Package 'spup'

July 3, 2018

Type Package

Title Spatial Uncertainty Propagation Analysis

Version 1.3-1

Date 2018-06-14

Description
Uncertainty propagation analysis in spatial environmental modelling following methodology described in Heuvelink et al. (2007) <doi:10.1080/13658810601063951> and Brown and Heuvelink (2007) <doi:10.1016/j.cageo.2006.06.015>. The package provides functions for examining the uncertainty propagation starting from input data and model parameters, via the environmental model onto model outputs. The functions include uncertainty model specification, stochastic simulation and propagation of uncertainty using Monte Carlo (MC) techniques. Uncertain variables are described by probability distributions. Both numerical and categorical data types are handled. Spatial auto-correlation within an attribute and cross-correlation between attributes is accommodated for. The MC realizations may be used as input to the environmental models called from R, or externally.

Depends R (>= 3.3.0)

Imports graphics, gstat, magrittr, methods, mvtnorm, purrr, raster, whisker

Suggests dplyr, GGally, gridExtra, knitr, png, readr, sp, testthat

License GPL (>= 3)

LazyData true

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Kasia Sawicka [aut, cre],
Gerard Heuvelink [aut],
Dennis Walvoort [aut],
Stefan van Dam [ctb],
Damiano Luzzi [ctb]

Maintainer Kasia Sawicka <katwic55@ceh.ac.uk>

Repository CRAN

Date/Publication 2018-07-03 14:20:03 UTC

R topics documented:

check_distribution *Simple check if distribution provided in defineUM() belongs to a list of supported distributions.*

Description

Simple check if distribution provided in defineUM() belongs to a list of supported distributions.

Usage

```
check_distribution(object)
```

Arguments

object	Any R object. In defineUM() it is used to examine if selected distribution is in supported list of ditributions.

Value

TRUE or FALSE.

Author(s)

Kasia Sawicka

check_if_Spatial *Simple check if class of provided object is Spatial*

Description

Simple check if class of provided object is Spatial

Usage

```
check_if_Spatial(object)
```

Arguments

object	Any R object. In defineUM() it is used to examine what type of data are delt with.

Value

TRUE or FALSE.

Author(s)

Kasia Sawicka

crm2vgm *Converting a spatial correlogram model to a variogram model*

Description

Used internally in genSample() in case of sampling by unconditional gaussian simulation.

Usage

```
crm2vgm(crm)
```

Arguments

crm	object of a class "SpatialCorrelogramModel", output of makeCRM().

Details

To assure equalfinality the sill parameter for spatially correlated random residuals is fixed and standardized to 1.

Value

An object of a class "variogramModel" extending data.frame.

Author(s)

Kasia Sawicka, Gerard Heuvelink

defineMUM *Define Mulivariate Uncertainty Model*

Description

Function that uses output of defineUM() to define joint probability distribution for uncertain cross-correlated variables.

Usage

```
defineMUM(UMlist, cormatrix, ...)
```

Arguments

UMlist	a list of uncertain objects creaded in defineUM().
cormatrix	matrix of cross-correlations.
...	additional parameters.

Details

The cormatrix is a square matrix of correlations, dimensionally equal to the number of objects, symmetrical (transposed must be the same as original), diagonal must all be 1 all values must be <-1, +1>) and all eigenvalues must be > 0. The marginal Um objects must have provided id.

Value

Object of a class "JointNumericSpatial" or "JointScalar".

Author(s)

Kasia Sawicka, Gerard Heuvelink

Examples

```
set.seed(12345)

data(OC, OC_sd, TN, TN_sd)
OC_crm <- makeCRM(acf0 = 0.6, range = 5000, model = "Sph")
OC_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
class(OC_UM)
TN_crm <- makeCRM(acf0 = 0.4, range = 5000, model = "Sph")
TN_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(TN, TN_sd), crm = TN_crm, id = "TN")
class(TN_UM)

soil_prop <- list(OC_UM,TN_UM)
mySpatialMUM <- defineMUM(soil_prop, matrix(c(1,0.7,0.7,1), nrow=2, ncol=2))
class(mySpatialMUM)

# scalar
scalarUM <- defineUM(uncertain = TRUE, distribution = "norm",
                     distr_param = c(1, 2), id="Var1")
scalarUM2 <- defineUM(uncertain = TRUE, distribution = "norm",
                      distr_param = c(3, 2), id="Var2")
scalarUM3 <- defineUM(uncertain = TRUE, distribution = "norm",
                      distr_param = c(10, 2.5), id="Var3")
myMUM <- defineMUM(UMlist = list(scalarUM, scalarUM2, scalarUM3),
               matrix(c(1,0.7,0.2,0.7,1,0.5,0.2,0.5,1), nrow = 3, ncol = 3))
class(myMUM)
```

`defineUM` *Define an uncertainty model for a single variable*

Description

Function that allows to define marginal uncertainty distributions for model inputs and subsequent Monte Carlo analysis.

Usage

```
defineUM(uncertain = TRUE, distribution = NULL, distr_param = NULL,
  crm = NULL, categories = NULL, cat_prob = NULL, id = NULL, ...)
```

Arguments

uncertain	"TRUE" or "FALSE", determines if specification of Uncertainty Model (UM) is needed. Currently not in use, but provided for furture implementation of contributions analysis.
distribution	a string that specifies which distribution to sample from. Only in use for continuos or discrete numerical variables. See Details for a list of supported distributions.
distr_param	a vector or a list with distribution parameters. For example, for the normal distribution in case of a spatial variable this must be a map of means and a map of standard deviations. Only in use for continuous or discrete numerical variables.
crm	a correlogram model, object of a class "SpatialCorrelogramModel", output of makecormodel(). Can only be specified for numerical variables.
categories	a vector of categories. Only in use for categorical (e.g. saved as character) or discrete numerical variables.
cat_prob	spatial data frame or raster stack; a list of probabilities for the vector of categories. Number of columns in the data frame cannot be smaller than number of categories. Only in use for categorical (e.g. saved as character) or discrete numerical variables.
id	identifier of the variable; only in use if the UM defined here is to be used in defineUM() to construct a joint UM for numerical variables.
...	additional parameters.

Details

If the uncertain object is a spatial object, the distribution parameters or the probabilities for categories must be provided by means of maps, for example if a spatial variable has a normal distribution, a map of means and standard deviations must be provided. If crm is provided and spatial correlation between the residuals is assumed only the normal distribution for residuals is allowed.

If no spatial correlations between residuals is assumed, allowed distributions for marginal uncertainty models are listed in Table 1.

Table 1 Parametric probability models allowed in defineUM(). For more details look up ?distribution.

Distribution	**Syntax**	**Parameters**
beta	"beta"	$shape1$, $shape2$, ncp
binomial	"binom"	$size$, $prob$
Cauchy	"cauchy"	$location$, $scale$
chi-squared	"chisq"	df, ncp
exponential	"exp"	$rate$
gamma	"gamma"	$shape$, $rate$

geometric	"geom"	$prob$
hypergeometric	"hyper"	m, n, k
log-normal	"lnorm"	$meanlog$, $sdlog$
negative binomial	"nbinom"	$size$, $prob$, mu
normal	"norm"	$mean$, sd
Poisson	"pois"	$lambda$
Student's	"t"	df, ncp
uniform	"unif"	min, max
Weibull	"weibull"	$shape$, $scale$

Value

Object of a class "MarginalXxx" that includes all necessary information for creating realizations of the uncertain variable. If provided arguments are: type of the distribution and corresponding parameters, and corresponding parameters are spatial objects - an object of class "MarginalNumericSpatial". If provided arguments are: type of the distribution and corresponding parameters, and corresponding parameters are non-spatial objects - an object of class "MarginalNumericSpatial". If provided arguments are: categories and probabilities, and probabilities are saved in a spatial object - an object of class "MarginalCategoricalSpatial". If provided arguments are: categories and probabilities, and probabilities are saved in a non-spatial object - an object of class "MarginalCategoricalDataFrame".

Author(s)

Kasia Sawicka, Gerard Heuvelink

Examples

```
# define uncertainty model for spatial numerical variable
data(dem30m, dem30m_sd)
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)
class(demUM)

# define uncertainty model for spatial categorical variable
data(woon)
woonUM <- defineUM(TRUE, categories = c(1,2,3), cat_prob = woon[, c(4:6)])
class(woonUM)

# define uncertainty model for a variable desribed by a scalar
scalarUM <- defineUM(uncertain = TRUE, distribution = "gamma", distr_param = c(1,2))
class(scalarUM)

# define uncertainty model for two spatial cross-correlated variables
data(OC, OC_sd, TN, TN_sd)

OC_crm <- makeCRM(acf0 = 0.6, range = 1000, model = "Sph")
OC_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
class(OC_UM)
```

```
TN_crm <- makeCRM(acf0 = 0.4, range = 1000, model = "Sph")
TN_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(TN, TN_sd), crm = TN_crm, id = "TN")
class(TN_UM)
```

---

dem30m *Digital Elevation Model of Zlatibor region in Serbia.*

Description

A dataset containing the mean an example Digital Elevation Model.

Usage

```
data(dem30m)
```

Format

a SpatialGridDataFrame with 15000 rows and 1 variable:

Elevation Digital Elevation Model, in meters

Source

The Zlatibor dataset was kindly provided by Prof. Branislav Bajat from the University of Belgrade, Serbia.

---

dem30m_sd *Standard deviation of Digital Elevation Model of Zlatibor region in Serbia.*

Description

A dataset containing the sd of an example Digital Elevation Model. It was calculated from dem30m using terrain funtion from raster package (opt = 'roughness').

Usage

```
data(dem30m_sd)
```

Format

a SpatialGridDataFrame with 15000 rows and 1 variable:

Elevation_sd Standard deviation of Digital Elevation Model, in meters

Source

The Zlatibor dataset was kindly provided by Prof. Branislav Bajat from the University of Belgrade, Serbia.

`distribution_sampling` *Sampling from a given distribution*

Description

Sampling from a given distribution

Usage

```
distribution_sampling(n, distribution, parameters)
```

Arguments

`n`	number of sampling runs
`distribution`	A string describing selected distribution. The same as a part of the string following the "r" in each random variate generation function in ?distributions.
`parameters`	vector of parameters to pass to the random variate generation function after number of observations.

Value

Sample of random deviates.

Author(s)

Kasia Sawicka

`distribution_sampling_raster`
Sampling from a given distribution

Description

Only used in samplemethod "randomSampling" for MarginalNumericSpatial.

Usage

```
distribution_sampling_raster(distribution, parameters_stack)
```

Arguments

distribution	A string describing selected distribution. The same as a part of the string following the "r" in each random variate generation function in ?distributions.
parameters_stack	parameters to pass to the random variate generation function after number of observations.

Value

Sample of random deviates.

Author(s)

Kasia Sawicka

---

executable *Wrapper function for calling executables in R*

Description

Wrapper function for calling executables in R

Usage

```
executable(filename)
```

Arguments

filename	a path with a name to the .exe file to be wrapped here.

Value

Executable output.

Author(s)

Dennis Walvoort

`find_strata` *Sampling from a given distribution*

Description

Sampling from a given distribution

Usage

```
find_strata(p, distribution, parameters, ...)
```

Arguments

`p`	a vector of quantiles.
`distribution`	a string indicating which distribution to sample from. See ?defineUM() for Details.
`parameters`	parameters to pass to the appropriate sampling funtion, e.g. mean and sd for "norm" distribution.
`...`	additional parameters.

Value

Strata of the distribution defined by given quantiles.

Author(s)

Kasia Sawicka, Stefan van Dam

`genSample` *Methods for generating Monte Carlo realizations from uncertain inputs.*

Description

Methods for classes: "MarginalNumericSpatial", "MarginalScalar", "MarginalCategoricalSpatial", "JointNumericSpatial", "JointScalar". Function that runs Monte Carlo simulations depending on the type of uncertain object. Facilitates unconditional Gaussian simulation of errors for spatially auto-correlated residuals, as well as random and stratified random sampling if no spatial auto-correlation is included.

Usage

```
genSample(UMobject, n, samplemethod, p = 0, asList = TRUE,
  debug.level = 1, ...)
```

Arguments

UMobject	an uncertain object to sample from, output of defineUM() or defineMUM().
n	integer, number of Monte Carlo realizations.
samplemethod	a string, "ugs", "randomSampling", "stratifiedSampling", "lhs" ("lhs" currently not in use).
p	A vector of quantiles. Optional. Only required if sample method is "stratifiedSampling" or "lhs".
asList	logical. If asList = TRUE returns list of all samples as a list. If asList = FALSE returns samples in a format of distribution parameters in UMobject.
debug.level	integer; set gstat internal debug level, see below for useful values. If set to -1 (or any negative value), a progress counter is printed.
...	Additional parameters that may be passed, e.g. in the "ugs" method. See examples.

Details

Sampling methods:

"ugs" Unconditional Gaussian simulation of spatially auto-correlated and/or cross-correlated errors.

"randomSampling" Sampling multivariate distribution using eigenvalue decomposition (based on 'mvtnorm' package).

"stratifiedSampling" Number of samples (n) must be dividable by the number of quantiles to assure that each quantile is evenly represented.

"lhs" Not implemented yet. Sampling method for at least two uncertain inputs. The uncertain.object is then a list of two or more. It uses a stratified sampling method to generate inputs for the latin hypercube algorithm.

NOTE. Version 1.3-1 includes bug fixing related to derivation of cross-correlation matrix for multivariate uncertainty propagation analysis.

Value

A Monte Carlo sample of the variables of interest. If asList = TRUE returns list of all samples as lists.

Author(s)

Kasia Sawicka, Stefan van Dam, Gerard Heuvelink

Examples

```
set.seed(12345)

### ------------------- "MarginalNumericSpatial" -------------------
# load data
data(dem30m, dem30m_sd)
```

```
# "ugs" method example
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)

# toy example
dem_sample <- genSample(UMobject = demUM, n = 2, samplemethod = "ugs", nmax = 4, asList = FALSE)
str(dem_sample)
# any meaningful Monte Carlo analysis should have normally much larger number of runs
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 100, samplemethod = "ugs", nmax = 20, asList = FALSE)
str(dem_sample)

## End(Not run)

# "startifiedSampling" method example
demUM <- defineUM(uncertain = TRUE, distribution = "norm", distr_param = c(dem30m, dem30m_sd))
# toy example
dem_sample <- genSample(UMobject = demUM, n = 5, samplemethod = "stratifiedSampling", p = 0:5/5)
# any meaningful Monte Carlo analysis should have normally much larger number of runs
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 100, samplemethod = "stratifiedSampling", p = 0:5/5)
str(dem_sample)

## End(Not run)

# Examples with rasters
# (raster with auto-correlation)
data(OC, OC_sd)
OC_crm <- makeCRM(acf0 = 0.6, range = 1000, model = "Sph")
OC_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
class(OC_UM)
# toy example
some_sample <- genSample(OC_UM, n = 2, "ugs", nmax = 4)
some_sample
# any meaningful Monte Carlo analysis should have normally much larger number of runs
## Not run:
some_sample <- genSample(OC_UM, n = 50, "ugs", nmax = 24)
some_sample

## End(Not run)


### ----------------------- "MarginalScalar" -----------------------
# example with normal distribution
scalarUM <- defineUM(uncertain = TRUE, distribution = "norm", distr_param = c(10, 1))
scalar_sample <- genSample(scalarUM, n = 10, samplemethod = "randomSampling")


### ----------------- "MarginalCategoricalSpatial" -----------------
# load data
data(woon)
```

```
woonUM <- defineUM(TRUE, categories = c(1,2,3), cat_prob = woon[, c(4:6)])
woon_sample <- genSample(woonUM, 10, asList = FALSE)
class(woon_sample)
str(woon_sample@data)


### -------------------- "JointNumericSpatial" ----------------------
# load data
data(OC, OC_sd, TN, TN_sd)

# define marginal UMs
OC_crm <- makeCRM(acf0 = 0.6, range = 1000, model = "Sph")
OC_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
TN_crm <- makeCRM(acf0 = 0.4, range = 1000, model = "Sph")
TN_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(TN, TN_sd), crm = TN_crm, id = "TN")

# define joint UM
soil_prop <- list(OC_UM, TN_UM)
mySpatialMUM <- defineMUM(soil_prop, matrix(c(1,0.7,0.7,1), nrow=2, ncol=2))
class(mySpatialMUM)

# sample - "ugs" method
# toy example
my_cross_sample <- genSample(mySpatialMUM, n = 2, "ugs", nmax = 4, asList = TRUE)
class(my_cross_sample)
# any meaningful Monte Carlo analysis should have normally much larger number of runs
## Not run:
my_cross_sample <- genSample(mySpatialMUM, n = 100, "ugs", nmax = 24, asList = TRUE)
class(my_cross_sample)

## End(Not run)


### ------------------------ "JointScalar" ------------------------
scalarUM <- defineUM(uncertain = TRUE, distribution = "norm",
                     distr_param = c(1, 2), id="Var1")
scalarUM2 <- defineUM(uncertain = TRUE, distribution = "norm",
                      distr_param = c(3, 2), id="Var2")
scalarUM3 <- defineUM(uncertain = TRUE, distribution = "norm",
                      distr_param = c(10, 2.5), id="Var3")
myMUM <- defineMUM(UMlist = list(scalarUM, scalarUM2, scalarUM3),
               matrix(c(1,0.7,0.2,0.7,1,0.5,0.2,0.5,1), nrow = 3, ncol = 3))
my_sample <- genSample(myMUM, n = 5, samplemethod = "randomSampling", asList = FALSE)
my_sample
```

`genSample.JointNumericSpatial` *Generating Monte Carlo sample from a list of uncertain objects that are cross-correlated.*

Description

Uncertain objects are described by joint PDF or a list from independent objects. Sampling can be done via three different sampling methods:

Usage

```
## S3 method for class 'JointNumericSpatial'
genSample(UMobject, n, samplemethod, p = 0,
  asList = TRUE, debug.level = 1, ...)
```

Arguments

`UMobject`	object of a class JointNumericSpatial. Output of defineMUM().
`n`	Integer. Number of Monte Carlo realizations.
`samplemethod`	"ugs" for spatially cross-correlated errors, "randomSampling" for joint PDF of non-spatial variables, "lhs" if no correlation of errors is considered.
`p`	A vector of quantiles. Optional. Only required if sample method is "lhs".
`asList`	Logical. If TRUE return sample in a form of a list, if FALSE returnsample in a format of distribution parameters.
`debug.level`	integer; set gstat internal debug level, see below for useful values. If set to -1 (or any negative value), a progress counter is printed.
`...`	Additional parameters that may be passed, e.g. in the "ugs" method. See examples.

Details

"ugs" Unconditional gaussian simulation of spatially cross-correlated errors.

"randomSampling" Sampling multivariate distribution using eigenvalue decomposition (based on 'mvtnorm' package).

"lhs" Not implemented yet. Sampling method for at least two uncertain inputs. The uncertain.object is then a list of two or more. It uses startified sampling method to generate the inputs for the latin hypercude algorithm, hence number of samples (n) must be dividable by the number of quantiles to assure each quantile is evenly represented.

NOTE. Version 1.3-1 includes bug fixing related to derivation of cross-correlation matrix for multivariate uncertainty propagation analysis.

Value

A Monte Carlo sample of the variables of interest. If asList = TRUE returns list of all samples as lists.

Author(s)

Kasia Sawicka, Stefan van Dam, Gerard Heuvelink

Examples

```
set.seed(12345)
# "ugs" method example
# load data
data(OC, OC_sd, TN, TN_sd)

# Test for SpatialGridDataFrames
OC <- as(OC, 'SpatialGridDataFrame')
TN <- as(TN, 'SpatialGridDataFrame')
OC_sd <- as(OC_sd, 'SpatialGridDataFrame')
TN_sd <- as(TN_sd, 'SpatialGridDataFrame')

# define marginal UMs
OC_crm <- makeCRM(acf0 = 0.6, range = 5000, model = "Sph")
OC_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
TN_crm <- makeCRM(acf0 = 0.4, range = 5000, model = "Sph")
TN_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(TN, TN_sd), crm = TN_crm, id = "TN")

# define joint UM
soil_prop <- list(OC_UM, TN_UM)
mySpatialMUM <- defineMUM(soil_prop, matrix(c(1,0.7,0.7,1), nrow=2, ncol=2))

# sample - "ugs" method
# toy example
my_cross_sample <- genSample(mySpatialMUM, n = 3, "ugs", nmax = 24, asList = TRUE)
class(my_cross_sample)
## Not run:
my_cross_sample <- genSample(mySpatialMUM, n = 50, "ugs", nmax = 24, asList = TRUE)
class(my_cross_sample)

## End(Not run)
```

---

genSample.JointScalar *Generating sample from cross-correlated variables described by a scalar.*

Description

Generating sample from cross-correlated variables described by a scalar.

Usage

```
## S3 method for class 'JointScalar'
genSample(UMobject, n, samplemethod, p = 0,
  asList = TRUE, ...)
```

Arguments

UMobject	object of a class JointScalar created using defineMUM.R
n	integer; number of Monte Carlo runs
samplemethod	"randomSampling" or "lhs".
p	a vector of quantiles. Optional. Only required if sample method is "lhs".
asList	logical. If asList = TRUE returns list of all samples as a list. If asList = FALSE returns samples in a format of distribution parameters in UMobject.
...	Additional parameters.

Value

Monte Carlo sample of cross-correlated scalar variables.

Author(s)

Kasia Sawicka, Gerard Heuvelink

Examples

```
set.seed(12345)
scalarUM <- defineUM(uncertain = TRUE, distribution = "norm",
                     distr_param = c(1, 2), id="Var1")
scalarUM2 <- defineUM(uncertain = TRUE, distribution = "norm",
                      distr_param = c(3, 2), id="Var2")
scalarUM3 <- defineUM(uncertain = TRUE, distribution = "norm",
                      distr_param = c(10, 2.5), id="Var3")
myMUM <- defineMUM(UMlist = list(scalarUM, scalarUM2, scalarUM3),
               matrix(c(1,0.7,0.2,0.7,1,0.5,0.2,0.5,1), nrow = 3, ncol = 3))
my_sample <- genSample(myMUM, n = 10, samplemethod = "randomSampling", asList = FALSE)
my_sample
```

---

genSample.MarginalCategoricalSpatial

Generating Monte Carlo sample from an uncertain object of a class 'MarginalCategoricalSpatial'

Description

Generating Monte Carlo sample from an uncertain object of a class 'MarginalCategoricalSpatial'

Usage

```
## S3 method for class 'MarginalCategoricalSpatial'
genSample(UMobject, n, samplemethod,
  p = 0, asList = TRUE, ...)
```

Arguments

UMobject	uncertain object defined using defineUM().
n	Integer. Number of Monte Carlo realizations.
samplemethod	not in use for categorical variables.
p	not in use for categorical variables.
asList	logical. If asList = TRUE returns list of all samples as a list. If asList = FALSE returns samples in a format of distribution parameters in UMobject.
...	additional parameters

Value

A Monte Carlo sample of a categorical spatial variable.

Author(s)

Kasia Sawicka

Examples

```
set.seed(12345)
# load data
data(woon)
woonUM <- defineUM(TRUE, categories = c(1,2,3), cat_prob = woon[, c(4:6)])
woon_sample <- genSample(woonUM, 10, asList = FALSE)
class(woon_sample)
str(woon_sample@data)
woon_sample <- genSample(woonUM, 10)
class(woon_sample)

# analyse probability of having snow
# load data
data(dem30m, dem30m_sd)

# generate dummy probabilities for categories "snow" and "no snow"
dem30m$snow_prob <- NA
dem30m$snow_prob[dem30m$Elevation > 1000] <- 0.75
dem30m$snow_prob[dem30m$Elevation <= 1000] <- 0.25
dem30m$no_snow_prob <- 1 - dem30m$snow_prob
summary(dem30m@data)
snowUM <- defineUM(uncertain = TRUE, categories = c("snow", "no snow"), cat_prob = dem30m[2:3])
class(snowUM)
snow_sample <- genSample(snowUM, 10, asList = FALSE)
```

```
head(snow_sample@data)

# case with raster
# load data
data(dem30m, dem30m_sd)
dem30m$snow_prob <- NA
dem30m$snow_prob[dem30m$Elevation > 1000] <- 0.75
dem30m$snow_prob[dem30m$Elevation <= 1000] <- 0.25
dem30m$no_snow_prob <- 1 - dem30m$snow_prob
summary(dem30m@data)
dem_stack <- raster::stack(dem30m)
snowUM <- defineUM(uncertain = TRUE, categories = c("snow", "no snow"), cat_prob = dem_stack[[2:3]])
snow_sample <- genSample(snowUM, 10, asList = FALSE)
require(sp)
spplot(snow_sample)
```

`genSample.MarginalNumericSpatial` *Generating Monte Carlo sample from an uncertain object of a class 'MarginalNumericSpatial'*

Description

Function that runs Monte Carlo simulations depending on the type of uncertain object. Facilitates unconditional gausian simulation of errors for spatially auto-correlated residulas, and random sampling, stratified sampling if no spatial auto-correlation is included.

Usage

```
## S3 method for class 'MarginalNumericSpatial'
genSample(UMobject, n, samplemethod, p = 0,
  asList = TRUE, debug.level = 1, ...)
```

Arguments

`UMobject`	uncertain object defined using defineUM().
`n`	Integer. Number of Monte Carlo realizations.
`samplemethod`	"ugs" for spatially correlated errors, "randomSampling" and "stratifiedSampling" if no spatial correlation of errors is considered.
`p`	A vector of quantiles. Optional. Only required if sample method is "stratifiedSampling" or "lhs".
`asList`	logical. If asList = TRUE returns list of all samples as a list. If asList = FALSE returns samples in a format of distribution parameters in UMobject.
`debug.level`	integer; set gstat internal debug level, see below for useful values. If set to -1 (or any negative value), a progress counter is printed.
`...`	Additional parameters that may be passed, e.g. in the "ugs" method. See examples.

Details

"ugs" Unconditional gaussian simulation of spatially auto-correlated errors.

"stratifiedSampling" Number of samples (n) must be dividable by the number of quantiles to assure each quantile is evenly represented.

"lhs" Sampling method for at least two uncertain inputs. The uncertain.object is then a list of two or more. It uses startified sampling method to generate the inputs for the latin hypercude algorithm, hence the p is restricted as above.

Value

A Monte Carlo sample of uncertain input of a class of distribution parameters.

Author(s)

Kasia Sawicka, Stefan van Dam, Gerard Heuvelink

Examples

```
set.seed(12345)
# load data
data(dem30m, dem30m_sd)

# "ugs" method example
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)

# toy example
dem_sample <- genSample(UMobject = demUM, n = 2, samplemethod = "ugs", nmax = 6, asList = FALSE)
str(dem_sample)
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 50, samplemethod = "ugs", nmax = 20, asList = FALSE)
str(dem_sample)

## End(Not run)


# "startifiedSampling" method example
demUM <- defineUM(uncertain = TRUE, distribution = "norm", distr_param = c(dem30m, dem30m_sd))
# toy example
dem_sample <- genSample(UMobject = demUM, n = 5, samplemethod = "stratifiedSampling", p = 0:5/5)
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 50, samplemethod = "stratifiedSampling", p = 0:5/5)
str(dem_sample)

## End(Not run)

# Examples with rasters
# (raster with auto-correlation)
data(OC, OC_sd)
```

```
OC_crm <- makeCRM(acf0 = 0.6, range = 1000, model = "Sph")
OC_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
class(OC_UM)
# toy example
some_sample <- genSample(OC_UM, n = 3, "ugs", nmax=6)
some_sample
## Not run:
some_sample <- genSample(OC_UM, n = 50, "ugs", nmax=20)
some_sample

## End(Not run)
```

`genSample.MarginalScalar` *Generating Monte Carlo sample from an uncertain object of a class 'MarginalScalar'*

Description

Function that runs Monte Carlo simulations for MarginalScalar class objects.

Usage

```
## S3 method for class 'MarginalScalar'
genSample(UMobject, n, samplemethod, p = 0,
  asList = TRUE, ...)
```

Arguments

`UMobject`	uncertain object defined using defineUM().
`n`	Integer. Number of Monte Carlo realizations.
`samplemethod`	"randomSampling" or "stratifiedSampling".
`p`	A vector of quantiles. Optional. Only required if sample method is "stratifiedSampling".
`asList`	logical. If asList = TRUE returns list of all samples as a list. If asList = FALSE returns samples in a format of distribution parameters in UMobject.
`...`	Additional parameters.

Details

"stratifiedSampling" Number of samples (n) must be dividable by the number of quantiles to assure each quantile is evenly represented.

Value

A Monte Carlo sample of uncertain input of a class of distribution parameters.

Author(s)

Kasia Sawicka

Examples

```
set.seed(12345)
# Example 1
scalarUM <- defineUM(uncertain = TRUE, distribution = "norm", distr_param = c(10, 1))
scalar_sample <- genSample(scalarUM, n = 10, samplemethod = "randomSampling")

# Example 2
scalarUM <- defineUM(uncertain = TRUE, distribution = "beta", distr_param = c(10, 1, 2))
scalar_sample <- genSample(scalarUM, n = 10, samplemethod = "stratifiedSampling", p = 0:5/5)
```

---

list_depth *Function to find the level of list nesting*

Description

Function to find the level of list nesting

Usage

```
list_depth(List)
```

Arguments

List	an object of class 'list'.

Value

an integer; level of list nesting

Author(s)

Kasia Sawicka

Examples

```
a <- list(1,2)
list_depth(a)

a <- list(list(1, 2), 3)
list_depth(a)
```

makeCRM *Defining a spatial correlogram model*

Description

Function that generates a spatial correlogram model, an object of class "SpatialCorrelogramModel".

Usage

```
makeCRM(acf0 = 1, range = NA, model, anis, kappa = 0.5, add.to, covtable,
  Err = 0)
```

Arguments

acf0	Aurocorrelation function value at distance near 0. Default is 1. Must fall in interval [0,1].
range	Range parameter of the correlogram model component.
model	Model type, e.g. "Exp", "Sph", "Gau", "Mat" that vgm() accepts. See ?gstat::vgm() for more #' details.
anis	Anisotropy parameters. See ?gstat::vgm() for more details.
kappa	Smoothness parameter for the Matern class of variogram models. See ?gstat::vgm() for more #' details.
add.to	See ?gstat::vgm() (currently not in use)
covtable	See ?gstat::vgm() (currently not in use)
Err	Numeric. See ?gstat::vgm() for more details.

Details

For the spatial variables allowed autocorrelation functions are listed in Table 4.1 of the gstat manual (http://www.gstat.org/gstat.pdf). Spatial correlation assumes stationarity, i.e. correlation depends only on the separation distance between points in space. Anisotropy is allowed (http://www.gstat.org/gstat.pdf). No nested models are allowed in the current version.

Value

An object of a class "SpatialCorrelogramModel". This is a list collating provided arguments.

Author(s)

Kasia Sawicka, Gerard Heuvelink

Examples

```
mycormodel <- makeCRM(acf0 = 0.8, range = 300, model = "Exp")
str(mycormodel)
```

mean_MC_sgdf *mean() function for MC sample saved in a SpatialGridDataFrame*

Description

Calculates mean from MC realizations for each location in a map.

Usage

```
mean_MC_sgdf(realizations, ...)
```

Arguments

realizations	MC sample saved in SpatialGridDataFrame.
...	additional parameters.

Value

SpatialGridDataFrame; a mean of a MC sample.

Author(s)

Kasia Sawicka

Examples

```
set.seed(12345)
# load data
data(dem30m, dem30m_sd)
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)
dem_sample <- genSample(UMobject = demUM, n = 50, samplemethod = "ugs",
                        nmax = 20, asList = FALSE)
dem_mean <- mean_MC_sgdf(dem_sample)
```

OC — *Soil organic carbon content in a south area (33 x 33km) of lake Alaotra in Madagascar.*

Description

A dataset containing the mean of soil OC content from 0-30 cm layer.

Usage

```
data(OC)
```

Format

a RasterLayer with dimensions : 134, 135, 18090 (nrow, ncol, ncell), resolution : 250, 250 (x, y).

Source

ISRIC soilgrid information. HENGL, T., MENDES DE JESUS, J., HEUVELINK, G. B. M., RUIPEREZ GONZALEZ, M., KILIBARDA, M., BLAGOTIC, A., SHANGGUAN, W., WRIGHT, M. N., GENG, X., BAUER-MARSCHALLINGER, B., GUEVARA, M. A., VARGAS, R., MACMILLAN, R. A., BATJES, N. H., LEENAARS, J. G. B., RIBEIRO, E., WHEELER, I., MANTEL, S. & KEMPEN, B. 2017. SoilGrids250m: Global gridded soil information based on machine learning. PLOS ONE, 12, e0169748.

OC_sd — *Standard deviation of soil organic carbon content in a south area (33 x 33km) of lake Alaotra in Madagascar.*

Description

A dataset containing the standard deviation of soil OC content from 0-30 cm layer.

Usage

```
data(OC_sd)
```

Format

a RasterLayer with dimensions : 134, 135, 18090 (nrow, ncol, ncell), resolution : 250, 250 (x, y).

Source

ISRIC soilgrid information. HENGL, T., MENDES DE JESUS, J., HEUVELINK, G. B. M., RUIPEREZ GONZALEZ, M., KILIBARDA, M., BLAGOTIC, A., SHANGGUAN, W., WRIGHT, M. N., GENG, X., BAUER-MARSCHALLINGER, B., GUEVARA, M. A., VARGAS, R., MACMILLAN, R. A., BATJES, N. H., LEENAARS, J. G. B., RIBEIRO, E., WHEELER, I., MANTEL, S. & KEMPEN, B. 2017. SoilGrids250m: Global gridded soil information based on machine learning. PLOS ONE, 12, e0169748.

`plot.SpatialCorrelogramModel`

Plots correlogram model

Description

Plots correlogram model

Usage

```
## S3 method for class 'SpatialCorrelogramModel'
plot(x, distance = 1, ylim = c(0, 1),
  xlab = "Distance", ylab = "Correlation", ...)
```

Arguments

`x`	Object of class "SpatialCorrelogramModel" as created by makeCRM().
`distance`	minimum distance between locations (unit should correspond with the unit of the range parameter in makeCRM()).
`ylim`	the y limits of the plot.
`xlab`	a title for the x axis.
`ylab`	a title for the y axis.
`...`	additional parameters.

Value

plot of correlogram model

Author(s)

Kasia Sawicka, Gerard Heuvelink

Examples

```
mycormodel <- makeCRM(acf0 = 0.8, range = 300, model = "Exp")
plot(mycormodel, distance = 1)
```

print.template *Print method for class "template."*

Description

Print method for class "template."

Usage

```
## S3 method for class 'template'
print(x, ...)
```

Arguments

x	Object of class "template".
...	additional parameters.

Value

Template file content.

Author(s)

Dennis Walvoort

propagate *Propagation function*

Description

A function that runs a model repeatedly with Monte Carlo samples of uncertain inputs.

Usage

```
propagate(realizations, model, n, ...)
```

Arguments

realizations	a list where each element is a single Monte Carlo realizations if only one parameter/variable is considered uncertain; a list of such lists if more than one parameter/variable is considered uncertain.
model	model that is written as a function in R.
n	number of Monte Carlo Runs.
...	any further arguments that the model takes.

Value

Model output Monte Carlo realizations.

Author(s)

Kasia Sawicka

Examples

```
set.seed(12345)
## continuous spatial data example with a single variable
# load data
data(dem30m, dem30m_sd)

# Slope model
Slope <- function(DEM, ...) {
  require(raster)
  require(purrr)
  demraster <-
    DEM %>%
    raster()
  demraster %>%
    terrain(opt = 'slope', ...) %>%
    as("SpatialGridDataFrame")
}

# uncertainty propagation
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)
# toy example
dem_sample <- genSample(UMobject = demUM, n = 3, samplemethod = "ugs", nmax = 20)
slope_sample <- propagate(dem_sample, model = Slope, n = 3)
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 50, samplemethod = "ugs", nmax = 20)
slope_sample <- propagate(dem_sample, model = Slope, n = 50)

## End(Not run)

## categorical spatial data example
# load data
data(woon)

# tax model
tax <- function(building_Function) {
  building_Function$tax2pay <- NA
  building_Function$tax2pay[building_Function$Function == 1] <- 1000
  building_Function$tax2pay[building_Function$Function == 2] <- 10000
  building_Function$tax2pay[building_Function$Function == 3] <- 10
  total_tax <- sum(building_Function$tax2pay)
  total_tax
```

```
}

# uncertainty propagation
woonUM <- defineUM(TRUE, categories = c(1,2,3), cat_prob = woon[, c(4:6)])
woon_sample <- genSample(woonUM, 10)
class(woon_sample)
tax # the model takes SpatialGridDataFrame with a column called "Function"
for (i in 1:10) names(woon_sample[[i]]) <- "Function"
tax_uncert <- propagate(realizations = woon_sample, n = 10, model = tax)
tax_uncert <- unlist(tax_uncert)
summary(tax_uncert)

## cross-correlated example
# load data
data(OC, OC_sd, TN, TN_sd)

# C/N model
C_N_model_raster <- function(OC, TN) {
  OC/TN
}

# define marginal UMs
OC_crm <- makeCRM(acf0 = 0.6, range = 1000, model = "Sph")
OC_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(OC, OC_sd), crm = OC_crm, id = "OC")
TN_crm <- makeCRM(acf0 = 0.4, range = 1000, model = "Sph")
TN_UM <- defineUM(TRUE, distribution = "norm", distr_param = c(TN, TN_sd), crm = TN_crm, id = "TN")

# define joint UM
mySpatialMUM <- defineMUM(list(OC_UM, TN_UM), matrix(c(1,0.7,0.7,1), nrow=2, ncol=2))

# sample - "ugs" method
# toy example
my_cross_sample <- genSample(mySpatialMUM, n = 3, "ugs", nmax = 24)
class(my_cross_sample)
# run propagation
CN_sample <- propagate(realizations = my_cross_sample, model = C_N_model_raster, n = 3)
CN_sample
## Not run:
my_cross_sample <- genSample(mySpatialMUM, 50, "ugs", nmax = 24)
class(my_cross_sample)
# run propagation
CN_sample <- propagate(realizations = my_cross_sample, model = C_N_model_raster, n = 50)
CN_sample

## End(Not run)
```

---

quantile_MC_sgdf *quantile() function for MC sample saved in a SpatialGridDataFrame*

Description

Calculates mean from MC realizations for each location in a map.

Usage

```
quantile_MC_sgdf(realizations, ...)
```

Arguments

realizations	MC sample saved in SpatialGridDataFrame.
...	additional parameters.

Value

SpatialGridDataFrame; quantiles of a MC sample

Author(s)

Kasia Sawicka

Examples

```
set.seed(12345)
data(dem30m, dem30m_sd)
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 50, samplemethod = "ugs",
                        nmax = 20, asList = FALSE)
dem_quantile <- quantile_MC_sgdf(dem_sample, probs = c(0.1, 0.9))

## End(Not run)
```

---

render *Rendering template*

Description

Rendering is the process of replacing the tags in moustaches by text. For this, we provide a set of render-methods. See the ‘whisker‘ package (or https://mustache.github.io) for more information.

Usage

```
render(x, ...)
```

Arguments

x	an object of class "character" or "template".
...	additional parameters.

Value

Rendered character template or a file on disk.

Author(s)

Dennis Walvoort

Examples

```
require(magrittr)
require(whisker)
# render character string
my_template <- "Hello {{name}}. How are you doing?"
my_template %>%
  render(name = "Winnie the Pooh")

# render table
my_template <- c(
     "| x | y |",
     "|---|---|",
     "{{#MY_TABLE}}",
     "| {{X}} | {{Y}} |",
     "{{/MY_TABLE}}")
my_table <- data.frame(X = 1:5, Y = letters[1:5])
my_table
my_template %>%
render(MY_TABLE = unname(rowSplit(my_table))) %>%
cat
```

---

render.character *Render method for "character" class.*

Description

Rendering is the process of replacing the tags in moustaches by text.

Usage

```
## S3 method for class 'character'
render(x, ...)
```

Arguments

x	an object of class "character".
...	additional parameters.

Value

Rendered character template.

Author(s)

Dennis Walvoort

Examples

```
require(magrittr)
require(whisker)
# render character string
my_template <- "Hello {{name}}. How are you doing?"
my_template %>%
  render(name = "Winnie the Pooh")

# render table
my_template <- c(
     "| x | y |",
     "|---|---|",
     "{{#MY_TABLE}}",
     "| {{X}} | {{Y}} |",
     "{{/MY_TABLE}}")
my_table <- data.frame(X = 1:5, Y = letters[1:5])
my_table
my_template %>%
render(MY_TABLE = unname(rowSplit(my_table))) %>%
cat
```

---

render.template *Render method for "template" class.*

Description

Rendering is the process of replacing the tags in moustaches by text.

Usage

```
## S3 method for class 'template'
render(x, ...)
```

Arguments

x	an object of class "template", a model input file with additional extension ".template".
...	additional parameters.

Value

Rendered template file.

Author(s)

Dennis Walvoort

---

sd_MC_sgdf *sd() function for MC sample saved in a SpatialGridDataFrame*

Description

Calculates sd from MC realizations for each location in a map.

Usage

```
sd_MC_sgdf(realizations, ...)
```

Arguments

realizations	MC sample saved in a SpatialGridDataFrame.
...	additional parameters.

Value

SpatialGridDataFrame; a sd of a MC sample.

Author(s)

Kasia Sawicka

Examples

```
set.seed(12345)
data(dem30m, dem30m_sd)
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 50, samplemethod = "ugs",
                        nmax = 20, asList = FALSE)
```

```
dem_sample_sd <- sd_MC_sgdf(dem_sample)

## End(Not run)
```

---

spup--pkg *spup - Package for spatial uncertainty propagation*

Description

Facilitates uncertainty propagation analysis using Monte Carlo methods. In particular, provides functions that allow to do uncertainty analysis with spatial variables/models.

Author(s)

Kasia Sawicka

---

stratsamp *Stratified sampling for spatial variables*

Description

Stratified sampling for spatial variables

Usage

```
stratsamp(n, distribution, parameters, p)
```

Arguments

n	sample size per stratum.
distribution	a string, distribution type to sample from.
parameters	given distribution parameters.
p	a vector of quantiles.

Value

Sample of spatial variable. Matrix with n rows and length(p)-1 columns.

Author(s)

Stefan van Dam, Kasia Sawicka

template *Constructor for class "template".*

Description

Class that stores all templates with model inputs. The aim of this class is to: 1. organise model input files; 2. perform some checks.

Usage

```
template(filenames)
```

Arguments

filenames	a string, a name of the model input file.

Details

A template is a model input file with: 1. the additional extension '.template'. 2. input that needs to be modified is replaced by mustache-style tags.

Value

An object of a class "template".

Author(s)

Dennis Walvoort

TN *Soil total nitrogen content in a south area (33 x 33km) of lake Alaotra in Madagascar.*

Description

A dataset containing the mean of soil TN content from 0-30 cm layer.

Usage

```
data(TN)
```

Format

a RasterLayer with dimensions : 134, 135, 18090 (nrow, ncol, ncell), resolution : 250, 250 (x, y).

Source

ISRIC soilgrid information. HENGL, T., MENDES DE JESUS, J., HEUVELINK, G. B. M., RUIPEREZ GONZALEZ, M., KILIBARDA, M., BLAGOTIC, A., SHANGGUAN, W., WRIGHT, M. N., GENG, X., BAUER-MARSCHALLINGER, B., GUEVARA, M. A., VARGAS, R., MACMILLAN, R. A., BATJES, N. H., LEENAARS, J. G. B., RIBEIRO, E., WHEELER, I., MANTEL, S. & KEMPEN, B. 2017. SoilGrids250m: Global gridded soil information based on machine learning. PLOS ONE, 12, e0169748.

TN_sd *Standard deviation of soil total nitrogen content in a south area (33 x 33km) of lake Alaotra in Madagascar.*

Description

A dataset containing the standard deviation of soil TN content from 0-30 cm layer.

Usage

```
data(TN_sd)
```

Format

a RasterLayer with dimensions : 134, 135, 18090 (nrow, ncol, ncell), resolution : 250, 250 (x, y).

Source

ISRIC soilgrid information. HENGL, T., MENDES DE JESUS, J., HEUVELINK, G. B. M., RUIPEREZ GONZALEZ, M., KILIBARDA, M., BLAGOTIC, A., SHANGGUAN, W., WRIGHT, M. N., GENG, X., BAUER-MARSCHALLINGER, B., GUEVARA, M. A., VARGAS, R., MACMILLAN, R. A., BATJES, N. H., LEENAARS, J. G. B., RIBEIRO, E., WHEELER, I., MANTEL, S. & KEMPEN, B. 2017. SoilGrids250m: Global gridded soil information based on machine learning. PLOS ONE, 12, e0169748.

---

varcov *Calculate variance covariance matrix*

Description

Calculate variance covariance matrix

Usage

```
varcov(sd_vector, cormat)
```

Arguments

sd_vector	vector of standard deviations.
cormat	correlation matrix.

Value

Variance-covariance matrix.

Author(s)

Kasia Sawicka

Examples

```
vc <- varcov(c(1,2,3), matrix(c(1,0.7,0.2,0.7,1,0.5,0.2,0.5,1), nrow = 3, ncol = 3))
vc
```

---

var_MC_sgdf *var() function for MC sample saved in a SpatialGridDataFrame*

Description

Calculates var from MC realizations for each location in a map.

Usage

```
var_MC_sgdf(realizations, ...)
```

Arguments

realizations	MC sample saved in SpatialGridDataFrame.
...	additional parameters.

Value

SpatialGridDataFrame; a variance of a MC sample.

Author(s)

Kasia Sawicka

Examples

```
set.seed(12345)
data(dem30m, dem30m_sd)
dem_crm <- makeCRM(acf0 = 0.78, range = 321, model = "Exp")
demUM <- defineUM(uncertain = TRUE, distribution = "norm",
                  distr_param = c(dem30m, dem30m_sd), crm = dem_crm)
## Not run:
dem_sample <- genSample(UMobject = demUM, n = 50, samplemethod = "ugs",
                        nmax = 20, asList = FALSE)
dem_var <- var_MC_sgdf(dem_sample)

## End(Not run)
```

---

vgm2crm *Convert vgm to crm*

Description

Convert vgm to crm

Usage

```
vgm2crm(vgm, psill, nugget, range, model, kappa = 0.5, Err = 0)
```

Arguments

vgm	See ?vgm
psill	See ?vgm
nugget	See ?vgm
range	See ?vgm
model	See ?vgm
kappa	See ?vgm
Err	See ?vgm

Value

Spatial correlogram model - standardised parameters of spatial variogram model

Author(s)

Kasia Sawicka

woon *Neighbourhood in Rotterdam.*

Description

The 'woon' object is a SpatialPolygonDataFrame where each building is represented by one polygon.

Usage

```
data(woon)
```

Format

a SpatialPolygonDataFrame with 723 polygons and 7 variables:

vbos number of addresses present in the building

woonareash residential area, in percent

Function assigned category depending on vbos and woonareash - for residential is 1, for offcie is 2, for other is 3

residential probability that the building is residential

office probability that the building is an office

other probability that the building has other function

check check if probabilities sum to 1

Source

Kadaster, NL.

Index