



Proposal/Contract no.: 033811
Project start: September 1, 2006
Project end: August 31, 2009



INTAMAP

Interoperability and Automated Mapping

SIXTH FRAMEWORK PROGRAMME

PRIORITY IST-2005-2.5.12

ICT for Environmental Risk Management

Deliverable 1.3

R implementation of a simple mapping algorithm and development / integration of support tools

Title of Deliverable	R implementation of a simple mapping algorithm and development / integration of support tools
Deliverable reference number	INTAMAP D1.3
Related WP and Tasks	WP1, Task 1.3
Type of Document	Public
Authors	UU and AST
Date	October 7, 2008
Version	1.0

Project coordinator

Dr. Edzer J. Pebesma

Utrecht University, The Netherlands

E-mail: e.pebesma@geo.uu.nl

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

Revision History

Version	Date	Changes	Authors
1.0	31-07-2008	First draft	J. O. Skøien, E. J. Pebesma

Active partners: UU, AST

Legal Notices

The information in this document is subject to change without notice. The Members of the INTAMAP Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the INTAMAP Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Contents

1	Introduction	2
2	Interpolation package	3
2.1	Creation of a data object for interpolation	3
2.2	Pre-processing	5
2.3	Interpolation service	6
2.4	Post-processing	7
3	Conclusions	8

Executive Summary

The R INTAMAP package is currently under development. This report gives a description of some of the features in the package at a certain stage in this development. The package is functional in its current form, although a lot of the promised functionality is still not finally implemented in the package.

The latest version of the package can be downloaded and installed from: <http://intamap.svn.sourceforge.net/viewvc/intamap/intamap/>

1 Introduction

The interpolation service developed within the INTAMAP project will serve as the back-end of the system. This back end will need to be able to handle not just the mapping applications but also a range of other common GIS functionality, such as conversion of coordinate systems, and computing statistics over a given geometry. The implementation of the service has been decided to take place within the statistical environment/language R, which is free and has an open source code.

This report describes the first implementation of such a back-end in R. It does not yet include the full functionality promised in the project application, but it includes a simple mapping algorithm in addition to a range of the supporting tools that is necessary for a further development of the interpolation service. The most recent version of the package can at all times be accessed through a Source Version Number (SVN) repository:

<http://intamap.svn.sourceforge.net/viewvc/intamap/intamap/>

The partners of the project can here include their contributions to the package. A simple description on how to contribute and work with this package is given in the wikipedia page: http://wiki.intamap.org/index.php/SVN_and_R

The development version of the interpolation package has been developed as a package that can later be uploaded to the Comprehensive R Archive Network (CRAN) of the R-project, as a package available for other users. This means that the package has been developed with help files for the functions available to a user or a web-service, tests for the R maintainers and demo-files for users. In a development phase, both pre-processing steps and interpolation steps are located within the same package. The data for testing has to be included separately, as these are restricted and cannot be stored on the same publicly accessible server as the package itself.

This report presents an overview of the implementation of the R package. Details about each function can be found in the appendix, which includes the help pages for each separate function. It should be noted that these help pages are also still under development.

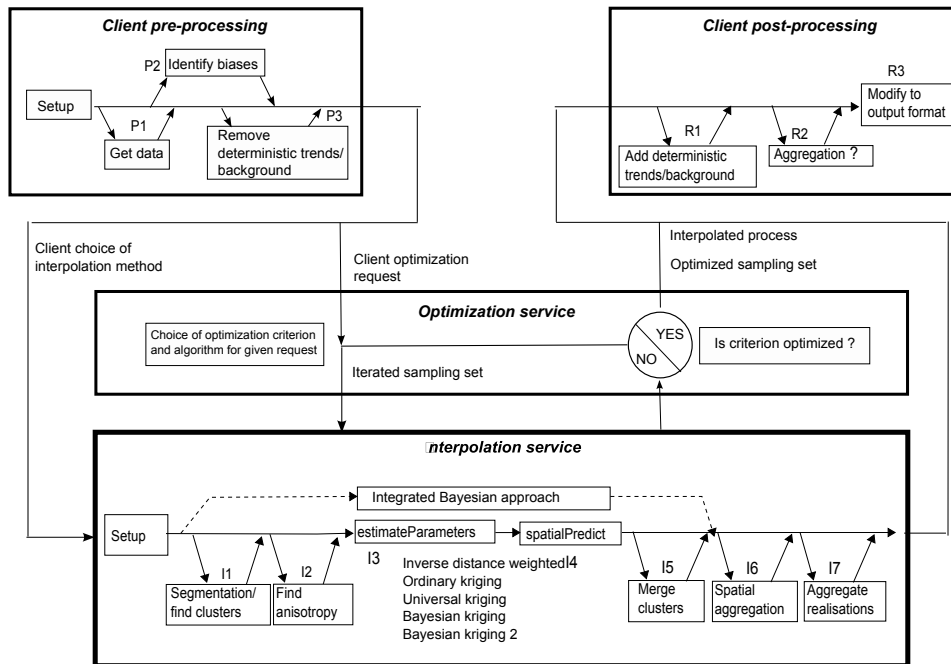


Figure 1: Flow scheme for interpolation service

2 Interpolation package

The package can be conceptualised to consist of four different main modules that can be stored on the same server, but where the communication between the modules also can take place in an interoperable manner if they are stored on different servers. This is conceptualized in Figure 1.

There are four different main modules in this figure; pre-processing, interpolation service, post-processing and optimization service. The last one is an example on how add-on services within the INTAMAP framework can integrate with the interpolation service. Further description of these will be in the deliverables dealing with optimization.

The current version of the intamap package includes all the three main modules listed above. Hence, there is no discussion on the possible difficulties of passing the variables between the modules, as could be the case if the different modules are applied on different servers. It should also be mentioned that these difficulties are not related to the intamap package itself.

Below each of the main modules will be described in a different sub-section. However, the first subsection will give a short description on the data object that we will use, and the use of classes within R and the intamap package.

2.1 Creation of a data object for interpolation

The INTAMAP package for interpolation will consist of a large number of functions that will handle different parts of the interpolation request. All these functions will take a large number of arguments, which could make the work flow rather complicated. We have for

this reason decided that all the work within the INTAMAP package should be performed on one data object, which will then contain all necessary information. The creation of this object can be seen as a part of the pre-processing steps, but will we have included a short description about the object below in this section.

The data object currently has to be created manually, as a `list()` where new data can be added as components by different functions. At the moment, for a data set to be interpolated, we can create the object as following:

```
> krigingObject = list(  
>pointData = observations,  
>predictionLocations = predictionLocations,  
>params = getIntamapParams()  
> )  
> class(krigingObject) = c("idw")
```

This creates the data object `krigingObject`, consisting of `observations`, `predictionLocations`, in addition to a set of parameters. Calling `getIntamapParams()` without parameters gives an object with default parameters for interpolation.

The R package `intamap` will have a set of generic functions such as `preProcess`, `estimateParameters`. We have chosen to let the more specific methods for each of these generic functions be called using the S3 method mechanism for class-structure within R. Using `krigingObject` above as an example, the class for this object has here been set to "idw". The methods applied on this object will then be `preProcess.idw` and `estimateParameters.idw`. In addition to a class defining the method, it will also be possible to set a class defining the type of data set, if this has implications for the methods that should be applied on the object. This will particularly influence pre- and post-processing functions. For the `eurdep` data set, we add "eurdep" as a class:

```
> class(krigingObject) = c("eurdep","idw")
```

This will allow for particular data dependent pre- and post-processing functions such as `preProcess.eurdep` and `postProcess.eurdep`

There will also be a set of default methods, such as `preProcess.default` and `estimateParameters.default`. These will be applied if there are no method available for a certain class.

The function `getParams` sets several parameters for both pre-processing and interpolation. The current set of parameters include (with default values):

```
formulaString= as.formula(value~1),  
doAnisotropy = FALSE,  
removeBias= NA,  
addBias = NA,  
biasRemovalMethod = "LM",  
doSegmentation = FALSE,  
maxCluster = 0,  
numberOfClusters = 1,
```

```

nmax = Inf,
nGrid = 1600,
processType="gaussian",
predictType = data.frame(pred = TRUE, var=TRUE,exc=TRUE,
                          threshCor=NA,yamamoto = FALSE,
                          blockMax=TRUE,blockFat=TRUE),
thresh=200,
isEmergency = FALSE,
confProj = FALSE

```

Not all of these parameters are used yet. The parameters of `removeBias` and `addBias` refers to which sorts of biases/drifts to remove/add. Also elevation and soil effects can be added to this parameter. If some parameters are to be changed from the default, this can be done e.g. by:

```
> params = getParams(doAnisotropy=TRUE)
```

when defining the object.

The object described above can also contain a range of other components that are useful or necessary for the other functions in the `intamap` package. These can be:

targetCRS the target projection

boundaries shapefile with regional boundaries

boundCRS projections of regional boundaries

regCode dataframe or array with regional codes

boundFile file with boundaries

boundariesPair a dataframe with points describing the boundaries between different regions

localBias earlier estimates of local biases

regBias earlier estimates of regional biases

The advantage of the current structure is that if a new component will need additional components to the main object, this can easily be added.

2.2 Pre-processing

The pre-processing module can be installed on a different server than the interpolation service itself. The procedures within this module can be data-dependent, hence a specific pre-processing service might be needed for modelling of a new variable. However, parts of the service have been made rather generic, to ease porting to other variables. Although the pre-processing steps have been tested on the EURDEP data, most of the steps are relevant also for other variables with minor need for modification.

Reading of (EURDEP)-data is currently seen as a part of the pre-processing steps. More generic methods for input through a web-service interface will be the source of data in the final version of this package. However, as the package will also be uploaded to CRAN as a new interpolation package with a range of different methods, some different methods for reading data will also be a part of the package.

In the pre-processing module, data is read, different parameters set, and the object mentioned in section 2.1 object is created for parameter estimation and prediction. This module can be data-dependent with different routines for different sorts of data, but there will also be developed a more generic pre-processing module if the data type is unknown. The help pages from all procedures are included in the appendix of this report. The pre-processing steps so far consist of following routines:

eurdepLoad function that can load different sorts of eurdep data

nuts2Boundaries function that calls getNuts, which downloads nuts boundaries from eurostat

preProcess a function that pre-processes the data, calling one or more of the following functions:

conformProjections Conforms the projections of different components of object

cleanData Removes errors and duplicated observations from object

findElevation Finds elevation of observations in pointData

setLocalGroup Sets group IDs for local groups. Can be replaced by lgFUN in preProcess

findLocalBias Find biases for overlapping networks

removeLocalBias Removes biases between overlapping networks

findRegionalBoundaries Find points that define adjacent boundaries between regions

findRegionalBias Find biases for neighbouring networks

removeRegionalBias Remove biases between neighbouring networks

2.3 Interpolation service

The interpolation service will be the core of the processes developed within the INTAMAP framework. There will be a range of different methods to choose from and/or combine to actually perform the interpolation, depending on the user's requests. This section gives a short overview of the different methods.

There are two main functions in the interpolation service:

estimateParameters a generic function for parameter estimation

spatialPredict a generic function for spatial predictions

Different methods are being developed for these functions, to be applied on the object depending on the class of the object. The early versions of the INTAMAP package consists of the following methods:

idw inverse distance weighted interpolation

linearKriging kriging using a linear variogram

automap automatic interpolation based on the automap package by Paul Hiemstra

There is additionally also methods to handle predictions for blocks and administrative regions, which also includes some additional point predictions methods that will later be directly applicable also for point predictions. See Deliverable 4.3 for more information about these methods.

There are also several other functions that will be parts of the interpolation service, and that can be called if the user is interested. These include:

doSegmentation find clusters in the data for anisotropy detection and prediction (The prediction methods do not yet take segmentation into account)

doAnisotropy Find anisotropy parameters for the data set, for each cluster (The prediction methods do not yet take the anisotropy parameters into account)

Some functions are indicated in Figure 1, but not yet finally implemented or correctly combined with the rest of the interpolation service. These are:

Merge clusters Merge predictions done for different clusters, will maybe be included in the spatialPredicition function

Spatial aggregation prediction for blocks and polygons are further described in Deliverable 4.3

Aggregate realisations Calculate and return statistics for prediction methods that involves simulations

The interpolation service will eventually return an object that includes the spatial predictions, for points or polygons. Predictions can in this case also refer to the predictive distributions, including the distribution function, moments, and/or percentiles.

2.4 Post-processing

Post-processing of data refers to the processing that might take place at the client side, but is currently a part of the flow chain. Post-processing can consist of different functions:

Add biases/trends It can be necessary to add back regional trends or biases to the prediction, to make the predictions correspond with actual observations. However, it is not possible to add local biases, as these cannot be attached to spatial locations of the predictions

Aggregation Additional aggregation (spatial or from the realisations) to be in accordance with the users requests

User requested post-processing This can include particular post-processing that might be interesting for a certain user group, this is usually data dependent

3 Conclusions

The R INTAMAP package is currently under development. This report gives a description of some of the features in the package at a certain stage in this development. The package is functional in its current form, although a lot of the promised functionality is still not finally implemented in the package.

00Introduction	<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:
obj = list(
  pointData = readOGR("PG:dbname=postgis", "eurdep.data"),
  predictionLocations = readOGR("PG:dbname=postgis", "eurdep1km.grid"),
  targetCRS = "+init=epsg:3035",
  params = getIntamapParams()
)
class(obj) = c("idw","eurdep")

# 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`, `maptools`, `akima`, `automap`). All four packages need to be available to the R session. `automap` is at the moment only downloadable from the web page of Paul Hiemstra, but it will be available from CRAN in due course.

```
# set up intamap object:
obj = list(
  pointData = readOGR("PG:dbname=postgis", "eurdep.data"),
  predictionLocations = readOGR("PG:dbname=postgis", "inspire1km.grid"),
  targetCRS = "+init=epsg:3051",
  params = getIntamapParams()
)
class(obj) = c("idw","eurdep")
```

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` and `eurdep`. This will determine which *versions* of `preProcess`, `parameterEstimate` etc will be used: R package `intamap` provides *methods* for each of the *generic* functions `preProcess`, `estimateParameters`, `spatialPredict`, `postProcess`. Two classes are used in this case, as the choice of pre- and post-processing steps tend to be data-dependent, whereas the two other methods are depends on which function we would like to apply on our data.

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.default      estimateParameters.idw
[3] estimateParameters.linearVariogram
```

Now if a partner provides 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`.

Source code management

It is not strictly necessary to have all code in a single package; it is also possible to provide add-on code with methods (and documentation) in separate packages, which can then be required by package intamap. They should however all be installed (and working) before testing can take place. We have a source code management system running (svn) to manage sources we jointly work on.

Input object components

`pointData` 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

`constantBIAS` numeric; bias to subtract from data

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

`formulaString` formula string for parameter estimation and prediction functions

`doAnisotropy` Defining whether anisotropy should be calculated

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

`addBias` Defining which biases to be added in the `postProcess` function.

`doCluster` Defining if the region should be divided into clusters. Not properly implemented yet

`maxCluster` Maximum number of clusters to use

`numberOfClusters` Fixing the number of clusters

`methodIdentifier` For identifying different methods. Is maybe obsolete

`isEmergency` If set, this parameter will make the program focus on predictions in the emergency areas. Parameters for identifying biases will be ignored due to the difficulties of estimating such biases in an emergency situation

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

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 [preProcess](#), see [00Introduction](#)
quiet logical; TRUE to suppress OK statement

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/>

cleanData

Check and clean data before interpolation

Description

The function is a help function for cleaning up obvious errors in the data. It depends on the actual data, but might include upper and lower limits for the data, removal of duplicate observations at a single locations (necessary to avoid singularities in the kriging matrix) etc.

Usage

```
## Default S3 method:  
cleanData(object)  
## S3 method for class 'eurdep':  
cleanData(object)
```

Arguments

`object` data frame with observations

Details

This function checks the data before interpolation. Current methods include:
`duplicate` checks if there are duplicated observations at any locations and deletes extra observations.

Value

data frame with observations without duplicated observations at any locations

Author(s)

Edzer J. Pebesma

References

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

`conformProjections` *Getting conformed projections*

Description

Getting a conformed projection for a set of `Spatial`* classes necessary for interpolation in the `intamap`-package.

Arguments

`object` a list object. Most arguments necessary for interpolation are passed through this object. See [00Introduction](#) for further description of the necessary content of this variable

Details

`conformProjections` is a function that attempts to reproject all projected components in `object` to one common projection. This is necessary because several of the functions in a typical spatial interpolation work flow inside the `INTAMAP` package require that the components have a common projection. In addition, there are some of the functions that are not able to deal with unprojected spatial objects, i.e. objects with coordinates given in latitude and longitude. `conformProjections` will hence also reproject all components that have coordinates in latitude and longitude, even in the cases where they all have the same projections.

The common projection will be the first possible of the following list:

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

defaultCRS A default projection is used when all the components have coordinates in latitude and longitude, and no interpolation or target projection is given. The chosen projection is CRS("init=epsg:3035") which is suitable for European data, but not necessarily for data from other regions.

Value

A list of the parameters to be included in the **object** described in [00Introduction](#)

Author(s)

Edzer J. Pebesma

References

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

estimateParameters *Automatic estimation of correlation structure parameters*

Description

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

Usage

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

Arguments

<code>object</code>	a list object. Most arguments necessary for parameter estimation are passed through this object. See 00Introduction for further description of the necessary content of this variable
<code>...</code>	other arguments that will be passed to the requested interpolation method. See the individual methods for more information
<code>idpRange</code>	range of idp (inverse distance weighting power) values over which to optimize mse
<code>nfolds</code>	number of folds in n-fold cross validation

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 the different methods.

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

Value

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

Author(s)

Edzer J. Pebesma

References

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

<code>findElevation</code>	<i>Check and clean data before interpolation</i>
----------------------------	--

Description

The function will find the elevation of a station from a DEM if it is currently not set. The function is not implemented yet.

Usage

```
## Default S3 method:  
findElevation(object)  
## S3 method for class 'eurdep':  
findElevation(object)
```

Arguments

<code>object</code>	data frame with observations
---------------------	------------------------------

Details

This help file is not finished!!!

Value

data frame with observations without duplicated observations at any locations

Author(s)

Edzer J. Pebesma

References

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

<code>findLocalBias</code>	<i>Finds biases between overlapping networks</i>
----------------------------	--

Description

The function tries to identify differences between different networks of observation stations that share a region. From these differences, biases are estimated.

Usage

```
## Default S3 method:  
findLocalBias(object,regCode,gid = "group")  
## S3 method for class 'eurdep':  
findLocalBias(object,regCode,gid = "group")
```

Arguments

<code>object</code>	data frame with observations
<code>regCode</code>	list of codes defining different regional networks
<code>gid</code>	name of column identifying groups of local networks

Details

The method

Value

data frame with observations, including network group numbers

Author(s)

Edzer J. Pebesma

References

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

`findRegionalBias` *Finding the regional biases*

Description

Method for identifying regional biases (in most cases biases between countries)

Arguments

`object` an object of class `SpatialPointsDataFrame`, at least containing observations and a regional identification code (`regCode`)

`regionalBoundaries` A `SpatialPointsDataFrame` with points defining the boundaries between regions. This can be found using `findRegionalBoundaries`.

`minKrige` Setting a minimum number of observations necessary for kriging

Value

A `data.frame` with the biases for each country with uncertainty.

Author(s)

Edzer J. Pebesma

References

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

`findRegionalBoundaries`
Finding the regional boundaries

Description

Method for identifying points on the boundaries between regions (in most cases biases between countries)

Arguments

`fileName` A character string with the name of a file with boundaries. If `fileType` is "Shape", only the name before the dot should be supplied.

`regions` A `SpatialPolygonsDataFrame` with the polygons defining the boundaries of each separate region.

`fileType` Defining the file type if a file (or set of files if `fileType` = "Shape") is to be read. Other possibilities are "rda" for R-binaries.

`projOrig` The original projection of the boundaries

`projNew` If a different projection is wanted for the output

Value

A [SpatialPointsDataFrame](#) with points defining the boundaries between regions.

Author(s)

Edzer J. Pebesma

References

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

`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 [00Introduction](#)

Usage

```
getIntamapParams(formulaString= as.formula(value~1),doAnisotropy = FALSE,
  removeBias= NA,  addBias = NA, biasRemovalMethod = "LM", doSegmentation = FALSE, maxCluster =
  numberOfClusters = 1,nmax = Inf,nGrid = 1600, processType="gaussian",
  predictType = data.frame(pred = TRUE, var=TRUE,exc=TRUE,threshCor=NA,yamamoto = FALSE,
    blockMax=TRUE,blockFat=TRUE),
  thresh=200, isEmergency = FALSE,confProj = FALSE )
```

Arguments

<code>formulaString</code>	formula string for parameter estimation and prediction functions
<code>doAnisotropy</code>	Defining whether anisotropy should be calculated
<code>removeBias</code>	Defining whether biases should be removed, and in case yes, which ones (<code>localBias</code> and <code>regionalBias</code> implemented)
<code>addBias</code>	Defining which biases to be added in the postProcess function.
<code>biasRemovalMethod</code>	character; specifies which methods to use to remove bias. See below.
<code>doSegmentation</code>	Defining if the predictions should be subject to segmentation. Segmentation has been implemented, but not the use of it.
<code>maxCluster</code>	Maximum number of clusters to use. May be obsolete
<code>numberOfClusters</code>	Fixing the number of clusters. May be obsolete
<code>nmax</code>	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.
<code>nGrid</code>	The number of grid points to be used if an Averaged Cumulative Distribution Function (ACDF) needs to be computed for unbiased kriging

<code>processType</code>	If known - the distribution of the data. Defaults to gaussian, analytical solutions also exists in some cases for logNormal
<code>predictType</code>	
<code>pred</code>	Usual kriging prediction
<code>var</code>	Usual kriging error
<code>exc</code>	Exceedance probability
<code>threshCor</code>	Correction of predictions according to the threshold given as parameter <code>thresh</code> . More information given in Skøien et al (2008).
<code>yamamoto</code>	To be set equal to TRUE if local variance is to be used instead of kriging variance
<code>blockMax</code>	Prediction of maximum within block, if block predictions
<code>blockFat</code>	Prediction of area within block above threshold
<code>thresh</code>	An eventual threshold which predictions should be compared with, either for unbiased predictions related to this threshold, or for predictions of exceedance probabilities.
<code>isEmergency</code>	If set, this parameter will make the program focus on predictions in the emergency areas. Parameters for identifying biases will be ignored due to the difficulties of estimating such biases in an emergency situation.
<code>confProj</code>	If set, the program will attempt conform projections in <code>preProcess</code> , calling the function <code>conformProjections</code> .

Value

A list of the parameters to be included in the codeobject described in [00Introduction](#)

Author(s)

Edzer J. Pebesma

References

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

<code>postProcess</code>	<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 'eurdep':
postProcess(object, ...)
## S3 method for class 'idw':
postProcess(object, ...)
## S3 method for class 'linearVariogram':
postProcess(object, ...)
```

Arguments

object see [00Introduction](#)
...

Author(s)

Edzer J. Pebesma

References

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

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

Description

pre-processing of the data for the INTAMAP package. If local biases or country biases are assumed to be present, these will be removed. If they are not present from earlier runs of the program, they will first be identified. If it is possible to add elevation to the pointData set, it will be added. If there are more observations at one location, these will be removed.

Usage

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

Arguments

object see [00Introduction](#); list that should at least contain (i) a component called `pointData` of class `SpatialPointsDataFrame`. The measured values should be named `value`, and (ii) a component called `predictionLocations`, of a class extending `Spatial`. In addition, parameters need to be set in the

component `params` of class `list`, by calling the function `getIntamapParams`. Other components can be necessary and/or reduce computation time, depending on the functions to be called from `preProcess`. Some of these are:

- `countryBoundaries` which is a `SpatialPointsDataFrame` with paired country borders, `localBias` which is the local biases found with e.g. a data set from an earlier time step, using the function `findLocalBias`.
- `lgFUN` function to give local groups for data, if they are not set with `gid` (see below). This function should have input and output similar to `setLocalGroup`
- `cid`
- `gid`
- `...` Additional parameters to the functions that might be called by `preProcess`, such as `findLocalBias`, `removeLocalBias`, `findRegionalBoundaries`, `findRegionalBias` and `removeRegionalBias`

Details

The function `preProcess` carries out several tasks to prepare an object for interpolation. Most of the methods are called from `preProcess` according to the settings in parameters in the `object`, set by the function `getIntamapParams`. The `preProcessing` step will call the following functions in this order:

- `conformProjections` Conforms the projections of different components of `object`
- `cleanData` Removes errors and duplicated observations from `object`
- `findElevation` Finds elevation of observations in `pointData`
- `setLocalGroup` or `lgFUN` Sets group IDs for local groups
- `findLocalBias` Find biases for overlapping networks
- `removeLocalBias` Removes biases between overlapping networks
- `findRegionalBoundaries` Find points that define adjacent boundaries between regions
- `findRegionalBias` Find biases for neighbouring networks
- `removeRegionalBias` Remove biases between neighbouring networks

Value

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

Author(s)

Jon Olav Skoien

References

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

Description

Method for identifying regional biases (in most cases biases between countries)

Arguments

`formulaString` formula string for parameter estimation and prediction functions
`doAnisotropy` Defining whether anisotropy should be calculated
`removeBias`
`addBias`
`doCluster` Defining if the region should be divided into clusters. Not properly implemented yet
`maxCluster`
`numberOfClusters` Fixing the number of clusters
`methodIdentifier`

`isEmergency` If set, this parameter will make the program focus on predictions in the emergency areas. Parameters for identifying biases will be ignored due to the difficulties of estimating such biases in an emergency situation
`confProj` If set, the program will attempt conform projections in `preProcess`, calling the function `{link{regionalBias}}`.

Value

A list of the parameters to be included in the object described in [00Introduction](#)

Author(s)

Edzer J. Pebesma

References

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

`removeLocalBias`

Removes identified biases from local networks

Description

removes the biases between networks sharing a region, identified in [findLocalBias](#)

Usage

```
## Default S3 method:  
removeLocalBias(object,localBias)  
## S3 method for class 'eurdep':  
removeLocalBias(object,localBias)
```

Arguments

<code>object</code>	Data frame with observations with same format as <code>pointData</code> described in OOIntroduction)
<code>localBias</code>	List of data frames, one for each country, each containing biases for different networks in the country. A further description of this object is found in findLocalBias .

Details

This function subtracts biases that are a function of local networks. The function is particularly meant to work together with [findLocalBias](#).

Value

Data frame with observations, with the identified biases removed.

Author(s)

Jon Olav Skoien

References

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

See Also

[findLocalBias](#)

`removeRegionalBias` *Removes identified biases from Regional networks*

Description

removes the biases between networks sharing a region, identified in `findRegionalBias`

Usage

```
## Default S3 method:  
removeRegionalBias(object, regionalBias)  
## S3 method for class 'eurdep':  
removeRegionalBias(object, regionalBias)
```

Arguments

<code>object</code>	Data frame with observations with same format as pointData described in 00Introduction)
<code>regionalBias</code>	List of data frames, one for each country, each containing biases for different networks in the country.

Details

This function subtracts biases that are a function of Regional networks. The function is particularly meant to work together with `findRegionalBias`.

Value

data frame with observations, with biases removed

Author(s)

Edzer J. Pebesma

References

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

`setLocalGroup`

Sets group number for national/regional networks

Description

Function to set the network group for local networks, networks sharing the same observation region.

Usage

```
## S3 method for class 'eurdep':  
setLocalGroup(object,...)
```

Arguments

`object` data frame with observations
`...` other arguments to be passed to different versions of the function

Details

This function is data dependent. Current methods include:

`eurdep` Sets network groups numbers for networks in Hungary and Slovenia

`default` does not do anything

Similar functions can be given as input to the function [preProcess](#)

Value

The data frame with observations, including network group numbers in a new column.

Author(s)

Jon Olav Skoien

References

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

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 [00Introduction](#). The prediction method is chosen based on the class of the object.

Usage

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

Arguments

<code>object</code>	a list object. Most arguments necessary for interpolation are passed through this object. See 00Introduction for further description of the necessary content of this variable
<code>...</code>	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 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

`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 = list(
  pointData = meuse,
  predictionLocations = meuse.grid,
  targetCRS = "+init=epsg:3035",
  params = getIntamapParams()
)
class(obj) = "linearVariogram"

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