

# Package ‘intamap’

January 2, 2012

**Version** 1.3-11

**Date** 2011-06-10

**Title** procedures for automated interpolation

**Author** Edzer Pebesma <edzer.pebesma@uni-muenster.de>, Jon Skoien  
<j.skoien@geo.uu.nl> and others

**Maintainer** Edzer J. Pebesma <edzer.pebesma@uni-muenster.de>

**Depends** R (>= 2.5.0), sp (>= 0.9-0), gstat (>= 0.9-36), rgdal (>= 0.5.2), akima, automap, mvtnorm, MASS, evd, lattice

**Suggests** doSNOW

**Enhances** psgp

**Description** A package that provides classes and methods for automated spatial interpolation.

**License** GPL (>= 2)

**URL** <http://www.intamap.org/>

**Repository** CRAN

**Date/Publication** 2011-06-15 18:02:24

## R topics documented:

intamap-package . . . . .	2
bayesCopula . . . . .	6
blockPredict . . . . .	7
checkSetup . . . . .	9
coarsenGrid . . . . .	10
conformProjections . . . . .	11
copulaEstimation . . . . .	13
createIntamapObject . . . . .	15
estimateAnisotropy . . . . .	18

estimateParameters . . . . .	21
estimateTimeModel . . . . .	23
generateTimeModels . . . . .	24
getIntamapParams . . . . .	25
getInterpolationMethodNames . . . . .	27
intamap . . . . .	28
intamapExampleObject . . . . .	28
interpolate . . . . .	29
methodParameters . . . . .	32
plotIntamap . . . . .	33
postProcess . . . . .	35
predictTime . . . . .	36
preProcess . . . . .	37
rotateAnisotropicData . . . . .	38
spatialPredict . . . . .	39
summaryIntamap . . . . .	41
timeModels . . . . .	42
unbiasedKrige . . . . .	43
yamamotoKrige . . . . .	45

<b>Index</b>	<b>47</b>
--------------	-----------

---

intamap-package	<i>A package providing methods for automatic interpolation: pre-processing, parameter estimation, spatial prediction and post processing</i>
-----------------	--

---

## Description

This package provides S3 methods for the R processing to be done in the INTAMAP project (<http://www.intamap.org>). In addition to the methods available through a web-based interface, the package includes several other options for automatic interpolation and pre- and post-processing of observations from monitoring networks.

## General setup

The normal work flow for working with the intamap package can best be illustrated with the following R-script. The procedure starts with reading data and meta data, then setting up an object which is used in the following functions: preprocess data, estimate parameters, compute spatial predictions, and post process them (i.e., write them out):

```
library(intamap)

# set up intamap object, either manually:
obj = list(
  observations = readOGR("PG:dbname=postgis", "eurdep.data"),
  predictionLocations = readOGR("PG:dbname=postgis", "eurdep1km.grid"),
  targetCRS = "+init=epsg:3035",
```

```

        params = getIntamapParams()
    )
    class(obj) = c("idw")

# or using createIntamapObject
obj = createIntamapObject(
    observations = readOGR("PG:dbname=postgis", "eurdep.data"),
    predictionLocations = readOGR("PG:dbname=postgis", "eurdep1km.grid"),
    targetCRS = "+init=epsg:3035", class = c("idw")
)

# run test:
checkSetup(obj)

# do interpolation steps:
obj = preProcess(obj)
obj = estimateParameters(obj) # faster
obj = spatialPredict(obj)
obj = postProcess(obj)

```

Our idea is that a script following this setup will allow the full statistical analysis required for the R back-end to the automatic interpolation service, and provides the means to extend the current (over-simplistic) code with the full-grown statistical analysis routines developed by INTAMAP partners. Running the package independently under R gives the user more flexibility in the utilization than what is possible through the web-interface.

Let us look into detail what the code parts do:

```
library(intamap)
```

The command `library(intamap)` loads the R code of the `intamap` package to the current R session, along with the packages required for this (`sp`, `rgdal`, `gstat`, `akima`, `automap`, `mvtnorm`, `evd`, `MASS`). All packages need to be available to the R session, which is possible after downloading them from the Comprehensive R Network Archives (CRAN) (<http://cran.r-project.org>)

```

# set up intamap object:
obj = createIntamapObject(
    observations = readOGR("PG:dbname=postgis", "eurdep.data"),
    predictionLocations = readOGR("PG:dbname=postgis", "inspire1km.grid"),
    targetCRS = "+init=epsg:3051",
    class = "idw"
)

```

This code sets up a list object called `obj`, and assigns a class (or a group of classes) to it. This list should hold anything we need in the next steps, and the bare minimum seems to be measured point data (which will be extended to polygon data) and prediction locations, and a suggestion what

to do with it. Here, the data are read from a PostGIS data base running on localhost; data base connections over a network are equally simple to set up. From the data base postgis the tables eurdep.data and inspire1km.grid are read; it is assumed that these have their SRID (spatial reference identifier) set.

The suggestion what to do with these data is put in the classes, idw. This will determine which *versions* of preProcess, parameterEstimate etc will be used: intamap provides *methods* for each of the *generic* functions preProcess, estimateParameters, spatialPredict, postProcess. Although it would be possible to apply two classes in this case (dataType in addition to idw), as the choice of pre- and post-processing steps tend to be data-dependent, we have tried to limit the number of classes to one for most applications.

The S3 method mechanism (used here) hence requires these versions to be called preProcess.idw, estimateParameters.idw, spatialPredict.idw, and postProcess.idw (and eventually also preProcess.eurdep and preProcess.eurdep).

To see that, we get in an interactive session:

```
> library(intamap)
Loading required package: sp
Loading required package: gstat
Loading required package: rgdal
Geospatial Data Abstraction Library extensions to R successfully loaded
> methods(estimateParameters)
[1] estimateParameters.automap*      estimateParameters.copula*
[3] estimateParameters.default*      estimateParameters.idw*
[5] estimateParameters.linearVariogram* estimateParameters.transGaussian*
[7] estimateParameters.yamamoto*
```

Now if a partner provides additional methods for BayesianKriging, one could integrate them by

```
class(obj) = "BayesianKriging"
```

and provide some or all of the functions preProcess.BayesianKriging, estimateParameters.BayesianKriging, spatialPredict.BayesianKriging, and postProcess.BayesianKriging, which would be called automatically when using their generic form (preProcess etc).

It is also possible to provide a method that calls another method. Further, for each generic there is a default method. For estimateParameter and spatialPredict these print an error message and stop, for the pre- and postprocessing the default methods may be the only thing needed for the full procedure; if no preProcess.BayesianKriging is found, preProcess.default will be used when the generic (preProcess) is called.

If a method does something, then it adds its result to the object it received, and returns this object. If it doesn't do anything, then it just passes (returns) the object it received.

To make these different methods exchangeable, it is needed that they can all make the same assumptions about the contents of the object that they receive when called, and that what they return complies with what the consequent procedures expect. The details about that are given in the descriptions of the respective methods, below.

Because a specific interpolation method implemented may have its peculiar characteristics, it may have to extend these prescriptions by passing more information than described below, for example information about priors from estimateParameters to spatialPredict.

The choice between methods is usually done based on the type of problem (extreme values present, computation time available etc.). The possibility for parallel processing of the prediction step is enabled for some of the main methods. To be able to take advantage of multiple CPUs on a computer, the package doSNOW must be downloaded, additionally the parameter nclus must be set to a value larger than 1.

### Input object components

`observations` object of class `SpatialPointsDataFrame`, containing a field value that is the target variable.

`predictionLocations` object extending class `Spatial`, containing prediction locations.

`targetCRS` character; target CRS or missing

`formulaString` formula string for parameter estimation and prediction functions

`params` list of parameters, to be set in `getIntamapParams`. These parameters include:

**doAnisotropy = FALSE** Defining whether anisotropy should be calculated

**removeBias = NA** Defining whether biases should be removed, and in case yes, which ones (localBias and regionalBias implemented)

**addBias = NA** Defining which biases to be added in the `postProcess` function. This has not yet been implemented.

**biasRemovalMethod = "LM"** character; specifies which methods to use to remove bias. See below.

**doSegmentation = FALSE** Defining if the predictions should be subject to segmentation. Segmentation has been implemented, but not the use of it.

**nmax = 50** for local kriging: the number of nearest observations that should be used for a kriging prediction or simulation, where nearest is defined in terms of the space of the spatial locations. By default, 50 observations are used.

**ngrid = 100** The number of grid points to be used if an Averaged Cumulative Distribution Function (ACDF) needs to be computed for unbiased kriging

**nsim=100** Number of simulations when needed

**block = numeric(0)** Block size; a vector with 1, 2 or 3 values containing the size of a rectangular in x-, y- and z-dimension respectively (0 if not set), or a data frame with 1, 2 or 3 columns, containing the points that discretize the block in the x-, y- and z-dimension to define irregular blocks relative to (0,0) or (0,0,0) - see also the details section of `predict.gstat`. By default, predictions or simulations refer to the support of the data values.

**processType = "gaussian"** If known - the distribution of the data. Defaults to gaussian, analytical solutions also exists in some cases for logNormal. This setting only affects a limited number of methods, e.g. the block prediction

**confProj = FALSE** If set, the program will attempt conform projections in `preProcess`, calling the function `conformProjections`.

**nclus = 1** The number of clusters to use, if applying to a method which can run processes in parallel. Currently implemented for methods automap, copula and psgp.

**debug.level = 0** Used in some functions for giving additional output. See individual functions for more information.

... Additional parameters that do not exist in the default parameter set, particularly parameters necessary for new methods within the intamap package

---

bayesCopula	<i>Performs spatial interpolation using copulas</i>
-------------	---

---

### Description

Calculates predictive mean, predictive variance, predictive quantiles and exceedance probabilities for certain thresholds in the spatial copula model.

### Usage

```
bayesCopula(obj, estimates, search=10, calc=list(mean=TRUE, variance=TRUE), testMean=FALSE)
```

### Arguments

obj	Intamap object including observations and predictionLocations, see <a href="#">intamap-package</a>
estimates	List of estimated parameters (typically obtained by calling <a href="#">copulaEstimation</a> )
search	local prediction: number of observed locations considered for prediction at each unknown point
calc	list of what prediction type is required: <ul style="list-style-type: none"> <li>• mean = TRUE if the predictive mean should be calculated, FALSE otherwise</li> <li>• variance = TRUE if the predictive variance should be calculated, FALSE otherwise</li> <li>• quantiles = NULL Vector of desired predictive quantiles, e.g. 0.95 or 0.05</li> <li>• excprob = NULL Vector of thresholds, where the probability of exceeding this threshold is desired</li> </ul>
testMean	Whether or not the predictive means (if calculated) should be tested for being reasonable.

### Details

bayesCopula is used for plug-in prediction at unobserved spatial locations. The name of the function is somewhat misleading since no Bayesian approach is implemented so far. It is possible to calculate numerically the predictive mean and variance for both the Gaussian and the chi-square spatial copula model. Exceedance probabilities and predictive quantiles are only supported for the Gaussian copula model. Note that it may occur that the predictive distribution has no finite moments. In this case, a possible predictor is the median of the predictive distribution. If testMean=TRUE and the predictive means have no reasonable values, the median is automatically calculated and a warning is produced.

The copula prediction method is computationally demanding. There is a possibility of running it as a parallel process by setting the parameter nclus > 1 for the interpolation process. This requires a previous installation of the package doSNOW.

**Value**

List with the following elements:

mean	Mean of the predictive distribution. NULL if not calculated.
variance	Variance of the predictive distribution. NULL if not calculated.
quantiles	Quantiles of the predictive distribution NULL if not calculated.
excprob	Probabilities for the predictive distribution to exceed predefined thresholds. NULL if not calculated.

**Author(s)**

Hannes Kazianka

**References**

Kazianka, H. and Pilz, J. (2009), Spatial Interpolation Using Copula-Based Geostatistical Models. GeoENV2008 - Geostatistics for Environmental Application (P. Atkinson, C. Lloyd, eds.), Springer, New York

**See Also**

[copulaEstimation](#), [spatialPredict](#), [estimateParameters](#)

**Examples**

```
## Not run:
data(intamapExampleObject)
## estimate parameters for the copula model
copula<-list(method="norm")
anisotropy<-list(lower=c(0,1),upper=c(pi,Inf),params=c(pi/3,2))
correlation<-list(model="Ste",lower=c(0.01,0.01,0.01),upper=c(0.99,Inf,20),params=c(0.05,4,3))
margin<-list(name="gev",lower=c(0.01,-Inf),upper=c(Inf,Inf),params=c(30,0.5))
trend<-list(F=as.matrix(rep(1,196)),lower=-Inf,upper=Inf,params=40)
estimates<-copulaEstimation(intamapExampleObject,margin,trend,correlation,anisotropy,copula)
## make predictions at unobserved locations
predictions<-bayesCopula(intamapExampleObject,estimates,search=25,calc=list(mean=TRUE,variance=TRUE,excprob=40)

## End(Not run)
```

---

blockPredict

*Spatial block prediction*

---

**Description**

blockPredict is a generic method for prediction of spatially aggregated variables within the [intamap-package](#) package.

**Usage**

```
blockPredict(object, ...)
```

**Arguments**

**object** a list object of the type described in [intamap-package](#)

**...** other arguments that will be passed to the requested interpolation method. See the individual interpolation methods for more information. The following arguments from object can be overrun through ...:

**block** Block size; a vector with 1, 2 or 3 values containing the size of a rectangular in x-, y- and z-dimension respectively (0 if not set), or a data frame with 1, 2 or 3 columns, containing the points that discretize the block in the x-, y- and z-dimension to define irregular blocks relative to (0,0) or (0,0,0) - see also the details section of [predict.gstat](#). By default, predictions or simulations refer to the support of the data values.

**cellsize** size of cells for discretization of blocks for numerical simulation

**Details**

The function `blockPredict` is a wrapper around the `spatialPredict.block` function within the [intamap-package](#) package, to simplify the calls for block predictions.

Block predictions are spatial predictions assumed to be valid for a certain area. The blocks can either be given by passing [SpatialPolygons](#) as the `predictonLocations` or by passing the `block`-argument through the parameters of the object or through the `...`-argument.

There are essentially two ways to solve the problems of block predictions.

- analyticalblock predictions can be found directly by block kriging
- numericalblock predictions can be found through numerical simulations over a set of points within the block, the requested output is found by averaging over these simulations

The analytical solutions are used when applicable. This is typically for ordinary kriging based methods and prediction types that can be found by linear aggregation (e.g. block mean).

If the prediction type necessitates simulations, this is done by subsampling the blocks. This can either be done block-wise, with a certain number of points within each block, with a certain cellsize, or with a certain number of points

`automap` Uses function [autoKrige](#) in the `automap` package. If object already includes a variogram model, [krige](#) in the `gstat`-package will be called directly.

**Value**

a list object similar to `object`, but extended with predictions at a the set of locations defined `object`.

**Author(s)**

Jon Olav Skoien

**References**

<http://www.intamap.org/>

**See Also**

[gstat,autoKrige](#)

**Examples**

```
# This example skips some steps that might be necessary for more complicated
# tasks, such as estimateParameters and pre- and postProcessing of the data
data(meuse)
coordinates(meuse) = ~x+y
meuse$value = log(meuse$zinc)
data(meuse.grid)
gridded(meuse.grid) = ~x+y
proj4string(meuse) = CRS("+init=epsg:28992")
proj4string(meuse.grid) = CRS("+init=epsg:28992")

# set up intamap object:
obj = createIntamapObject(
  observations = meuse,
  predictionLocations = meuse.grid,
  targetCRS = "+init=epsg:3035",
  class = "automap"
)

# do interpolation step:
obj = conformProjections(obj)
obj = estimateParameters(obj) # blockPredict
obj = blockPredict(obj,block=c(100,100)) # blockPredict
```

---

checkSetup

*check setup*

---

**Description**

checkSetup will do some sanity checks on input data provided through object.

**Usage**

```
checkSetup(object, quiet = FALSE)
```

**Arguments**

object	object, to be passed to <a href="#">preProcess</a> , see <a href="#">intamap-package</a>
quiet	logical; TRUE to suppress OK statement

## Details

checkSetup is a function that makes certain tests on the intamap object to make sure that it is suited for interpolation. Particularly, it will issue a warning or an error if one of the following conditions are met:

- observations is not an element of object
- observations contain less than 20 observations
- Some of the observation locations are duplicated
- formulaString is not an element of object
- None of the columns of observations has a name that corresponds to the independent variable of formulaString
- predictionLocations is not an element of object
- predictionLocations is not a [Spatial](#) object
- targetCRS is given, but observations and predictionLocations do not have CRS set
- addBias includes biases that are not part of removeBias

The function will issue a warning if it appears that predictionLocations and observations share a small region. This warning is given as it is a likely cause of errors, although it can also happen if predictionLocations are limited to one small cluster.

## Value

returns TRUE if check passes, will halt with error when some error condition is met.

## Author(s)

Edzer J. Pebesma

## References

<http://www.intamap.org/>

---

coarsenGrid

*Coarsening of a spatial grid*

---

## Description

coarsenGrid is a function that resamples a SpatialGridDataFrame.

## Usage

```
coarsenGrid(object,coarse=2,offset = sample(c(0:(coarse-1)),2,replace=TRUE))
```

**Arguments**

object	a <a href="#">SpatialGridDataFrame</a> or gridded <a href="#">SpatialPixelsDataFrame</a>
coarse	an integer telling how much the grid should be coarsened
offset	integer giving the relative offset of the first point, see details below for a closer description

**Details**

The function `coarsenGrid` is a function that samples from a [SpatialGridDataFrame](#). The argument `coarse` indicates that every `coarse` row and column will be sampled, starting with the row and column represented in `offset`. `offset = c(0,0)` implies that the smallest x- and y-coordinates will be a part of the resampled data set, `offset = c(1,1)` implies that sampling will start on the second row and column.

**Value**

a [SpatialGridDataFrame](#).

**Author(s)**

Jon Olav Skoien

**References**

<http://www.intamap.org/>

**Examples**

```
data(meuse.grid)
gridded(meuse.grid) = ~x+y
newMeuseGrid = coarsenGrid(meuse.grid,coarse=2,offset=c(1,1))
```

---

conformProjections      *Getting conformed projections*

---

**Description**

Getting a conformed projection for a set of [Spatial](#)\* elements necessary for interpolation in the [intamap-package](#).

**Usage**

```
conformProjections(object)
```

**Arguments**

**object** an object of the type described in [intamap-package](#)

**Details**

conformProjections is a function that attempts to reproject all projected elements in object to one common projection. The function is usually called with an intamap object as argument from [createIntamapObject](#) if the parameter confProj = TRUE. Thus it is a function that is usually not necessary to call separately.

The need for this function is because several of the functions in a typical spatial interpolation work flow inside the [intamap-package](#) require that the elements have a common projection. In addition, there are some functions which are not able to deal with unprojected spatial objects, i.e. objects with coordinates given in latitude and longitude. conformProjections will hence also attempt to reproject all elements that have coordinates in latitude and longitude, even in the cases where they all have the same projections.

If only one of observations or predictionLocations has a projection (or is longlat), the other one is assumed to be equal. A warning is issued in this case.

The common projection depends on the object that is passed to conformProjections. First of all, if intCRS (see below) is present as an element of the object, all elements will be reprojected to this projection. If not, intCRS will be set equal to the first projection possible in the list below.

**intCRS** Can be given as a component in object - and is the user-defined common projection used for interpolation

**targetCRS** Can be given as a component in object - and is the user-defined target projections

**predCRS** The projection of the predictionLocations in object

**obsCRS** The projection of the observations

**Value**

A list of the parameters to be included in the object described in [intamap-package](#)

**Author(s)**

Jon Olav Skoien

**References**

<http://www.intamap.org/>

**Examples**

```
library(intamap)

data(meuse)
coordinates(meuse) = ~x+y
proj4string(meuse) <- CRS("+proj=stere +lat_0=52.15616055555555 +lon_0=5.387638888888889 +k=0.999908 +x_0=155000 +y_0=463000")
```

```

predictionLocations = spsample(meuse, 50, "regular")

krigingObject = createIntamapObject(
  observations = meuse,
  predictionLocations = predictionLocations,
  formulaString = as.formula("log(zinc)~1"),
  intCRS = "+init=epsg:3035"
)

krigingObject = conformProjections(krigingObject)
proj4string(meuse)
proj4string(krigingObject$observations)

```

---

copulaEstimation	<i>ML-estimation of the spatial copula model parameters</i>
------------------	---

---

### Description

Estimates parameters of the spatial copula model using maximum likelihood.

### Usage

```
copulaEstimation(obj, margin, trend, correlation, anisotropy, copula, tol=0.001, ...)
```

### Arguments

obj	Intamap object, see description in <a href="#">intamap-package</a>
margin	list with the following elements: params Starting values for the parameters of the marginal distribution (excluding trend parameters) lower Lower bounds for the values of the parameters of the marginal distribution (excluding trend parameters) upper Upper bounds for the values of the parameters of the marginal distribution (excluding trend parameters) name Name of the family of marginal distributions. Possible names are: "norm", "lnorm", "gev", "t" and "logis"
trend	list with the following elements: params Starting values for the parameters of the trend model (location parameter of the marginal distribution) lower Lower bounds for the values of the parameters of the trend model upper Upper bounds for the values of the parameters of the trend model F Design matrix.
correlation	list with the following elements: model Correlation function model. Possible models are: "Ste", "Sph", "Gau" and "Exp"

	params	Starting values for the parameters of the correlation function model
	lower	Lower bounds for the values of the parameters of the correlation function model
	upper	Upper bounds for the values of the parameters of the correlation function model
anisotropy		list with the following elements:
	params	Starting values for the parameters of geometric anisotropy. If NULL, then no anisotropy is considered.
	lower	Lower bounds for the values of the parameters of geometric anisotropy. Usually $c(0, 1)$
	upper	Upper bounds for the values of the parameters of geometric anisotropy. Usually $c(\pi, \text{Inf})$
copula		list with the following elements:
	method	Either "norm" or "chisq", depending on which spatial copula model is used, the Gaussian or the chi-squared copula.
	params	Only used in case of the chi-squared copula: the squared non-centrality parameter of the non-central chi-squared distribution. Controls how far the chi-squared copula is from the Gaussian copula.
	lower	Only used in case of the chi-squared copula: the lower bound for the copula parameter. Usually set to 0
	upper	Only used in case of the chi-squared copula: the upper bound for the copula parameter. Usually set to Inf
tol		Tolerance level for the optimization process.
...		Arguments to be passed to <code>optim</code> .

### Details

`copulaEstimation` performs maximum likelihood estimation of all possible parameters included in the Gaussian and chi-squared spatial copula model: parameters of the predefined family of marginal distributions (including spatial trend or external drift), correlation function parameters, parameters for geometric anisotropy and parameters for the copula (only used for the chi-squared copula model). Due to the large number of variables that need to be optimized, a profile-likelihood approach is used. Although convergence to a global optimum is not assured, the profile-likelihood method makes it less likely that the optimization routine, `optim`, gets stuck in a local optimum. The result of `copulaEstimation` is a list containing all parameter point estimates that are needed for plug-in spatial prediction. It is advisable to check the output of the algorithm by trying different starting values for the optimization.

### Value

A list with the following elements:

margin	Same as the input except that the list element "params" now consists of the optimized parameters of the marginal distribution function.
trend	Same as the input except that the list element "params" now consists of the optimized parameters of the trend model.

correlation	Same as the input except that the list element "params" now consists of the optimized parameters of the correlation function model.
anisotropy	Same as the input except that the list element "params" now consists of the optimized parameters of geometric anisotropy.
copula	Same as the input except that the list element "params" now consists of the optimized copula parameters.

**Author(s)**

Hannes Kazianka

**References**

Kazianka, H. and Pilz, J. (2009), Spatial Interpolation Using Copula-Based Geostatistical Models. GeoENV2008 - Geostatistics for Environmental Application (P. Atkinson, C. Lloyd, eds.), Springer, New York

**See Also**

[bayesCopula](#), [spatialPredict](#), [estimateParameters](#)

**Examples**

```
data(intamapExampleObject)
## estimate parameters for the copula model

## Not run: copula<-list(method="norm")
anisotropy<-list(lower=c(0,1),upper=c(pi,Inf),params=c(pi/3,2))
correlation<-list(model="Ste",lower=c(0.01,0.01,0.01),upper=c(0.99,Inf,20),params=c(0.05,4,3))
margin<-list(name="gev",lower=c(0.01,-Inf),upper=c(Inf,Inf),params=c(30,0.5))
trend<-list(F=as.matrix(rep(1,196)),lower=-Inf,upper=Inf,params=40)
estimates<-copulaEstimation(intamapExampleObject,margin,trend,correlation,anisotropy,copula)
## make predictions at unobserved locations
predictions<-bayesCopula(intamapExampleObject,estimates,search=25,calc=list(mean=TRUE,variance=TRUE,excprob=40)
## End(Not run)
```

---

createIntamapObject     *Create an object for interpolation within the intamap package*

---

**Description**

This is a help function for creating an object (see [intamap-package](#) to be used for interpolation within the [intamap-package](#)

**Usage**

```
createIntamapObject(observations, obsChar, formulaString,
  predictionLocations=100, targetCRS, boundaries, boundaryLines,
  intCRS, params=list(), boundFile, lineFile, class="idw",
  outputWhat, blockWhat = "none",...)
```

**Arguments**

observations	a <a href="#">SpatialPointsDataFrame</a> , <a href="#">SpatialPixelsDataFrame</a> , <a href="#">SpatialGridDataFrame</a> , <a href="#">SpatialLinesDataFrame</a> or <a href="#">SpatialPolygonsDataFrame</a> with observations. Note that there are only few methods that can actually handle interpolation of observations with a support
obsChar	list with observation characteristics, used by some interpolation methods
formulaString	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name z, for ordinary and simple kriging use the formula $z \sim 1$ ; for universal kriging, suppose z is linearly dependent on x and y, use the formula $z \sim x + y$ . The formulaString defaults to "value~1" if value is a part of the data set. If not, the first column of the data set is used.
predictionLocations	either a <a href="#">Spatial</a> * object with prediction locations or an integer with the requested number of prediction locations. If boundaries are supported, the sampled prediction locations will be sampled within the boundaries
targetCRS	the wanted projection for the interpolated map
boundaries	<a href="#">SpatialPolygonsDataFrame</a> with the boundaries of regions in the prediction region
boundaryLines	<a href="#">SpatialPointsDataFrame</a> with the boundaries between pairs of regions discretized as points. Will be read from file if lineFile is given or will be created from boundaries if not.
intCRS	a particular projection requested for the interpolation
params	parameters for the interpolation, given as exceptions to the default parameters set in the function <a href="#">getIntamapParams</a>
boundFile	Filename where boundaries can be found, e.g. a shapefile
lineFile	Filename where paired points on boundaries can be found
class	setting the class(es) of the object, see <a href="#">intamap-package</a>
outputWhat	List defining the requested type of output. Parameters: mean = TRUE Usual kriging prediction variance = TRUE Usual kriging error quantile The estimated quantile for a certain threshold excpb Exceedance probability for a certain threshold cumdistr The cumulative distribution for a certain value MOK Assumed unbiased prediction using the MOK method for the threshold given. See <a href="#">unbiasedKrige</a> IWQSEL Assumed unbiased prediction using the IWQSEL method for the threshold given. See <a href="#">unbiasedKrige</a> ... Additional prediction types that do not exist in the default parameter set, particularly parameters necessary for new methods within the <a href="#">intamap-package</a> .

The list defaults to list (mean = TRUE) for objects of class IDW and list(mean=TRUE, variance = TRUE) for all other objects.

blockWhat	List defining particular output for block predictions. These include:
blockMax	logical; whether to predict maximum within block, if block predictions
blockMin	logical; whether to predict minimum within block, if block predictions
fat	Prediction of area within block above a threshold (fat = threshold
blockMaxVar	logical; whether to predict the variance of the prediction of max within the block, similarly it is possible to set blockMinVar = TRUE and fatVar = threshold
...	<ul style="list-style-type: none"> <li>• Either: other elements that can be used by particular interpolation methods. These are added to the object as named elements.</li> <li>• Or: elements that have been created in earlier calls to one of the functions in the <a href="#">intamap-package</a>, and that are not supposed to change in the second call. By adding these elements to the object in <code>createIntamapObject</code>, they can be reused without having to re-estimate them. Typical examples are the elements created from a call to <code>preProcess</code></li> </ul>

## Details

This function is a help function for creating an object (see [intamap-package](#)) for interpolation within the [intamap-package](#). The function uses some default values if certain elements are not included.

If `createIntamapObject` is called without `predictionLocations`, or if a number is given, the function will sample a set of `predictionLocations`. These will be sampled from a regular grid.

`targetCRS` and `intCRS` are not mandatory variables, but are recommended if the user wants predictions of a certain projection. `intCRS` is not necessary if the `targetCRS` is given and has a projection (is not lat-long). It is recommended to include the argument `intCRS` if all projected elements are lat-long, as many of the interpolation methods do not work optimal with lat-long data.

The `...`-argument can be used for arguments necessary for new methods not being a part of the [intamap-package](#). It is also a method for reusing previously calculated elements that can be assumed to be unchanged for the second interpolation.

## Value

An object with observations, prediction locations, parameters and possible additional elements for automatic interpolation. The object will have class equal to the value of argument `class`, and methods in the [intamap-package](#) will dispatch on the object according to this class.

## Author(s)

Jon Olav Skoien

## References

<http://www.intamap.org/>

**See Also**

[intamap-package](#) and [getIntamapParams](#)

**Examples**

```
library(intamap)

# set up data:
data(meuse)
coordinates(meuse) = ~x+y
meuse$value = log(meuse$zinc)
data(meuse.grid)
gridded(meuse.grid) = ~x+y
proj4string(meuse) = CRS("+init=epsg:28992")
proj4string(meuse.grid) = CRS("+init=epsg:28992")

# set up intamap object:
idwObject = createIntamapObject(
  observations = meuse,
  predictionLocations = meuse.grid,
  targetCRS = "+init=epsg:3035",
  class = "idw"
)
```

---

estimateAnisotropy     *estimateAnisotropy*

---

**Description**

This function estimates geometric anisotropy parameters for 2-D scattered data using the CTI method.

**Usage**

```
estimateAnisotropy(object, depVar, formulaString)
```

**Arguments**

object	(i) An Intamap type object (see <a href="#">intamap-package</a> ) containing one <a href="#">SpatialPointsDataFrame</a> data frame named observations which includes the observed values (ii) or a <a href="#">SpatialPointsDataFrame</a> which includes both coordinates and observations.
depVar	name of the dependent variable; this is used only in case (ii).
formulaString	formula that defines the dependent variable as a linear model of independent variables, only used for case (ii); suppose the dependent variable has name z, for ordinary and simple kriging use the formula $z \sim 1$ ; for universal kriging, suppose z is linearly dependent on x and y, use the formula $z \sim x + y$ . The formulaString defaults to "value~1" if value is a part of the data set. If not, the first column of the data set is used.

## Details

Given the input object that defines  $N$  coordinate pairs  $(x,y)$  and observed values  $(z)$ , this method estimates of the geometric anisotropy parameters. Geometric anisotropy is a statistical property, which implies that the iso-level contours of the covariance function are elliptical. In this case the anisotropy is determined from the anisotropic ratio ( $R$ ) and the orientation angle ( $\theta$ ) of the ellipse.

Assuming a Cartesian coordinate system of axes  $x$  and  $y$ ,  $\theta$  represents the angle between the horizontal axis and PA1, where PA1 is one of the principal axes of the ellipse, arbitrarily selected (PA2 will denote the other axis).  $R$  represents the ratio of the correlation along PA1 divided by the correlation length PA2. Note that the returned value of  $R$  is always greater than one (see value below.)

The estimation is based on the Covariance Tensor Identity (CTI) method. In CTI, the Hessian matrix of the covariance function is estimated from sample derivatives. The anisotropy parameters are estimated by explicit solutions of nonlinear equations that link  $(R,\theta)$  with ratios of the covariance Hessian matrix elements.

To estimate the sample derivatives from scattered data, a background square lattice is used. The lattice extends in the horizontal direction from  $x.min$  to  $x.max$  where  $x.min$  ( $x.max$ ) is equal to the minimum (maximum)  $x$ -coordinate of the data, and similarly in the vertical direction. The cell step in each direction is equal to the length of the lattice to the respective direction divided by the square root of  $N$ .

BiLinear interpolation, as implemented in `akima` package, is used to interpolate the field's  $z$  values at the nodes of the lattice.

The CTI method is described in detail in (Chorti and Hristopulos, 2008).

Note that to be compatible with `gstat` the returned estimate of the anisotropy ratio is always greater than 1.

For observations assumed to have a trend, the trend is first subtracted from the data using universal kriging. This is an approximation, as the trend subtraction does not take anisotropy into account.

## Value

(i) If the input is an `Intamap` object, the value is a modification of the input object, containing a list element `anisPar` with the estimated anisotropy parameters. (ii) if the input is a `SpatialPointsDataFrame`, then only the list `anisPar` is returned. The list `anisPar` contains the following elements:

<code>ratio</code>	The estimate of the anisotropy ratio parameter. Using the degeneracy of the anisotropy under simultaneous ratio inversion and axis rotation transformations, the returned value of the ratio is always greater than 1.
<code>direction</code>	The estimate of the anisotropy orientation angle. It returns the angle between the major anisotropy axis and the horizontal axis, and its value is in the interval $(-90,90)$ degrees.
<code>Q</code>	A $3 \times 1$ array containing the sample estimates of the diagonal and off-diagonal elements ( $Q_{11}, Q_{22}, Q_{12}$ ) of the covariance Hessian matrix evaluated at zero lag.
<code>doRotation</code>	Boolean value indicating if the estimated anisotropy is statistically significant. This value is based on a statistical test of the isotropic ( $R= 1$ ) hypothesis using a non-parametric approximation for the 95 percent confidence interval for $R$ . This

approximation leads to conservative (wider than the true) estimates of the confidence interval. If `doRotation==TRUE` then an isotropy restoring transformation (rotation and rescaling) is performed on the coordinates. If `doRotation==FALSE` no action is taken.

### Note

This function uses `akima` package to perform "bilinear" interpolation. The source code also allows other interpolation methods, but this option is not available when the function is called from within INTAMAP.

In the `gstat` package, the anisotropy ratio is defined in the interval (0,1) and the orientation angle is the angle between the vertical axis and the major anisotropy axis, measured in the clockwise direction. If one wants to use ordinary kriging inside INTAMAP the necessary transformations are performed in the function `estimateParameters.automap`. If one wants to use ordinary kriging in the `gstat` package (but outside INTAMAP) the required transformations can be found in the source code of the `estimateParameters.automap` function.

### Author(s)

A.Chorti, D.T.Hristopulos,G. Spiliopoulos

### References

- [1] <http://www.intamap.org>,
- [2] A. Chorti and D. T. Hristopulos (2008). Non-parametric Identification of Anisotropic (Elliptic) Correlations in Spatially Distributed Data Sets, *IEEE Transactions on Signal Processing*, 56(10), 4738-4751 (2008).
- [3] Em.Petrakis and D. T. Hristopulos (2009). A non-parametric test of statistical isotropy for Differentiable Spatial Random Fields in Two Dimensions. Work in progress. email: [dionisi@mred.tuc.gr](mailto:dionisi@mred.tuc.gr)

### Examples

```
library(intamap)
data(sic2004)
coordinates(sic.val)=~x+y
sic.val$value=sic.val$dayx

params=NULL

estimateAnisotropy(sic.val,depVar = "joker")
```

---

estimateParameters      *Automatic estimation of correlation structure parameters*

---

## Description

Function to estimate correlation structure parameters. The actual parameters depend on the method used.

## Usage

```
estimateParameters(object, ...)
```

## Usage

```
## S3 method for class 'automap'
estimateParameters(object, ... )
## S3 method for class 'copula'
estimateParameters(object, ... )
## Default S3 method:
estimateParameters(object, ...)
## S3 method for class 'idw'
estimateParameters(object, ... , idpRange = seq(0.1, 2.9, 0.1), n folds = 5)
## S3 method for class 'linearVariogram'
estimateParameters(object, ...)

## S3 method for class 'transGaussian'
estimateParameters(object, lambda, significant = TRUE, ... )
## S3 method for class 'yamamoto'
estimateParameters(object, ... )
```

## Arguments

object	an intamap object of the type described in <a href="#">intamap-package</a>
...	other arguments that will be passed to the requested interpolation method. See the individual methods for more information
idpRange	range of idp (inverse distance weighting power) values over which to optimize mse
n folds	number of folds in n-fold cross validation
lambda	lambda parameter for <a href="#">boxcox</a> -transformation
significant	logical; if TRUE only transform if any of the four tests described under <a href="#">interpolate</a> are TRUE.

## Details

The function `estimateParameters` is a wrapper around different methods for estimating correlation parameters to be used for the spatial prediction method `spatialPredict`. Below are some details about and/or links to the different methods currently implemented in the `intamap-package`.

`automap` It is possible but not necessary to estimate variogram parameters for this method. If `estimateParameters` is called with an object of class `automap`, `autofitVariogram` will be called. If object already includes a variogram model when `spatialPredict` is called, `krige` in the `gstat`-package will be called directly.

`copula` finding the best copula parameters using `copulaEstimation`

`default` a default method is not really implemented, this function is only created to give a sensible error message if the function is called with an object for which no method exist

`idw` fits the best possible `idw`-power to the data set by brute force searching within the `idpRange`

`linearVariogram` this function just returns the original data, no parameter fitting is necessary for linear variogram kriging

`transGaussian` Finding the best model parameters for `transGaussian` kriging (`krigeTg`). This means finding the best `lambda` for the `boxcox`-transformation and the fitted variogram parameters for the transformed variable. If `significant = TRUE` will `lambda` only be estimated if the data show some deviation from normality, i.e., that at least one of the tests described under `interpolate` is `TRUE`. Note that `transGaussian` kriging is only possible for data with strictly positive values.

`yamamoto` a wrapper around `estimateParameters.automap`, only to assure that there is a method also for this class, difference to `automap` is more important in `spatialPredict`

It is also possible to add to the above methods with functionality from other packages, if wanted. See description on <http://www.intamap.org/newMethods.php> You can also check which methods are available from other packages by calling

```
>methods(estimateParameters)
```

## Value

a list object similar to `object`, but extended with correlation parameters.

## Author(s)

Jon Olav Skoien

## References

<http://www.intamap.org/>

## See Also

`createIntamapObject`, `spatialPredict`, `intamap-package`

**Examples**

```

library(intamap)

set.seed(13131)

# set up data:
data(meuse)
coordinates(meuse) = ~x+y
meuse$value = log(meuse$zinc)
data(meuse.grid)
gridded(meuse.grid) = ~x+y
proj4string(meuse) = CRS("+init=epsg:28992")
proj4string(meuse.grid) = CRS("+init=epsg:28992")

# set up intamap object:
idwObject = createIntamapObject(
  observations = meuse,
  formulaString=as.formula(zinc~1),
  predictionLocations = meuse.grid,
  class = "idw"
)

# run test:
checkSetup(idwObject)

# do interpolation steps:
idwObject = estimateParameters(idwObject, idpRange = seq(0.25,2.75,.25),
                              nfold=3) # faster
idwObject$inverseDistancePower

```

---

```
estimateTimeModel      estimateTimeModel
```

---

**Description**

Function that takes time samples function that can read intamap objects

**Usage**

```
estimateTimeModel(FUN, class, formulaString, debug.level, ...)
```

**Arguments**

<code>FUN</code>	A string with function's name
<code>class</code>	class of intamapObject, which interpolation method to be used.
<code>formulaString</code>	the formula of the request, mainly to see if the request has independent variables.
<code>debug.level</code>	if <code>debug.level &gt;= 1</code> , the function will store tables with the prediction times for each model in the workspace.
<code>...</code>	other arguments as defined in the <a href="#">createIntamapObject</a> function

**Details**

This function uses `createIntamapObject` function to create synthetic data, in order to take time samples for the function with string name "FUN". The calculated model is stored, as an element of a list, in a local file (workspace for now) and it's used in order to give quick time estimates.

**Value**

The function does not return a variable but stores the result in an element list with the same name.

---

<code>generateTimeModels</code>	<i>Generate time models</i>
---------------------------------	-----------------------------

---

**Description**

function that generates time models and saves them in workspace.

**Usage**

```
generateTimeModels(genClasses = NULL, noGenClasses = NULL, nSam = 1, test = FALSE,
                  debug.level = 0)
```

**Arguments**

<code>genClasses</code>	list of particular classes for which time models should be generated
<code>noGenClasses</code>	list of particular classes for which time models should not be generated
<code>nSam</code>	number of attempts to be tried for each combination of predictions and observations, defaults to 1, higher number should be used for better accuracy. <code>nSam/2</code> is used for copulas, to reduce computation time.
<code>test</code>	logical; if true, the time models are generated based on fewer iterations, for speed
<code>debug.level</code>	if <code>debug.level &gt;= 1</code> , the function will store tables with the prediction times for each model in the workspace.

**Details**

This function calculates a time model for different interpolation types in the [intamap-package](#) and returns a list object with the estimated models. It's users responsibility to store the model in the workspace. The normal procedure would be to run the function without arguments. However, it is both possible to define a list for which classes the user want to generate models, or a list of classes that are not of interest.

The time model is based on creation of a set of synthetical data sets of different size, both regarding number of observations and prediction locations. The function will estimate parameters and make predictions with the different combinations, and for each method, fit a local polynomial regression model ([loess](#))

This model can then be used by [predictTime](#) to estimate the prediction time for an interpolation request with a certain number of observations and prediction locations.

**Value**

The function generates a `timeModels` object, which can be used to estimate prediction times for different requests to the `interpolate` function in the `intamap-package`, via `predictTime`.

**Examples**

```
## Not run:
timeModels=generateTimeModels()
q("yes")
## restart R in the same directory

## End(Not run)
```

---

getIntamapParams

*Setting parameters for the intamap package*


---

**Description**

This function sets a range of the parameters for the `intamap-package`, to be included in the object described in `intamap-package`

**Usage**

```
getIntamapParams(oldPar, newPar, ...)
```

**Arguments**

<code>oldPar</code>	An existing set of parameters for the interpolation process, of class <code>IntamapParams</code> or a list of parameters for modification of the default parameters
<code>newPar</code>	A list of parameters for updating <code>oldPar</code> or for modification of the default parameters. Possible parameters with their defaults are given below
<code>...</code>	Individual parameters for updating <code>oldPar</code> or for modification of the default parameters. Possible parameters with their defaults are given below
	<code>doAnisotropy = FALSE</code> Defining whether anisotropy should be calculated
	<code>removeBias = NA</code> Defining whether biases should be removed, and in case yes, which ones ( <code>localBias</code> and <code>regionalBias</code> implemented)
	<code>addBias = NA</code> Defining which biases to be added in the <code>postProcess</code> function. This has not yet been implemented.
	<code>biasRemovalMethod = "LM"</code> character; specifies which methods to use to remove bias. See below.
	<code>doSegmentation = FALSE</code> Defining if the predictions should be subject to segmentation. Segmentation has been implemented, but not the use of it.
	<code>testMean</code> logical; for copula method only; whether or not the predictive means (if calculated) should be tested for being reasonable

`nmax = 50` for local kriging: the number of nearest observations that should be used for a kriging prediction or simulation, where nearest is defined in terms of the space of the spatial locations. By default, 50 observations are used.

`ngrid = 100` The number of grid points to be used if an Averaged Cumulative Distribution Function (ACDF) needs to be computed for unbiased kriging

`nsim=100` Number of simulations when needed

`block = numeric(0)` Block size; a vector with 1, 2 or 3 values containing the size of a rectangular in x-, y- and z-dimension respectively (0 if not set), or a data frame with 1, 2 or 3 columns, containing the points that discretize the block in the x-, y- and z-dimension to define irregular blocks relative to (0,0) or (0,0,0) - see also the details section of [predict.gstat](#). By default, predictions or simulations refer to the support of the data values.

`processType = "gaussian"` If known - the distribution of the data. Defaults to gaussian, analytical solutions also exists in some cases for logNormal. This setting only affects a limited number of methods, e.g. the block prediction

`confProj = FALSE` If set, the program will attempt conform projections in [preProcess](#), calling the function [conformProjections](#).

`debug.level = 0` Used in some functions for giving additional output. See individual functions for more information.

`nclus = 1` it is possible to use parallel processing for some interpolation methods (currently only the copula method), `nclus` defines the number of processes to spawn. This requires previous installation of the `doSNOW` package

... Additional parameters that do not exist in the default parameter set, this could be parameters necessary for new methods within or outside the [intamap-package](#)

**Value**

A list of the parameters with class `intamapParams` to be included in the object described in [intamap-package](#)

**Note**

This function will mainly be called by [createIntamapObject](#), but can also be called by the user to create a parameter set or update an existing parameter set. If none of the arguments is a list of class `IntamapParams`, the function will assume that the argument(s) are modifications to the default set of parameters.

If the function is called with two lists of parameters (but the first one is not of class `IntamapParams`) they are both seen as modifications to the default parameter set. If they share some parameters, the parameter values from the second list will be applied.

**Author(s)**

Jon Olav Skoien

## References

<http://www.intamap.org/>

## See Also

[createIntamapObject](#)

## Examples

```
# Create a new set of intamapParameters, with default parameters:
params = getIntamapParams()
# Make modifications to the default list of parameters
params = getIntamapParams(newPar=list(removeBias = "local",
                                     secondParameter = "second"))
# Make modifications to an existing list of parameters
params = getIntamapParams(oldPar = params,newPar = list(predictType = list(exc=TRUE)))
```

---

`getInterpolationMethodNames`  
*get interpolation method names*

---

## Description

get interpolation method names

## Usage

```
getInterpolationMethodNames()
```

## Details

none

## Value

character array with names for available interpolation methods

## Author(s)

Edzer Pebesma

## References

<http://www.intamap.org/>

## Examples

```
getInterpolationMethodNames()
```

---

`intamap`*Example data for the intamap package*

---

**Description**

Salted (slightly modified) observations from the European gamma radiation network

**Usage**

```
data(intamap)
```

**Details**

The data set is a salted data set from the European gamma radiation network <http://eurdep.jrc.ec.europa.eu/Basic/Pages/Public/Home/Default.aspx>. Salted does in this context mean that all locations and observation values have been randomized through addition of a random value.

---

`intamapExampleObject`*Simulated Intamap Object*

---

**Description**

Intamap object of class "copula" containing a simulated data set with 196 spatial locations.

**Usage**

```
data(intamapExampleObject)
```

**Details**

The data set is a realization of a random field generated using a Gaussian copula and generalized extreme value distributed margins (location=40,shape=0.5, scale=30). The correlation function is Matern (Stein's representation) with range=4, kappa=3 and nugget=0.05. Furthermore, there is geometric anisotropy with direction=pi/3 and ratio=2.

**See Also**

[spatialPredict.copula](#), [estimateParameters.copula](#)

**Examples**

```
## Not run:
data(intamapExampleObject)
## estimate parameters for the copula model
intamapExampleObject<-estimateParameters(intamapExampleObject)
## make predictions at unobserved locations
intamapExampleObject<-spatialCopula(intamapExampleObject)

## End(Not run)
```

---

interpolate	<i>spatial interpolation</i>
-------------	------------------------------

---

**Description**

interpolate is a function that interpolates spatial data

**Usage**

```
interpolate(observations, predictionLocations, outputWhat, obsChar = NA,
            methodName = "automatic", maximumTime = 30, optList = list())
interpolateBlock(observations, predictionLocations, outputWhat,
                 blockWhat = "none", obsChar = NA, methodName = "automatic",
                 maximumTime = 30,
                 optList = list())
```

**Arguments**

observations	observation data, object of class <a href="#">SpatialPointsDataFrame</a> . The observation to be interpolated has to be identified through the column name value
predictionLocations	prediction locations, object of class <a href="#">SpatialPointsDataFrame</a>
outputWhat	list with names what kind of output is expected, e.g. <code>outputWhat = list(mean=TRUE, variance=TRUE, nsim = 5)</code>
blockWhat	List defining particular output for block predictions. See <a href="#">createIntamapObject</a>
obsChar	list with observation characteristics, used by some interpolation methods
methodName	name of interpolation method to be used, see <a href="#">spatialPredict</a> for more details, or <code>automatic</code> , to let the method be decided by the program, based on <code>maximumTime</code> and type of variables input
maximumTime	the maximum time available for interpolation, will be compared to the result of <a href="#">predictTime</a> for the requested method, or for finding the best interpolation method able to finish within this time
optList	list; further options, mainly passed to <a href="#">createIntamapObject</a> as the argument <code>params</code> , directly as arguments, but some are used locally in <code>interpolate</code> and <code>interpolateBlock</code> :

**formulaString** passed as argument to `createIntamapObject`, if no formulaString is given, it will default to `value~1` if observations has a column named `value` or to `col1 ~ 1` where `col1` is the first column of the observations

**set.seed** the possibility to pass a seed value to `interpolate`, to assure reproducible results also for methods relying on random numbers

## Details

The functions `interpolate` and `interpolateBlock` are particularly implemented for being called by a Web Processing Server (WPS), but they can also be used interactively. The only necessary arguments are `observations` and `predictionLocations`. It is also recommended to set `outputWhat`, and `blockWhat` if necessary. If `outputWhat` contains `nsim`, the return table will also contain a number of realisations, for methods able to return simulations.

`interpolate` can use different interpolation methods for the result. The function will internally call the following functions which can be method specific.

- `preProcess`
- `estimateParameters`
- `spatialPredict`
- `postProcess`

An indication of available methods can be given by `methods(estimateParameters)` or `methods(spatialPredict)`. The method can be set through the argument `methodName`, or through the built-in automatic selection method. There are different criteria that helps in selecting the right method for a particular data set. There are four methods that are available for the automatic choice: `automap`, `psgp` (from the separate package `psgp`) `copula` and `transgaussian` are the possibilities. First of all, if observation errors are present, the `psgp` method is preferred. If not, it is checked whether the data appear to deviate significantly from normality. This is assumed to be the case if any of the tests below are TRUE:

```
test[1] = length(boxplot.stats(dataObs)$out)/length(dataObs) > 0.1
test[2] = fivenum(dataObs)[3] - fivenum(dataObs)[2] < IQR(dataObs)/3
test[3] = fivenum(dataObs)[4] - fivenum(dataObs)[3] < IQR(dataObs)/3
g = boxcox(dataObs ~ 1,lambda=seq(-2.5,2.5,len=101),plotit=FALSE)$y
test[4] = g[71] < sort(g)[91]
```

where `fivenum` defines the Tukey five number statistic and `IQR` finds the interquartile range of the data. If the minimum of `dataObs` is  $\leq 0$ , `min(dataObs) + sdev(dataObs)` is added to all values. At last, the function calls `predictTime` for an estimate of the prediction time. If any of the tests above were true and the estimated prediction time for `copula` prediction is below `maximumTime`, the `copula` method is chosen. If any of the tests were TRUE and the estimated prediction time is too long, `transGaussian` kriging is chosen, as long as all values are above zero. If any of the tests are true for a set of observations with negative or zero-values, `automap` is chosen, but a warning is issued.

The element `methodParameters` in the object being returned is a string that makes it possible to regenerate the variogram model or the copula parameters in `createIntamapObject`. This is particularly useful when the function is called through a WPS, when the element with the estimated

parameters cannot be preserved in a state that makes it possible to use them for a later call to `interpolate`.

The possibility for doing parallel processing is enabled for some of the main methods. To be able to take advantage of multiple CPUs on a computer, the package `doSNOW` must be downloaded, additionally the parameter `nclus` must be set to a value larger than 1. Parallel computation is not taken into account when estimating the prediction times.

## Value

An `intamap` object, which is a list with elements, see [intamap-package](#). The exact number and names of these elements might vary due to different methods applied, but the list below shows the most typical:

<code>observations</code>	the observations, as a <code>Spatial*DataFrame</code>
<code>predictionLocations</code>	the prediction locations, as a <code>Spatial</code> -object
<code>formulaString</code>	the relationship between independent and dependent variables, <code>value</code> or <code>obs</code> used if not given
<code>outputWhat</code>	a list of the prediction types to return
<code>returnPlot</code>	logical; if <code>TRUE</code> a diagnostic plot is returned
<code>anisPar</code>	the estimated anisotropy parameters
<code>variogramModel</code>	the estimated parameter for the method, can also be e.g. <code>copulaParams</code> for the copula method or <code>inverseDistancePower</code> for inverse distance power method.
<code>methodParameters</code>	a string, that when parsed, can be used to regenerate the variogram model or copula parameters. Useful for repeated calls to <code>interpolate</code> when it is not necessary to reestimate the parameters.
<code>predictions</code>	a <code>Spatial*DataFrame</code> with predictions, for most methods with a format equal to the output from <code>krige</code> with predicted mean and variance as <code>var1.pred</code> and <code>var1.var</code>
<code>outputTable</code>	a matrix, organized in a convenient way for the calling WPS; first row: x-coordinates, second row: y-coordinates; further rows: output elements as specified by <code>outputWhat</code>
<code>processPlot</code>	a diagnostic plot from the interpolation
<code>processDescription</code>	some textual descriptions of the interpolation process, including warnings

## Author(s)

Edzer Pebesma

## References

<http://www.intamap.org/>

**See Also**

[createIntamapObject](#), [estimateParameters](#), [spatialPredict](#), [intamap-package](#)

**Examples**

```
data(meuse)
coordinates(meuse) = ~x+y
meuse$value = meuse$zinc
data(meuse.grid)
gridded(meuse.grid) = ~x+y
x = interpolate(meuse, meuse.grid, list(mean=TRUE, variance=TRUE))
summary(t(x$outputTable))
```

---

methodParameters

*generate string for generation of method parameters*

---

**Description**

function that generates a parsable string of identified method parameters for an intamap interpolation object

**Usage**

```
## Default S3 method:
methodParameters(object)
## S3 method for class 'copula'
methodParameters(object)
## S3 method for class 'idw'
methodParameters(object)
```

**Arguments**

**object** a list object. Most arguments necessary for interpolation are passed through this object. See [intamap-package](#) for further description of the necessary content of this variable

**Details**

The function creates a text-string that makes it possible to add the the method parameters (anisotropy and idw-parameter, variogram model or copula parameters) to the object in a later call to [createIntamapObject](#) or [interpolate](#) without having to re-estimate the parameters. This function is particularly useful when [interpolate](#) is called from a Web Processing Service, and the user wants to reuse the method parameters. The function is mainly assumed to be called from within [interpolate](#).

The default method assumes a variogram model of *gstat* type, e.g. a variogram similar to what can be created with a call to [vgm](#). Also *psgp* uses this variogram model.

**Value**

A string that, when parsed, will recreate the methodParameters

**Author(s)**

Jon Olav Skoien

**References**

<http://www.intamap.org/>

**Examples**

```
library(intamap)
data(meuse)
coordinates(meuse) = ~x+y
meuse$value = log(meuse$zinc)
# set up intamap object:
krigingObject = createIntamapObject(
  observations = meuse,
  formulaString = as.formula('value~1'), class = "automap")
# do estimation steps:
krigingObject = estimateParameters(krigingObject)
krigingObject = methodParameters(krigingObject)

# Create a new object
krigingObject2 = createIntamapObject(observations = meuse,
  formulaString = as.formula('value~1'),
  params = list(methodParameters = krigingObject$methodParameters))

krigingObject$variogramModel
krigingObject2$variogramModel
```

---

plotIntamap

*plot intamap objects*

---

**Description**

Plotting function for intamap-objects of the type described in [intamap-package](#)

**Usage**

```
plotIntamap(object, zcol = "all", sp.layout = NULL, plotMat = c(2,2), ...)
## S3 method for class 'copula'
plot(x, ...)
## S3 method for class 'idw'
plot(x, ...)
## S3 method for class 'automap'
```

```

plot(x, ...)

## S3 method for class 'linearVariogram'
plot(x, ...)
## S3 method for class 'transGaussian'
plot(x, ...)
## S3 method for class 'yamamoto'
plot(x, ...)

```

### Arguments

object	a list object. Most arguments necessary for interpolation are passed through this object. See <a href="#">intamap-package</a> for further description of the necessary content of this variable
x	intamap object, when plot is called directly
zcol	a list of column names to be plotted; if equal to all, the column names will correspond to all possible column names from outputWhat (see <a href="#">createIntamapObject</a> )
sp.layout	an object that can contain lines, points and polygons that function as extra layout; see <a href="#">spplot</a> for more information
plotMat	an array of length two with the number of rows and columns of plots per page
...	other parameters that can be passed to other plot functions (e.g. <a href="#">plot</a> , <a href="#">spplot</a> , <a href="#">automapPlot</a> and <a href="#">xyplot</a> )

### Details

All the plot methods above are simple wrapper functions around the plotIntamap function.

### Value

A plot of some of the elements of object. This will typically be the sample variogram and the fitted variogram model (if a method based on variograms has been used) and all the predictions.

### Author(s)

Jon Olav Skoien

### References

<http://www.intamap.org/>

### Examples

```

data(meuse)
meuse$value = log(meuse$zinc)
data(meuse.grid)
coordinates(meuse) = ~x+y
coordinates(meuse.grid) = ~x+y
object = interpolate(meuse,meuse.grid,outputWhat = list(mean = TRUE, variance = TRUE, excprob = 7, excprob = 8, quar
plot(object)

```

---

postProcess	<i>pre-process data</i>
-------------	-------------------------

---

## Description

post-processing of data for the [intamap-package](#). The function will typically call other functions for adding back biases, aggregation etc.

## Usage

```
## Default S3 method:  
postProcess(object, ...)  
## S3 method for class 'idw'  
postProcess(object, ...)
```

## Arguments

object	a list object. Most arguments necessary for interpolation are passed through this object. See <a href="#">intamap-package</a> for further description of the necessary content of this variable
...	other parameters that can be passed to functions called from preProcess

## Details

The function postProcess includes code for postprocessing an object after interpolation. The function can easily be replaced by more specific methods relevant for a certain data set, doing more data specific things in addition to what is done in the default method.

## Value

An object of same type as above, but with new elements. Most important from the default method is the outputTable, a matrix, organized in a convenient way for the calling WPS; first row: x-coordinates, second row: y-coordinates; further rows: output elements as specified by outputWhat (see [createIntamapObject](#))

## Author(s)

Edzer J. Pebesma

## References

<http://www.intamap.org/>

---

 predictTime

*Time prediction for intamap package methods*


---

### Description

Functions that gives a time estimate for an intamap function given the number of observations and predictionLocations

### Usage

```
predictTime(nObs, nPred, class, formulaString, calibration=FALSE,
            outputWhat, FUN = "spatialPredict",...)
```

### Arguments

nObs	An integer or an array of integers containing the number of observations.
nPred	An integer or an array of integers containing the number of predictions.
class	class of intamapObject, which interpolation method to be used
formulaString	the formula of the request, mainly to see if the request has independent variables
calibration	enables or disables time calibration - not properly implemented yet
outputWhat	List defining the requested type of output, see <a href="#">createIntamapObject</a>
FUN	A string with the intamap functions name, now obsolete
...	other arguments needed to define the intamap object.

### Details

The function is based on timeModels being available in the workspace. This is a [loess](#)-model that has been fitted to different calls to a range of interpolation requests with synthetically generated data in [generateTimeModels](#).

### Value

An integer or an array of integers with the predicted times.

### Note

RUN FIRST `generateTimeModels()` or `estimateTimeModel()` in order to save time Models to workspace. It might take some time!

---

preProcess	<i>pre-processing of data</i>
------------	-------------------------------

---

### Description

pre-processing of the data for the [intamap-package](#) package.

### Usage

```
## Default S3 method:  
preProcess(object, ...)  
## S3 method for class 'idw'  
preProcess(object, ...)
```

### Arguments

object	see <a href="#">intamap-package</a> ; list that should at least contain (i) an element called observations of class <a href="#">SpatialPointsDataFrame</a> . The measured values should be named value, and (ii) an element params of class list, by calling the function <a href="#">getIntamapParams</a> . (iii) Usually, the object will also contain an element called predictionLocations, of a class extending <a href="#">Spatial</a> .
...	Additional parameters

### Details

The function `preProcess` includes code for preprocessing an object before interpolation. The function can easily be replaced by more specific methods relevant for a certain data set. Functions can be called from `preProcess` according to the settings in parameters in the object, set by the function [getIntamapParams](#).

### Value

The input object is returned, after its components have been pre-processed.

### Author(s)

Jon Olav Skoien

### References

<http://www.intamap.org/>

---

rotateAnisotropicData *rotateAnisotropicData*

---

### Description

This function applies an isotropic transformation of the coordinates specified in object.

### Usage

```
rotateAnisotropicData(object,anisPar)
```

### Arguments

object	(i) An Intamap type object (see <a href="#">intamap-package</a> ) containing one <a href="#">SpatialPointsDataFrame</a> data frame named observations which includes the observed values (ii) or a <a href="#">SpatialPointsDataFrame</a> which includes both coordinates and observations or (iii) <a href="#">SpatialPoints</a> which includes only coordinates to be rotated.
anisPar	An array containing the anisotropy parameters (anisotropy ratio and axes orientation) (see <a href="#">estimateAnisotropy</a> ) for the rotation. If object is the output of <a href="#">estimateAnisotropy</a> function, these parameters are part of object. In cases (ii) and (iii) anisPar defines the two anisotropy parameters. For the definition of the anisotropy parameters see <a href="#">estimateAnisotropy</a> .

### Details

This function performs a rotation and rescaling of the coordinate axes in order to obtain a new coordinate system, in which the observations become statistically isotropic. This assumes that the estimates of the anisotropy ratio and the orientation angle provided in anisPar are accurate.

### Value

(i) A modified object with transformed coordinates if rotateAnisotropicData is called with an Intamap object as input (see [intamap-package](#)) or (ii) the transformed coordinates if a [SpatialPointsDataFrame](#) is used as input or (iii) the transformed coordinates if a [SpatialPoints](#) object is the input.

### Author(s)

Hristopulos Dionisis, Spiliopoulos Giannis

### References

- [1] <http://www.intamap.org>
- [2] A. Chorti and D. T. Hristopulos (2008). Non-parametric Identification of Anisotropic (Elliptic) Correlations in Spatially Distributed Data Sets, *IEEE Transactions on Signal Processing*, 56(10), 4738-4751 (2008).

**See Also**

estimateAnisotropy

**Examples**

```
library(gstat)
data(sic2004)
coordinates(sic.val)=~x+y
sic.val$value=sic.val$dayx

anisPar <- estimateAnisotropy(sic.val)
print(anisPar)

rotatedObs <- rotateAnisotropicData(sic.val,anisPar)

newAnisPar <- estimateAnisotropy(rotatedObs)
print(newAnisPar)
```

---

spatialPredict

*Spatial prediction*


---

**Description**

spatialPredict is a generic method for spatial predictions within the [intamap-package](#). A series of methods have been implemented, partly based on other R-packages (as [krige](#)), other methods have been developed particularly for the INTAMAP project. The object has to include a range of variables, further described in [intamap-package](#). The prediction method is chosen based on the class of the object.

**Usage**

```
## S3 method for class 'automap'
spatialPredict(object, nsim = 0, ...)
## S3 method for class 'copula'
spatialPredict(object, ...)
## Default S3 method:
spatialPredict(object, ...)
## S3 method for class 'idw'
spatialPredict(object, ...)
## S3 method for class 'linearVariogram'
spatialPredict(object, nsim = 0, ...)

## S3 method for class 'transGaussian'
spatialPredict(object, nsim = 0, ...)
## S3 method for class 'yamamoto'
spatialPredict(object, nsim = 0, ...)
```

**Arguments**

object	a list object. Most arguments necessary for interpolation are passed through this object. See <a href="#">intamap-package</a> for further description of the necessary content of this variable
nsim	number of simulations to return, for methods able to return simulations
...	other arguments that will be passed to the requested interpolation method. See the individual interpolation methods for more information.

**Details**

The function `spatialPredict` is a wrapper around different spatial interpolation methods found within the `link{intamap-package}` or within other packages in R. It is for most of the methods necessary to have parameters of the correlation structure included in `object` to be able to carry out the spatial prediction. Below are some details about particular interpolation methods

`default` a default method is not really implemented, this function is only created to give a sensible error message if the function is called with an object for which no method exist

`automap` If the object already has an element `variogramModel` with variogram parameters, `krige` is called. If the this is not a part of the object, `estimateParameters` is called to create this element.

`copula` spatial prediction using [bayesCopula](#)

`idw` applies inverse distance modelling with the `idp-power` found by `estimateParameters.idw`

`linearVariogram` this function estimates the process using an unfitted linear variogram; although variance is returned it can not be relied upon

`transGaussian` spatial prediction using [krigeTg](#)

`yamamoto` spatial prediction using [yamamotoKrige](#)

It is also possible to add to the above methods with functionality from other packages, if wanted. See description on <http://www.intamap.org/newMethods.php> You can also check which methods are available from other packages by calling

```
>methods(spatialPredict)
```

**Value**

a list object similar to `object`, but extended with predictions at a the set of locations defined `object`.

**Author(s)**

Jon Olav Skoien

**References**

<http://www.intamap.org/>

**See Also**

[gstat](#), [autoKrige](#), [createIntamapObject](#), [estimateParameters](#), [intamap-package](#)

**Examples**

```

# This example skips some steps that might be necessary for more complicated
# tasks, such as estimateParameters and pre- and postProcessing of the data
data(meuse)
coordinates(meuse) = ~x+y
meuse$value = log(meuse$zinc)
data(meuse.grid)
gridded(meuse.grid) = ~x+y
proj4string(meuse) = CRS("+init=epsg:28992")
proj4string(meuse.grid) = CRS("+init=epsg:28992")

# set up intamap object:
obj = createIntamapObject(
  observations = meuse,
  predictionLocations = meuse.grid,
  targetCRS = "+init=epsg:3035",
  params = getIntamapParams(),
  class = "linearVariogram"
)

# do interpolation step:
obj = spatialPredict(obj) # spatialPredict.linearVariogram

```

---

summaryIntamap

*summary intamap objects*


---

**Description**

summary function for intamap-objects of the type described in [intamap-package](#)

**Usage**

```

summaryIntamap(object, ...)
## S3 method for class 'copula'
summary(object, ...)
## S3 method for class 'idw'
summary(object, ...)
## S3 method for class 'automap'
summary(object, ...)

## S3 method for class 'linearVariogram'
summary(object, ...)
## S3 method for class 'transGaussian'
summary(object, ...)
## S3 method for class 'yamamoto'
summary(object, ...)

```

**Arguments**

- object a list object. Most arguments necessary for interpolation are passed through this object. See [intamap-package](#) for further description of the necessary content of this variable
- ... parameters to be passed to the default summary function for each element

**Value**

A summary of some of the elements of object.

**Author(s)**

Jon Olav Skoien

**References**

<http://www.intamap.org/>

---

timeModels

*models for estimating prediction time in intamap package*

---

**Description**

This is a standard model for estimating the prediction time within the [intamap-package](#). It was created by the function [generateTimeModels](#), on a 64 bits Linux server running R version 2.9.0 and [intamap-package](#) version 1.1.15.

The prediction time will depend on the speed of the local machine, on the version of R and [intamap-package](#), and on the installed libraries. It is therefore strongly recommended to run [generateTimeModels](#) on the local machine and store the result in the workspace if the predicted interpolation time is of real interest. `timeModels` in the workspace will be chosen if available.

It is not necessary to load the data set, this happens automatically in `predictTime` if `timeModels` if the object is not already existing in the workspace.

**Usage**

```
data(timeModels)
```

**Author(s)**

Jon Olav Skoien

**References**

<http://www.intamap.org/>

---

unbiasedKrige	<i>Unbiased kriging</i>
---------------	-------------------------

---

### Description

unbiasedKrige is a function for modifying a kriging prediction to a prediction that can be assumed to be unbiased for a certain threshold.

### Usage

```
unbiasedKrige(object, formulaString, observations, predictionLocations,
              model, outputWhat, nmax, nsim, yamamoto, debug.level, ...)
```

### Arguments

object	either an object of the <code>intamap</code> type (see <a href="#">intamap-package</a> for further description of the necessary content of this variable) or the output from the function <code>krige</code> in <code>gstat</code> . If the object is a result from the <code>intamap</code> procedure <code>spatialPredict</code> , the remaining arguments are not necessary.
formulaString	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name $z$ , for ordinary and simple kriging use the formula $z \sim 1$ ; for universal kriging, suppose $z$ is linearly dependent on $x$ and $y$ , use the formula $z \sim x + y$
observations	a <code>Spatial*</code> <code>DataFrame</code> with observations; should contain the dependent variable, independent variables, and coordinates
predictionLocations	the predictionLocations, only necessary if the method is "IWQSEL" and formulaString contains independent variables. Should preferentially be a grid if the method is "IWQSEL"
model	variogram model of dependent variable (or its residuals), defined by a call to <code>vgm</code> or <code>autofitVariogram</code>
outputWhat	Argument with type of unbiasedness method ("MOK" or "IWQSEL") and the thresholds.
nmax	for local kriging: the number of nearest observations that should be used in simulations for the "IWQSEL" method in terms of the space of the spatial locations. Defaults to <code>nmax = 10</code> when object is a <code>Spatial*</code> <code>DataFrame</code> .
nsim	number of simulations necessary if the method is "IWQSEL". Defaults to <code>nsim = 100</code> when object is a <code>Spatial*</code> <code>DataFrame</code> .
yamamoto	logical describing if the yamamoto approach is to be used in simulations. Defaults to <code>yamamoto = FALSE</code> when object is a <code>Spatial*</code> <code>DataFrame</code> .
debug.level	debug level, passed to subfunctions
...	other arguments that will be passed to subfunctions

## Details

It is a fact that predictions from kriging tend to be biased towards the mean of the process. The function `unbiasedKrige` is a function that adds one or more predictions to the original output, which are assumed to be unbiased relative to a certain threshold. The two methods supported are the IWQSEL-method (Craigmile, 2006) and MOK (Skoien et al, 2008).

## Value

an object of type `intamap`, as described in [intamap-package](#), or a `Spatial*DataFrame` with one or more new prediction columns, representing different methods and thresholds.

## Author(s)

Jon Olav Skoien

## References

Craigmile, P. F., N. Cressie, T. J. Santner, and Y. Rao. 2006. A loss function approach to identifying environmental exceedances. *Extremes*, 8, 143-159.

Skoien, J. O., G. B. M. Heuvelink, and E. J. Pebesma. 2008. Unbiased block predictions and exceedance probabilities for environmental thresholds. In: J. Ortiz C. and X. Emery (eds). Proceedings of the eight international geostatistics congress. Gecamin, Santiago, Chile, pp. 831-840.

<http://www.intamap.org/>

## Examples

```
library(intamap)
data(meuse)
data(meuse.grid)
coordinates(meuse) = ~x+y
gridded(meuse.grid) = ~x+y

predictionLocations = coarsenGrid(meuse.grid,5,)
vmod = autofitVariogram(log(zinc)~1,meuse)$var_model
prediction = krige(log(zinc)~1,meuse,predictionLocations,vmod)
summary(prediction)

prediction = unbiasedKrige(prediction,log(zinc)~1,
  meuse, model = vmod, outputWhat = list(MOK = 6.0, MOK = 7.0, IWQSEL=7.0))
summary(prediction)
```

yamamotoKrige

*kriging and simulation with alternative kriging variance***Description**

ordinary kriging and simulation with an alternative kriging variance

**Usage**

```
yamamotoKrige(formula,Obs, newPoints,model,nsim = 0,nmax = 20)
```

**Arguments**

formula	formula that defines the dependent variable as a linear model of independent variables; suppose the dependent variable has name $z$ , for ordinary and simple kriging use the formula $z \sim 1$ ; only ordinary kriging is currently implemented, formula is hence mainly used to identify the dependent variable
Obs	<a href="#">SpatialPointsDataFrame</a> with observations
newPoints	<a href="#">Spatial</a> object with prediction locations, either points or grid
model	variogram model - of the type that can be found by a call to <a href="#">vgm</a>
nsim	integer; if set to a non-zero value, conditional simulation is used instead of kriging interpolation. For this, sequential Gaussian simulation is used, following a single random path through the data.
nmax	for local kriging: the number of nearest observations that should be used for a kriging prediction or simulation, where nearest is defined in terms of the space of the spatial locations. By default, all observations are used.

**Details**

The term `yamamotoKrige` comes from the paper of Yamamoto (2000) where he suggests using local variance around the kriging estimate (weighted with the kriging weights) as an alternative kriging variance. This as a solution to more reliable estimates of the kriging variance also when the stationarity assumption has been violated. The method was applied by Skoien et al. (2008), who showed that it can have advantages for cases where the stationarity assumption behind kriging is violated.

If the number of observations is high, it is recommended have `nmax` lower. This is partly because the method relies on positive kriging weights. The method to do this adds the norm of the largest negative weight to all weights, and rescales. This tends to smooth the weights, giving a prediction closer to the average if a too large number of observation locations is used.

**Value**

Either a [Spatial](#)\*[DataFrame](#) with predictions and prediction variance, in the columns `var1.pred` and `var1.var`, together with the classical ordinary kriging variance in `var1.ok`, or simulations with column names `simx` where  $x$  is the number of the simulation.

**Author(s)**

Jon Olav Skoien

**References**

Skoien, J. O., G. B. M. Heuvelink, and E. J. Pebesma. 2008. Unbiased block predictions and exceedance probabilities for environmental thresholds. In: J. Ortiz C. and X. Emery (eds). Proceedings of the eight international geostatistics congress. Santiago, Chile: Gecamin, pp. 831-840.

Yamamoto, J. K. 2000. An alternative measure of the reliability of ordinary kriging estimates. *Mathematical Geology*, 32 (4), 489-509.

<http://www.intamap.org>

**Examples**

```
library(intamap)
data(sic2004)
coordinates(sic.val) = ~x+y
coordinates(sic.test) = ~x+y
variogramModel = autofitVariogram(joker~1,sic.val)$var_model
newData = yamamotoKrige(joker~1,sic.val,sic.test,variogramModel,nmax = 20)
summary(newData)
plot(sqrt(var1.ok)~var1.pred,newData)
# Kriging variance the same in regions with extreme values
plot(sqrt(var1.var)~var1.pred,newData)
# Kriging standard deviation higher for high predictions (close to extreme values)
```

# Index

- \*Topic **datasets**
  - intamap, [28](#)
  - intamapExampleObject, [28](#)
- \*Topic **htest**
  - estimateAnisotropy, [18](#)
- \*Topic **nonparametric**
  - estimateAnisotropy, [18](#)
- \*Topic **spatial**
  - bayesCopula, [6](#)
  - blockPredict, [7](#)
  - checkSetup, [9](#)
  - coarsenGrid, [10](#)
  - conformProjections, [11](#)
  - copulaEstimation, [13](#)
  - createIntamapObject, [15](#)
  - estimateAnisotropy, [18](#)
  - estimateParameters, [21](#)
  - estimateTimeModel, [23](#)
  - generateTimeModels, [24](#)
  - getIntamapParams, [25](#)
  - getInterpolationMethodNames, [27](#)
  - intamap-package, [2](#)
  - interpolate, [29](#)
  - methodParameters, [32](#)
  - plotIntamap, [33](#)
  - postProcess, [35](#)
  - preProcess, [37](#)
  - rotateAnisotropicData, [38](#)
  - spatialPredict, [39](#)
  - summaryIntamap, [41](#)
  - unbiasedKrige, [43](#)
  - yamamotoKrige, [45](#)
- autofitVariogram, [22, 43](#)
- autoKrige, [8, 9, 40](#)
- automapPlot, [34](#)
- bayesCopula, [6, 15, 40](#)
- blockPredict, [7](#)
- boxcox, [21, 22](#)
- checkSetup, [9](#)
- coarsenGrid, [10](#)
- conformProjections, [5, 11, 26](#)
- copulaEstimation, [6, 7, 13, 22](#)
- createIntamapObject, [12, 15, 22, 23, 26, 27, 29, 30, 32, 34–36, 40](#)
- estimateAnisotropy, [18, 38](#)
- estimateParameters, [7, 15, 21, 30, 32, 40](#)
- estimateParameters.copula, [28](#)
- estimateTimeModel, [23](#)
- fivenum, [30](#)
- generateTimeModels, [24, 36, 42](#)
- getIntamapParams, [5, 16, 18, 25, 37](#)
- getInterpolationMethodNames, [27](#)
- gstat, [9, 40, 43](#)
- intamap, [28](#)
- intamap-package, [6–9, 11–13, 15–18, 21, 22, 24–26, 31–35, 37–44](#)
- intamap-package, [2](#)
- intamapExampleObject, [28](#)
- interpolate, [21, 22, 25, 29, 32](#)
- interpolateBlock (interpolate), [29](#)
- IQR, [30](#)
- krige, [8, 22, 31, 39, 40, 43](#)
- krigeTg, [22, 40](#)
- loess, [24, 36](#)
- methodParameters, [32](#)
- observations (intamap), [28](#)
- optim, [14](#)
- plot, [34](#)
- plot.automap (plotIntamap), [33](#)
- plot.copula (plotIntamap), [33](#)

`plot.default` (`plotIntamap`), 33  
`plot.idw` (`plotIntamap`), 33  
`plot.linearVariogram` (`plotIntamap`), 33  
`plot.transGaussian` (`plotIntamap`), 33  
`plot.yamamoto` (`plotIntamap`), 33  
`plotIntamap`, 33  
`postProcess`, 5, 25, 30, 35  
`predict.gstat`, 5, 8, 26  
`predictTime`, 24, 25, 29, 30, 36  
`preProcess`, 5, 9, 17, 26, 30, 37

`rotateAnisotropicData`, 38

`Spatial`, 5, 10, 11, 16, 31, 37, 43–45  
`SpatialGridDataFrame`, 11, 16  
`SpatialLinesDataFrame`, 16  
`SpatialPixelsDataFrame`, 11, 16  
`SpatialPoints`, 38  
`SpatialPointsDataFrame`, 5, 16, 18, 19, 29, 37, 38, 45  
`SpatialPolygons`, 8  
`SpatialPolygonsDataFrame`, 16  
`spatialPredict`, 7, 15, 22, 29, 30, 32, 39  
`spatialPredict.block` (`blockPredict`), 7  
`spatialPredict.copula`, 28  
`spplot`, 34  
`summary.automap` (`summaryIntamap`), 41  
`summary.copula` (`summaryIntamap`), 41  
`summary.idw` (`summaryIntamap`), 41  
`summary.linearVariogram` (`summaryIntamap`), 41  
`summary.transGaussian` (`summaryIntamap`), 41  
`summary.yamamoto` (`summaryIntamap`), 41  
`summaryIntamap`, 41

`timeModels`, 42

`unbiasedKrige`, 16, 43

`vgm`, 32, 43, 45

`xyplot`, 34

`yamamotoKrige`, 40, 45