246, 249, 310
RF (RFfit), 92
RF__name__, 229
RF_fit (RFfit-class), 81
RFboxcox, 7, 15, 18, 35, 37, 64, 65, 306, 307, 311, 329, 331, 334, 340
RFcalc, 7, 67, 92, 342
RFcalc (Mathematial C functions), 45
RFcov, 7, 54, 66, 66, 67, 92, 241, 246
RFcovmatrix, 7, 54, 66, 67, 92
RFcovmatrix (RFcov), 66
RFCrossvalidate, 7, 69, 92, 120, 121, 127
RFDATAFRAME (RFsp-class), 158
RFDATAFRAME (RFsp-class), 158
RFDISTR (RFDISTR), 71
RFDISTR, 25, 71, 92, 123, 313
RFearth2cartesian, 23, 93
RFearth2cartesian (RMtrafo), 293
RFearth2dist, 93
RFearth2dist (RMtrafo), 293
RFempiricalvariogram, 6, 44, 54, 65, 72, 73, 74, 77, 84, 92, 96, 107, 113, 150, 157
RFempVARIO, 75, 81, 242
<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFsimulateAdvanced</td>
<td>32, 78, 111, 148–150, 152, 153, 153</td>
</tr>
<tr>
<td>RFsp.</td>
<td>20, 37, 59, 73, 81, 106, 111, 145, 149, 150, 154–156, 159, 161, 162, 164, 242, 244, 328</td>
</tr>
<tr>
<td>RFsp-class</td>
<td>158</td>
</tr>
<tr>
<td>RFsp2conventional</td>
<td>159</td>
</tr>
<tr>
<td>RFspatialDataframe (RFsp-class)</td>
<td>158</td>
</tr>
<tr>
<td>RFspatialDataFrame-class (RFsp-class)</td>
<td>158</td>
</tr>
<tr>
<td>RFspatialGridDataFrame.</td>
<td>21, 61, 106, 150, 156, 158–161, 164</td>
</tr>
<tr>
<td>RFspatialPointsDataFrame</td>
<td>159, 161, 163</td>
</tr>
<tr>
<td>RFspatialPointsDataFrame-class</td>
<td>160</td>
</tr>
<tr>
<td>RFspatialPointsDataFrame-class (RFspatialPointsDataFrame-class)</td>
<td>163</td>
</tr>
<tr>
<td>RFspatialPointsDataFrame-class (RFspatialPointsDataFrame-class)</td>
<td>163</td>
</tr>
<tr>
<td>RFspDataframe2conventional</td>
<td>156, 159, 161, 164</td>
</tr>
<tr>
<td>RFspDataframe2conventional (RFsp2conventional)</td>
<td>159</td>
</tr>
<tr>
<td>RFspDataframe2conventional, RFgridDataFrame-method</td>
<td>104</td>
</tr>
<tr>
<td>RFspDataframe2conventional, RFpointsDataFrame-method</td>
<td>144</td>
</tr>
<tr>
<td>RFspDataframe2conventional, RFspatialGridDataFrame-class</td>
<td>160</td>
</tr>
<tr>
<td>RFspDataframe2conventional, RFspatialGridDataFrame-class (RFspatialGridDataFrame-class)</td>
<td>163</td>
</tr>
<tr>
<td>RFspDataframe2dataArray</td>
<td>159, 161, 164</td>
</tr>
<tr>
<td>RFspDataframe2dataArray (RFsp2conventional)</td>
<td>159</td>
</tr>
<tr>
<td>RFspDataframe2dataArray, RFgridDataFrame-method</td>
<td>104</td>
</tr>
<tr>
<td>RFspDataframe2dataArray, RFspatialGridDataFrame-class (RFspatialGridDataFrame-class)</td>
<td>160</td>
</tr>
<tr>
<td>RFvariogram.</td>
<td>7, 54, 66, 67, 92</td>
</tr>
<tr>
<td>RFvariogram (RFcov)</td>
<td>66</td>
</tr>
<tr>
<td>RM.</td>
<td>7, 9, 20, 25, 44, 56, 93, 241, 252–254, 256, 310, 339, 342</td>
</tr>
<tr>
<td>RM (RMmodels Overview)</td>
<td>248</td>
</tr>
<tr>
<td>RM_modelFit-class (RMmodelFit-class)</td>
<td>244</td>
</tr>
<tr>
<td>RMangle.</td>
<td>55, 165, 165, 239, 240, 251, 294</td>
</tr>
<tr>
<td>RMaskey.</td>
<td>166, 167, 177, 220, 221, 249, 332, 338</td>
</tr>
<tr>
<td>RMave.</td>
<td>168, 168, 169, 255</td>
</tr>
<tr>
<td>RMBall.</td>
<td>25, 55, 170, 270, 319</td>
</tr>
<tr>
<td>RMBcw.</td>
<td>170, 171, 219, 220, 332, 338</td>
</tr>
<tr>
<td>RMBernoulli.</td>
<td>172, 172, 250, 305, 339</td>
</tr>
<tr>
<td>RMBessel.</td>
<td>173, 173, 174, 198, 199, 249, 301, 302</td>
</tr>
<tr>
<td>RMBigweining.</td>
<td>167, 175, 175, 176, 221, 224, 225, 252</td>
</tr>
<tr>
<td>RMBiendland (RMBigweining)</td>
<td>175</td>
</tr>
<tr>
<td>RMBiwm.</td>
<td>177, 177, 178, 252, 267, 304</td>
</tr>
<tr>
<td>RMBr2bg.</td>
<td>179, 179, 181</td>
</tr>
<tr>
<td>RMBr2eg.</td>
<td>180, 180, 181</td>
</tr>
<tr>
<td>RMBrownresnick.</td>
<td>182, 182, 337, 339</td>
</tr>
<tr>
<td>RMCcardinalsine (RMCwave).</td>
<td>301</td>
</tr>
<tr>
<td>RMCcauchy.</td>
<td>103, 183, 183, 184, 185, 204, 205, 217–219, 226, 227, 240</td>
</tr>
<tr>
<td>RMCcauchytbmb.</td>
<td>184, 184, 185, 205, 219</td>
</tr>
<tr>
<td>RMCchoquet.</td>
<td>186, 186, 260, 283, 284</td>
</tr>
<tr>
<td>RMCcircular.</td>
<td>103, 187, 187, 188, 249, 332, 338</td>
</tr>
<tr>
<td>RMCconstant.</td>
<td>138, 188, 189, 189, 249, 332, 338</td>
</tr>
<tr>
<td>RMCov.</td>
<td>56, 189, 189</td>
</tr>
<tr>
<td>RMCovariate.</td>
<td>123, 190, 191, 211, 250, 299</td>
</tr>
<tr>
<td>RMCoxisham.</td>
<td>191, 191, 192, 255</td>
</tr>
<tr>
<td>RMCubic.</td>
<td>193, 193, 249, 332, 338</td>
</tr>
<tr>
<td>RMCut.</td>
<td>94, 194, 202, 203, 233, 252, 253, 255, 301</td>
</tr>
<tr>
<td>RMCutoff.</td>
<td>14, 16, 195, 195, 196, 250</td>
</tr>
<tr>
<td>RMcircular.</td>
<td>249, 332, 338</td>
</tr>
<tr>
<td>RMcddatamethod.</td>
<td>174, 198, 198, 199, 249</td>
</tr>
<tr>
<td>RMDelay.</td>
<td>103, 199, 200, 252</td>
</tr>
<tr>
<td>RMDewijsian.</td>
<td>170, 171, 201, 201, 250</td>
</tr>
<tr>
<td>RMDivfree.</td>
<td>194, 195, 202, 202, 233, 252, 253, 255, 301</td>
</tr>
<tr>
<td>RMeaxxa (RMMeaxxa)</td>
<td>55, 203, 204</td>
</tr>
<tr>
<td>RMeaspauchy.</td>
<td>204, 204, 205</td>
</tr>
<tr>
<td>RMetaxxa (RMMeaxxa)</td>
<td>203</td>
</tr>
<tr>
<td>RMexp.</td>
<td>103, 206, 206, 240, 286, 287, 303, 304, 332, 338</td>
</tr>
<tr>
<td>RMexponential.</td>
<td>207, 207, 208, 250, 252, 257, 274</td>
</tr>
</tbody>
</table>
RMfixcov, 44, 123, 189, 191, 210, 210, 211, 240, 250, 299
RMfixed, 86, 87, 212, 240
RMflatpower, 212, 212, 213, 220, 250
RMfractdiff, 214, 214, 249
RMfractgauss, 215, 215, 249
RMgencauchy, 170, 171, 183–185, 196, 217, 217, 218, 234, 240, 254, 332, 338
RMgenfbm, 170, 171, 209, 210, 212, 213, 219, 219, 250
RMgengneiting, 167, 175–177, 220, 220, 221, 223–225, 249, 322, 332, 338
RMgennst, 222, 222, 255, 264
RMgenst, 167, 177, 217, 220, 221, 223, 223, 224, 225, 240, 332, 338
RMgenstingdiff, 225, 225, 249
RMhyperbolic, 184, 226, 226, 227, 249, 304
RMIaco, 227, 228, 255
RMid, 55, 228, 229
RMintern, 229
RMintexp, 230, 230, 231, 250
RMintinsic, 14, 16, 231, 231, 232, 250
RMjbbessel (RMBessel), 173
RMBessel (RMBhittletamon), 302
RMBolmogorov, 233, 233, 252
RMlgd, 205, 218, 234, 234, 235, 249
RMm2r, 55, 180, 181, 183, 324
RMm2r (Stokor's Functions), 336
RMm3b, 55, 183
RMm3b (Stokor's Functions), 336
RMma, 235, 235, 236, 249
RMmastein, 236, 236, 237, 255
RMmatern, 34, 178, 217, 240, 262, 302, 303, 332, 338
RMmatern (RMBhittletamon), 302
RMmatrix, 238, 238, 253, 281, 297
RMmixed (RMMttern), 229
RMMult (RMM.ttern), 229
RMParswm, 178, 253, 266, 266
RMParswmX (RMParswm), 266
RMPenta, 249, 267, 267, 268
RMPolygon, 55, 170, 269
RMPolynome, 251, 270, 271
RMPower, 249, 250, 271, 271, 272
RMMultinverse (RMM.ttern), 229
RMMultquad, 186, 187, 259, 259, 260
RMNqam, 98, 250, 261, 261
RMMstw, 254, 262, 304
RMMstst, 222, 225, 263, 263, 341
RMMugget, 37, 87, 103, 154, 240, 264, 264, 264, 265, 332, 338
RMMnull (RMM.ttern), 229
RMMstw, 178, 253, 266, 266
INDEX

RPmodel (RPprocess), 309
RPmodels (RPprocess), 309
RPmppplus (RMintern), 229
RPmult (RMintern), 229
RPnugget, 30, 31, 135, 265
RPnugget (Independent Variables), 37
RPopitz, 49, 308
RPopitz (Extremal t), 26
RPplus (RMintern), 229
RPoisson, 34, 297, 308
RPprocess, 309
RPprocesses (RPprocess), 309
RPS (RMintern), 229
RPschlather, 49, 180, 181, 280, 308
RPschlather (Extremal Gaussian), 27
RPsequential, 30, 31, 121, 135, 335
RPsequential (Square roots), 334
RPsimson, 49, 153, 246, 256, 337
RPsmrith, 49, 153, 246, 256, 337
RPsmrith (Smith), 324
RPspecific, 30, 31, 329
RPspecific (Specific), 329
RPspectral, 19, 30, 31, 38, 121, 137, 341
RPspectral (Spectral), 330
RPt, 308, 310
RPtbm, 19, 30, 31, 38, 121, 125, 137, 291, 292, 329, 331
RPtbm (Tbm), 339
RPtrend (RMintern), 229
RR, 9, 20, 35, 44, 55, 72, 93, 241, 246, 249, 282, 310, 313, 317
RR (Distribution Families), 24
RRarcsqrt (RMintern), 229
RRdeterm, 25, 312, 312
RRdistr, 25, 34, 71, 312, 312, 313–315, 317, 318, 320
RRggauss, 25, 72, 217, 308, 312, 314, 314, 315, 318, 320
RRloc, 25, 315, 315
RRmcmc, 316, 316
RRmodel (Distribution Families), 24
RRmodels, 34, 247
RRmodels (Distribution Families), 24
RRrectangular, 120, 126, 317, 318, 324
RRsetdistr (RMIntern), 229
RRsign (RMsign), 282
RRspheric, 25, 319, 319, 320
RRunif, 25, 315, 320, 320
S02, 57, 321
S10, 40, 57, 204, 277, 278, 321
SBS14, 57, 322
scale, 31
Schlather, 339
Schoenberg’s representation (RMchoquet), 186
ScreenDevice (Internal functions), 38
seq, 107
Sequential (Square roots), 334
set.seed, 123, 146
show, RFempVariog-method (RFempVariog-class), 75
show, RFFit-method (RFFit-class), 81
show, RFgridDataFrame-method (RFgridDataFrame-class), 104
show, RFpointsDataFrame-method (RFpointsDataFrame-class), 144
show, RFspatialGridDataFrame-method (RFspatialGridDataFrame-class), 160
show, RFspatialPointsDataFrame-method (RFspatialPointsDataFrame-class), 163
show, RMmodel-method (RMmodel-class), 241
show, RMmodelFit-method (RMmodelFit-class), 244
show, RMmodelgenerator-method (RMmodelgenerator-class), 245
showManpages (Internal functions), 38
sin (Mathematical C functions), 45
sine power function (RMsinepower), 283
sinh (Mathematical C functions), 45
Smith, 324
Sobolev (RMwhittlematern), 302
soil, 6, 108, 325
sp2RF, 7, 159, 160, 162–164, 327, 328
space-time (RMmodelsSpacetime), 255
spam, 31, 334
Spatial, 158, 163
SpatialGridDataFrame, 105, 160, 161
SpatialPoints, 163
SpatialPointsDataFrame, 144, 163
spConform, 79
spConform=FALSE, 7
spConform=TRUE, 7
Specific, 329
Spectral, 330
sphere (Spherical models), 332
Spherical models, 248, 332
spherical models, 6, 22, 285
spherical models (Spherical models), 332
sqrt (Mathematial C functions), 45
Square roots, 334
SS12, 40, 56, 57, 336
StartExample (Internal functions), 38
stationary max-stable random fields, 148, 153
stationary Poisson fields, 148, 153
str, 7, 243
str. RMmodel (RMmodel-class), 241
Strokorb’s Functions, 336
summary, 7
summary, RFempVariog-method
(RFempVariog-class), 75
summary, RFfit-method (RFfit-class), 81
summary, RFfit-method (RFfit-class), 81
summary, RFsp-method (RFsp-class), 158
summary, RMmodelfit-method
(RMmodelfit-class), 244
summary.crossvalidate
(RFcrossvalidate), 69
summary. RF_empVariog
(RFempVariog-class), 75
summary. RF_fit (RFfit-class), 81
summary. RM_modelfit (RMmodelfit-class), 244
Swave, 122
t field, 310
t fields, 153
Tail correlation functions, 248
Tail correlation functions (Tail
Correlation Functioncs), 338
tail correlation functions, 45
tail correlation functions (Tail
Correlation Functioncs), 338
Tail Correlation Functioncs, 338
tan (Mathematial C functions), 45
tanh (Mathematial C functions), 45
tb, 339
tbmdim (Changings), 13
tcf (Tail Correlation Functioncs), 338
trend, 45
trend (Rmtrend), 295
Trend Modelling, 342
trend modelling, 241, 248
trend modelling (Trend Modelling), 342
trunc (Mathematial C functions), 45
truncated power function (RMaskey), 166
unif, 320
user, 45
variab_units (Changings), 13
variance, RFsp-method (RFsp-class), 158
Variogram (Obsolete Functions), 53
weather, 6, 40, 57, 70, 79, 147, 343
which submodels, 98
whittle-matern, 254
whittle-matern (RMwhittle-matern), 302
zenit (Coordinate systems), 21