Package 'IsoriX'

September 11, 2016

Version 0.4-1

Date 2016-09-10

Encoding UTF-8

Title Isoscape Computation and Inference of Spatial Origins using Mixed Models

Author Alexandre Courtiol [aut, cre], François, Rousset [aut], Stephanie Kramer-Schadt [aut]

Maintainer Alexandre Courtiol <alexandre.courtiol@gmail.com>

Depends R (>= 3.1.0)

Imports graphics, grDevices, grid, latticeExtra, numDeriv, proxy, RandomFields, raster, rasterVis (>= 0.30), sp, spaMM (>= 1.10), stats, utils

Description Building isoscapes using mixed models and inferring the geographic origin of organisms based on their isotopic ratios. This package is essentially a simplified interface to several other packages. It uses 'spaMM' for fitting and predicting isoscapes, and assigning an organism's origin depending on its isotopic ratio. 'IsoriX' also relies heavily on the package 'rasterVis' for plotting the maps using lattice.

License GPL (>= 2)

Suggests Cairo, lattice, maps, R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-11 16:32:52

R topics documented:

IsoriX-package	*Isoscape Computation and Inference of Spatial Origins using Mixed Models*

Description

IsoriX can be used for building isoscapes using mixed models and inferring the geographic origin of organisms based on their isotopic signature. This package is essentially a simplified interface combining several other packages. It uses the package **spaMM** for fitting and predicting isoscapes, and for performing the assignment. **IsoriX** also heavily relies on the package **rasterVis** for plotting the maps using the powerful **lattice** visualization system.

Details

We describe below, step-by-step, the general work-flow that aims at performing the construction of an isoscape and the assignment of organisms of unknown geographic origin(s) based on their isotopic signature. We advise to also read all dedicated help pages of functions mentioned hereafter.

The statistical methods will not be detailed in this document but should soon be available as a vignette, a publication, or both (we are currently working on it).

1. Fitting the isoscape model with `Isofit`:

   The function `Isofit` fits a geostatistical model, which approximates the relationship between the topographic features of a location and its isotopic signature (see `Isofit` for details). The model fits observations of isotopic delta values at several geographic locations (hereafter, called *sources*). One common type of sources used in ecology is the delta values for deuterium in precipitation water collected at weather stations, but one may also use measurements performed on sedentary organisms. In either case, the accuracy of the isoscape (and thereby the accuracy of assignments) increases with the number and spatial coverage of the sources. Because fitting a geostatistical model may take several hours for large datasets, we have stored an already fitted model for users willing to explore our package (see `GNIP_Model`).

2. Preparing the elevation raster with RElevate:

   Building isoscapes and assigning organisms to their origin requires an adequate elevation raster, i.e. a matrix representing altitude data on a spatial grid. The function RElevate allows restricting the extent of the raster to the area covered by isoscape data (in order to avoid extrapolation) and to reduce the resolution of the original elevation raster (in order to speed up computation in all following steps). Note that aggregating the raster may lead to different results for the assignment, because the elevation of raster cells changes depending on the aggregation function, which in turn will affect model predictions.

   We will soon provide a link to download an elevation raster for the entire world at a resolution of one altitude per square-km, and other rasters may be used. We have also stored a low resolution raster in our package (see elevationrastersmall) for users to try things out, but we do not encourage its use for real application.

3. Predicting the isoscape across the area covered by the elevation raster with Isoscape:

   The function Isoscape generates the isoscape: it uses the fitted geostatistical model to predict the isotopic values for each raster cell defined by the elevation raster. Our package allows the production of fine-tuned isoscape figures (using the function plot.isoscape). Alternatively, the isoscape rasters can be exported as ascii raster and edited in any Geographic Information System (GIS) software (see Isoscape for details).

4. Fitting the calibration model with Calibfit:

   In most cases, organisms are of another kind than the sources used to build the isoscape (e.g. the isoscape is built on precipitation isotopic values and organisms are not water drops, i.e. the deuterium values of their keratin structures were modulated by their distinct physiology). In this situation, one must use sedentary organisms to study the relationship between the isotopic values in organisms and that of their environment. The function Calibfit fits a statistical model on such a calibration dataset.

   If the isoscape is directly built from isotopic values of organisms, there is no need to fit a calibration model.

5. Inferring spatial origins of organisms with Isorix:

   The function Isorix tests for each location across the isoscape if it presents a similar isotopic signature than the unknown origin of a given individual(s). This assignment procedure considered the uncertainty stemming from the model fits (geostatistical model and calibration model). The function plot.isorix then draw such assignment by plotting the most likely origin with the prediction region around it. When several organisms are being assigned, both individuals assignments and a single assignment for the whole group can be performed.

Note

Please note that for now, the geographic coordinates (latitude, longitude) of any spatial data (locations, rasters) must be given in decimal degrees following the WGS84 spheroid standard.

Author(s)

Alexandre Courtiol <alexandre.courtiol@gmail.com>,

Stephanie Kramer-Schadt <kramer@izw-berlin.de>

References

Bowen, G. J., Wassenaar, L. I., Hobson, K. A. (2005). Global application of stable hydrogen and oxygen isotopes to wildlife forensics. Oecologia, 143(3): 337-348.

Examples

```
### A simple workflow for IsoriX
### is provided in a vignette:
### vignette("Workflow", "IsoriX")
```

---

assignmentdata	*Assignment dataset*

Description

This dataset contains data from Voigt, Lehmann and Greif (2015). It contains deuterium delta values of fur keratin from bats captured in 2008, 2009 and 2013 from their roosting sites in Bulgaria. We only retained the bats of the genus Myotis from the original study. The data can be used as an example to perform assignments using the function `Isorix`.

Usage

```
data(assignmentdata)
```

Format

A *dataframe* with 244 observations on 3 variables:

[, 1]	animalID	(*Factor*)	Identification of the animal
[, 2]	species	(*Factor*)	Animal species name
[, 3]	tissue.value	(*numeric*)	Deuterium delta value of the tissue

Source

data directly provided by the authors of the following publication

References

Voigt, C.C., Lehmann, D., Greif, S. (2015). Stable isotope ratios of hydrogen separate mammals of aquatic and terrestrial food webs. Methods in Ecology and Evolution 6(11).

See Also

`Isorix` to perform assignments

Examples

```
data(assignmentdata)
head(assignmentdata)
str(assignmentdata)
```

Calibfit   *Fit the calibration model*

Description

This function fits a model that establishes the relationship between the isotopic values of organisms (e.g. tissues such as hair, horn, ivory or feathers) and the isotopic values of their environment (e.g. precipitation).

Usage

```
Calibfit(
calib.data,
isofit,
verbose=interactive(),
control.optim=list()
)
```

Arguments

calib.data	A *dataframe* containing the calibration data (see note below)
isofit	The fitted isoscape model created by `Isofit`
verbose	A *logical* indicating whether information about the progress of the procedure should be displayed or not while the function is running. By default verbose is *TRUE* if users use an interactive R session and *FALSE* otherwise.
control.optim	A *list* to pass information to the argument control of the optim call (for advanced users only).

Details

The calibration model is a linear mixed-effects model (LMM) that fits the isotopic values of sedentary organisms as a linear function of the isotopic values in their environment (e.g. precipitation).

This function considers that the isotopic values from the environment (e.g. from precipitation) at the locations at which organisms were sampled are not known. The function therefore predicts these isotopic values from the geostatistical model fitted by the function `Isofit`, which is provided to Calibfit using the argument isofit.

The LMM used to fit the calibration function has a simple fixed-effect structure: an intercept and a slope. The random effect is more complex: it is normally distributed with mean zero, a certain variance between locations proportional to the squared fixed slope, and a covariance structure defined by the prediction covariance matrix of the isoscape model between the calibration locations. All models used in **IsoriX** will be soon detailed in an additional document.

This function is only needed in the case for which the assignment of organisms has to be performed within an isoscape that was built using another source of isotopic values (e.g., precipitation). This implies that if the isoscape had been fitted using isotopic ratios from sedentary animals, then this calibration step is not needed.

If source isotopic values are known at the locations where sedentary organisms were sampled, users should calibrate their data directly using the function [lm](#) by fitting tissue isotopic values as a function of source isotopic values.

Value

This function returns a *list* of class *calibfit* containing the fixed-effect estimates of the calibration function, the covariance of the fixed effects, the fitted calibration model, the original calibration data set with additional information added during the fit, and the location of the calibration points as spatial points.

Note

See [calibrationdata](#) to know which variables are needed to perform the calibration fit and their names.

See Also

[IsoriX](#) for the complete workflow

Examples

```
## Not run:
## The following example takes some time and will therefore not
## be run unless you type: example(Calibfit, run.dontrun=TRUE)

data(calibrationdata)
data(GNIP_Model)

## fit the calibration model
calib <- Calibfit(
calib.data=calibrationdata,
isofit=GNIP_Model)

## display minimal information
calib

## display more information
summary(calib)

## plot the calibration function
plot(calib)

## End(Not run)
```

---

calibrationdata   *Calibration dataset*

Description

This dataset contains deuterium delta values of fur keratin from sedentary bat species captured between 2005 and 2009 from Popa-Lisseanu et al. (2012). The data can be used as an example to fit a calibration model using the function `Calibfit`.

Usage

```
data(calibrationdata)
```

Format

A *dataframe* with 178 observations on 7 variables:

[, 1]	siteID	(*Factor*)	Identification of the sampling site
[, 2]	long	(*numeric*)	Longitude coordinate [decimal degrees]
[, 3]	lat	(*numeric*)	Latitude coordinate [decimal degrees]
[, 4]	elev	(*numeric*)	Elevation asl [m]
[, 5]	animalID	(*Factor*)	Identification of the sampled animal
[, 6]	species	(*Factor*)	Species name
[, 7]	tissue.value	(*numeric*)	Deuterium delta value of the tissue

Details

Users who wish to use their own dataset for calibration should create a *dataframe* of similar structure than this one (only the column 'species' can be dropped). The columns should possess the same names as the ones described above. If the elevation is unknown at the sampling sites, elevation information can be extracted from a high resolution elevation raster using the function **extract**. In this dataset, we retrieved elevations from the Global Multi-resolution Terrain Elevation Data 2010. Note that the original study used a different source of elevation data.

Source

data directly provided by the authors of the following publication

References

Popa-Lisseanu, A. G., Soergel, K., Luckner, A., Wassenaar, L. I., Ibanez, C., Kramer-Schadt, S., Ciechanowski, M., Goerfoel, T., Niermann, I., Beuneux, G., Myslajek, R. W., Juste, J., Fonderflick, J., Kelm, D., Voigt, C. C. (2012). A triple isotope approach to predict the breeding origins of European bats. PLoS ONE 7(1):e30388.

See Also

[Calibfit](#) to fit a calibration model

Examples

```
data(calibrationdata)
head(calibrationdata)
str(calibrationdata)

## The following example require to have downloaded
## a large elevation raster with the function GetElev()
## and will therefore not run unless you type:
## example(calibrationdata, run.dontrun=TRUE)

## Not run:
if(require(raster)){
    ### DELETE AND RECREATE ELEVATION DATA
    calibrationdata$elev <- NULL  ## we delete them

    ## we reconstruct the elevation data using an elevation raster:
    ## (see ?GetElev for details on how to get the tif file)
    elevationrasterbig <- raster("elevation_raster_gmted2010_30mn.tif")
    calibrationdata$elev <- extract(
        elevationrasterbig,
        cbind(calibrationdata$long, calibrationdata$lat))
    head(calibrationdata)
}


## End(Not run)
```

---

elevationrastersmall   *The raster of world elevation*

Description

This raster contains the elevation of the world surface [meters above sea level] with a resolution of approximately 100 square-km.

Usage

```
data(elevationrastersmall)
```

Format

A *RasterLayer*

Details

This raster contains elevation data of the world in a highly aggregated form corresponding to a resolution of approximately one elevation value per 100 square-km. This is only for the purpose of having a small and easy-to-handle file to practice, but it should not be used to perform real assignments!

In the example below, we show how we generated this small raster from a large original *DEM* (digital elevation model). The original raster has a resolution of approximately one elevation value per square-km (cell size of 30 arcseconds, i.e. 0.0083 decimal degrees). Although working on large rasters is technically problematic (memory and CPU greedy), we highly recommend to rely on high-resolution rasters with small to moderate aggregation levels in order to perform reliable assignment analyses. Indeed, large aggregation of raster cells can bias assignments due to the transformation of all elevations into a single value per aggregated raster cell.

We downloaded "Global Multi-resolution Terrain Elevation Data 2010" from:
http://topotools.cr.usgs.gov/gmted_viewer/
and converted it into a *tif* file. Because the original file is very large, we directly provide the url link of the *tif* file in the example below.

Source

http://topotools.cr.usgs.gov/gmted_viewer/

See Also

RElevate to crop and/or aggregate the elevation raster

Examples

```
## The following example require to have downloaded
## a large elevation raster with the function GetElev()
## and will therefore not run unless you type:
## example(elevationrastersmall, run.dontrun=TRUE)

## Not run:
### CREATING THE OBJECT elevationrastersmall

## download the tif file (ca. 700 Mb):
## (see ?GetElev for details on how to get the tif file)
# GetElev()

## convert the tif into R raster format:
library(raster)
elevationrasterbig <- raster("elevation_raster_gmted2010_30mn.tif")

## create the highly agregated elevation raster
## (90 sec on one of our computers):
elevationrastersmall <- RElevate(
    elevationrasterbig,
    aggregation.factor=100)
```

```
## End(Not run)
```

GetElev   *Download an elevation raster from internet*

Description

This function allows for the download of an elevation raster from internet. By default, it downloads the "Global Multi-resolution Terrain Elevation Data 2010" from our server. The file was orifginally downloaded from:
<http://topotools.cr.usgs.gov/gmted_viewer/>
and converted into a *tif* file by us.

Usage

```
GetElev(
path=NULL,
address=NULL,
overwrite=FALSE,
verbose=interactive()
)
```

Arguments

path	A *string* indicating where to store the file on the hard drive
address	A *string* indicating where to download the file from
overwrite	A *logical* indicating if an existing file should be re-downloaded
verbose	A *logical* indicating whether information about the progress of the procedure should be displayed or not while the function is running. By default verbose is TRUE if users use an interactive R session and FALSE otherwise.

Details

In the argument "path" is not provided, the file will be stored in the current working directory. The function can create new directories, so you can also indicate a new path. To download the elevation we prepared, leave the address to NULL.

Source

<http://topotools.cr.usgs.gov/gmted_viewer/>

Examples

```
## To download the high resolution
## raster in your current working
## directory, just type:
## GetElev()
```

GNIP_Europe	*European weather station data*

Description

This dataset contains the mean and variance of Deuterium delta precipitation values of European weather stations sampled between 1960 and 2013. Original data have been compiled by the International Atomic Energy Agency IAEA in Vienna (GNIP Project: Global Network of Isotopes in Precipitation).

Usage

```
data(GNIP_Europe)
```

Format

The *dataframe* includes 296 observations for the following variables:

[, 1]	stationID	(*Factor*)	Identification number of weather station
[, 2]	isoscape.value	(*numeric*)	Mean Deuterium delta value [per thousand]
[, 3]	var.isoscape.value	(*numeric*)	Variance in Deuterium delta value [per million]
[, 4]	lat	(*numeric*)	Latitude coordinate [decimal degrees]
[, 5]	long	(*numeric*)	Longitude coordinate [decimal degrees]
[, 6]	elev	(*numeric*)	Elevation asl [m]

Note

Users who wish to use their own dataset for isoscape model fitting should create a similar *dataframe*. All columns should possess the same names as the ones described above. We found occasional strong discrepancies between elevations from the GNIP_World dataset and elevation from the GMTED2010 satellite dataset. We also found occasional variation among elevations from the GNIP_World same weather stations. For these reasons, we decided to compute all elevations for GNIP_Europe using satellite estimation only, before we clarify the situation.

Source

Data extracted from IAEA (see below for more details).

References

GNIP Project IAEA Global Network of Isotopes in Precipitation:

http://www.iaea.org

See Also

QueryGNIP to see how we created this dataset from GNIP_World Isofit to fit an isoscape based on these data GNIP_World for the full non-aggregated dataset

Examples

```
data(GNIP_Europe)
head(GNIP_Europe)
```

GNIP_Model   *The fitted isoscape model for GNIP_Europe*

Description

This dataset contains an object of class isofit containing the mean model and residual dispersion model fitted on the data GNIP_Europe.

Usage

```
data(GNIP_Model)
```

Format

An object of class isofit

Details

This fitted isoscape model has been obtained by running the function Isofit on the isotopic source data stored in GNIP_Europe (see example).

See Also

Isofit for information about how to fit an isoscape model

GNIP_Europe for information about the weather station data used to fit this isoscape model

Examples

```
data(GNIP_Model)
GNIP_Model
plot(GNIP_Model)

## The following example takes a lot of time and will therefore not
## be run unless you type: example(GNIP_Model, run.dontrun=TRUE)

## Not run:
## We load the data
data(GNIP_Europe)

## We fit the isoscape model
GNIP_Model <- Isofit(iso.data=GNIP_Europe,
    mean.model.fix=list(elev=TRUE, lat.abs=TRUE),
    mean.model.rand=list("uncorr"=TRUE),
```

```
        disp.model.rand=list("uncorr"=TRUE))

## We created the object GNIP_Model stored in this package using
## save(GNIP_Model, file="GNIP_Model.rda", compress="xz")

GNIP_Model
plot(GNIP_Model)

## End(Not run)
```

GNIP_World   *World-wide weather station data*

Description

This dataset contains the mean and variance of Deuterium delta precipitation values of weather stations sampled between 1960 and 2013. Data have been compiled by the International Atomic Energy Agency IAEA in Vienna (GNIP Project: Global Network of Isotopes in Precipitation).

Usage

```
data(GNIP_World)
```

Format

The *dataframe* includes 54513 observations on the following variables:

[, 1]	lat	(*numeric*)	Latitude coordinate [decimal degrees]
[, 2]	long	(*numeric*)	Longitude coordinate [decimal degrees]
[, 3]	elev	(*numeric*)	Elevation asl [m]
[, 4]	isoscape.value	(*numeric*)	Deuterium stable hydrogen delta value [per thousand]
[, 5]	year	(*numeric*)	Year of sampling
[, 6]	month	(*numeric*)	Month of sampling
[, 7]	stationID	(*Factor*)	The unique identifier of the weather station

Details

The dataset contains non-aggregated data for 858 weather stations across the world.

This dataset is the raw data source and should not be directly used for fitting isoscapes.

Please use QueryGNIP to filter the dataset by time and location.

Note

Some elevations seem to be wrong... see GNIP_Europe for details.

Source

Data extracted from IAEA (see below for more details).

References

GNIP Project IAEA Global Network of Isotopes in Precipitation:

<http://www.iaea.org>

See Also

GNIP_Europe for information on what your working dataset should look like QueryGNIP to filter the dataset by time and location.

Examples

```
data(GNIP_World)
head(GNIP_World)
```

Isofit   *Fit the isoscape model*

Description

This function fits the isocape as a mixed model. The fitting procedures are done by the package **spaMM** which we use to jointly fit the mean isotopic values and their associated residual dispersion variance in a spatially explicit manner.

Usage

```
Isofit(
    iso.data,
  mean.model.fix=list(elev=FALSE, lat.abs=FALSE, lat.2=FALSE, long=FALSE, long.2=FALSE),
  disp.model.fix=list(elev=FALSE, lat.abs=FALSE, lat.2=FALSE, long=FALSE, long.2=FALSE),
   mean.model.rand=list(uncorr=FALSE, spatial=TRUE),
   disp.model.rand=list(uncorr=FALSE, spatial=TRUE),
  uncorr.terms=list(mean.model="lambda", disp.model="lambda"), ## or: "nugget"
  spaMM.method=list(mean.model="fitme", disp.model="fitme"), ## or: "corrHLfit", "HLfit"
   dist.method="Earth", ## or: "Euclidean"
   control.mean=list(),
   control.disp=list(),
   verbose=interactive()
)
```

Arguments

`iso.data`	The *dataframe* containing the data used for fitting the isoscape model
`mean.model.fix`	A *list* of *logical* indicating which fixed effects to consider in mean.fit
`disp.model.fix`	A *list* of *logical* indicating which fixed effects to consider in disp.fit
`mean.model.rand`	
	A *list* of *logical* indicating which random effects to consider in mean.fit
`disp.model.rand`	
	A *list* of *logical* indicating which random effects to consider in disp.fit
`uncorr.terms`	A *list* of two strings defining the parametrization used to model the uncorrelated random effects for mean.fit and disp.fit
`spaMM.method`	A *list* of two strings defining the spaMM functions used for mean.fit and disp.fit
`dist.method`	A *string* indicating the distance method
`control.mean`	A *list* of additional arguments to be passed to the call of mean.fit
`control.disp`	A *list* of additional arguments to be passed to the call of disp.fit
`verbose`	A *logical* indicating whether information about the progress of the procedure should be displayed or not while the function is running. By default verbose is TRUE if users use an interactive R session and FALSE otherwise.

Details

The detailed statistical definition of the isoscape model will be soon available in another document. Briefly, the fitting procedure of the isoscape model is divided into two fits: `mean.fit` and `disp.fit`. `mean.fit` corresponds to the fit of the "mean model", which we will use to predict the mean isotopic values at any location in other functions of the package. `disp.fit` corresponds to the fit of the "residual dispersion model", which we will use to predict the residual dispersion variance associated to the mean predictions. `mean.fit` is a linear mixed-effects model (LMM) with fixed effects, an optional spatial random effect with a Matern correlation structure and an optional uncorrelated random effect accounting for variation between weather station unrelated to their location. `disp.fit` is a Gamma Generalized LMM (Gamma GLMM) that also has fixed effects, an optional spatial random effect with a Matern correlation structure and an optional uncorrelated random effect. For the GLMM the residual variance is fixed to its theoretical expectation.

The *dataframe* `iso.data` must contain a single row per source location with the following columns: `isoscape.value` (the isotopic value), `var.isoscape.value` (the unbiased variance estimate of the isotopic value at the location), `n.isoscape.value` (the number of measurements performed at the location, could be 1) and `stationID` (a factor defining the identity of the sources at a given location).

The arguments `mean.model.fix` and `disp.model.fix` allow the user to choose among different fixed-effect structures for each model. These arguments are lists of booleans (TRUE or FALSE), which define which of the following fixed effects must be considered: the elevation (`elev`), the absolute value of the latitude (`lat.abs`), the squared latitude (`lat.2`), the longitude (`long`) and the squared longitude (`long.2`). An intercept is always considered in both models. By default, only intercept are being fitted. To fit an isoscape with the same fixed effect structure as the seminal Bowen model (Bowen et al. 2005), one should thus change `mean.model.fix` to `list(elev=TRUE, lat.abs=TRUE, lat.2=TRUE)`.

In the models, the mean (for the mean model) or the log residual variance (for the residual dispersion model) follow a Gaussian distribution around a constant value. The arguments `mean.model.rand`

and `disp.model.rand` allow to choose among different random effects for each model influencing the realizations of these Gaussian random processes. For each model one can choose not to include random effects or to include an uncorrelated random effect, a spatial random effect, or both (default). Setting `"uncorr"=TRUE` implies that the different realizations are identical for a given weather station (e.g. some micro-climate or some measurement errors trigger a shift in all measurements (mean model) or a shift in the variance between measurements (residual dispersion model) performed on a given weather station by the same amount). Setting `"spatial"=TRUE` (default) implies that the random realizations of the Gaussian process follow a Matern correlation structure. Put simply, this implies that the closer two locations are, the more similar the means (or the log residual variance) in isotopic values are (e.g. because they are likely to be traversed by the same air masses).

The arguments `uncorr.terms` allow the choice between two alternative parameterizations for the uncorrelated random effect in the fits: `"lambda"` or `"nugget"` for each model. When using `"lambda"`, the variance of the uncorrelated random terms is classically modeled by a variance. When a spatial random effect is considered, one can alternatively choose `"nugget"`, which modifies the Matern correlation value when distance between location tends to zero. If no random effect is considered, one should stick to the default setting and it will be ignored by the function. The choice of the parametrization is a matter of personal preferences and it does not change the underlying models, so the estimations for all the other parameters of the models will not be impacted by whether one chooses lambda or nugget. Depending on the data one parametrization may lead to faster fit than the other.

The argument `spaMM.method` is also a list of two *strings* where the first element defines the spaMM functions used for fitting the mean model and the second element defines the spaMM method used for fitting the residual dispersion model. The possible options are "HLfit", "corrHLfit" and "fitme". Note that "HLfit" shall only be used in the absence of a Matern correlation structure and "corrHLfit" shall only be used in the presence of it. In contrast, "fitme" should work in all situations. Which method is best remains to be determined and it is good practice to try different methods (if applicable) to check for the robustness of the results. If all is well one should obtain very similar results with the different methods. If this is not the case, carefully check the model output to see if one model fit did not get stuck at a local minimum during optimization (which would translate in a lower likelihood).

The argument `dist.method` allows modifying how the distance between locations is computed to estimate the spatial correlation structure. By default, we consider the so-called "Earth" distances which are technically called orthodromic distances. They account for earth curvature. The alternative "Euclidean" distances do not. For studies performed on a small geographic scale, both distance methods should lead to similar results.

The arguments `control.mean` and `control.dist` are lists that are transmitted to the **spaMM** fitting functions (defined by spaMM.method). These lists can be used to finely control the fitting procedure, so advanced knowledge of the package **spaMM** is required before messing around with these inputs.

We highly recommend users to examine the output produced by `Isofit`. Sometimes, poor fit may occur and such models should therefore not be used for building isoscapes or performing assignments.

Value

This function returns a *list* of class isofit containing two inter-related fits: `mean.fit` and `disp.fit`. The returned *list* also contains the object `info.fit` that contains all the call arguments.

Note

There is no reason to restrict mean.fit and disp.fit to using the same parametrization for fixed and random effects.

Never use a mean.fit object to draw predictions without considering a disp.fit object: mean.fit is not fitted independently from disp.fit.

For all methods, fixed effects are being estimated by Maximum Likelihood (ML) and dispersion parameters (i.e. random effects and Matern correlation parameters) are estimated by Restricted Maximum Likelihood (REML). Using REML provides more accurate prediction intervals but impedes the accuracy of Likelihood Ratio Tests (LRT). Our choice for REML was motivated by the fact that our package is more likely to be used for drawing inferences than null hypothesis testing. Users interested in model comparisons may rely on the AIC values that can be extracted from fitted models using the function AIC from the **spaMM**.

Variable names for iso.data must be respected to ensure a correct utilization of this package. Alteration to the fixed effect structure is not implemented so far (beyond the different options proposed) to avoid misuse of the package. Users that would require more flexibility should consider using spaMM directly (at their own risks). We will soon provide a document explaining how to do so.

Source

<http://kimura.univ-montp2.fr/~rousset/spaMM.htm>

References

Rousset, F., Ferdy, J. B. (2014). Testing environmental and genetic effects in the presence of spatial autocorrelation. Ecography, 37(8):781-790.

Bowen, G. J., Wassenaar, L. I., Hobson, K. A. (2005). Global application of stable hydrogen and oxygen isotopes to wildlife forensics. Oecologia, 143(3):337-348.

See Also

spaMM for an overview of the **spaMM** package

fitme and corrHLfit for information about the two possible fitting procedures that can be used here

Matern.corr for information about the Matern correlation structure

IsoriX for the complete work-flow of our package

Examples

```
## We fit the isoscape model (on the first 50 rows only,
## otherwise it takes too long for a simple example)

data(GNIP_Europe)

isoscapemodel1 <- Isofit(iso.data=GNIP_Europe[1:50, ],
    mean.model.fix=list(elev=TRUE, lat.abs=TRUE))
```

```
isoscapemodel1

## Diagnostics for the fits
plot(isoscapemodel1)


## The following example takes some time and will therefore not
## be run unless you type: example(Isofit, run.dontrun=TRUE)

## Not run:
## Model with one uncorrelated random effect using the Lambda
## parametrization for the fitting the mean model and with one
## uncorrelated random effect using the Nugget parametrization
## for the fitting the residual dispersion model:

isoscapemodel2 <- Isofit(iso.data=GNIP_Europe[1:50, ],
    mean.model.fix=list(elev=TRUE, lat.abs=TRUE),
    mean.model.rand=list(uncorr=TRUE),
    disp.model.rand=list(uncorr=TRUE),
    uncorr.terms=list(disp.model="nugget"))

isoscapemodel2

## End(Not run)
```

Isorix   *Infer spatial origins*

Description

This function performs the assignment of organims of unknown origins.

Usage

```
Isorix(
  assign.data,
  isoscape,
  calibfit,
  mask=NULL,
  verbose=interactive()
)
```

Arguments

assign.data   A *dataframe* containing the assignment data (see note below)

isoscape   The output of the function `Isoscape`

calibfit   The output of the function `Calibfit`

mask	A *SpatialPolygons* of a mask to replace values on all rasters by NA inside polygons (see details)
verbose	A *logical* indicating whether information about the progress of the procedure should be displayed or not while the function is running. By default verbose is *TRUE* if users use an interactive R session and *FALSE* otherwise.

Details

An assignment is a comparison, for a given organism, of the predicted isotopic value at its location of origin and the predicted isotopic value at each location of the isoscape. The difference between these two values constitute the statistic of the assingment test. Under the null hypothesis (when the organism is at a location with the same isotopic value than its original location), the test statistics follows a normal distribution with mean zero and a certain variance that stems from both the isoscape model fits and the calibration fit. The function Isorix computes the map of p-value for such assignment test (i.e. the p-values in all locations of the isoscape) for all individual in the dataframe assign.data. The function also perform a single assignment for the entire group by combining the individual p-value maps using the Fisher's method (Fisher 1925).

A mask can be used so to remove all values falling in the mask. This can be useful for performing for example assignments on lands only and discarded anything falling in large bodies of water (see example).

Value

This function returns a *list* of class *isorix* containing itself three lists (indiv, group, and sp.points) storing all rasters built during assignment and the spatial points for sources and calibration. The *list* indiv contains three stack of rater layers: one storing the value of the test statistic ("stat"), one storing the value of the variance of the test statistic ("var") and one storing the p-value of the test ("pv"). The *list* group contains one rater storing the p-values of the assignment for the group. The *list* sp.points contains two spatial point objects: source.points and calib.points.

Note

See assignmentdata to know which variables are needed to perform the assignment and their names.

References

Fisher, R.A. (1925). Statistical Methods for Research Workers. Oliver and Boyd (Edinburgh). ISBN 0-05-002170-2.

See Also

IsoriX for the complete workflow

Examples

```
## Not run:
## The following example takes some time and will therefore not
## be run unless you type: example(Isorix, run.dontrun=TRUE)
```

```
data(calibrationdata)
data(assignmentdata)
data(elevationrastersmall)
data(worldcountries)
data(oceanmask)
data(GNIP_Model)

## prepare the elevation raster
elevationraster <- RElevate(
    elevation.raster=elevationrastersmall,
    isofit=GNIP_Model)

## build the isoscape
isoscape <- Isoscape(
    elevation.raster=elevationraster,
    isofit=GNIP_Model)

## fit the calibration model
calib <- Calibfit(
    calib.data=calibrationdata,
    isofit=GNIP_Model)


## perform the assignment on land and water
assignment <- Isorix(
    assign.data=subset(assignmentdata,
      species=="Myotis_bechsteinii"),
    isoscape=isoscape,
    calibfit=calib)

assignment


## perform the assignment on land only
assignment.dry <- Isorix(
    assign.data=subset(assignmentdata,
      species=="Myotis_bechsteinii"),
    isoscape=isoscape,
    calibfit=calib,
    mask=oceanmask)

## plot the group assignment
plot(assignment, who="group") ## without decoration

plot(assignment.dry, who="group") ## without decoration

plot(assignment.dry, who="group",
  borders=list(borders=worldcountries),
  mask=list(mask=oceanmask))

## plot the assignment for the 4 first individuals
plot(assignment.dry, who=1:4,
```

```
  borders=list(borders=worldcountries),
  mask=list(mask=oceanmask),
  sources=list(draw=FALSE),
  calib=  list(draw=FALSE))

## plot the assignment for the individual "Mbe_8"
plot(assignment.dry, who="Mbe_8",
  borders=list(borders=worldcountries),
  mask=list(mask=oceanmask))


## End(Not run)
```

---

isorixpalette   *Colour palettes for plotting*

Description

These datasets contain colour vectors that can be used for plotting. In our examples, we use the isoscapepalette for plotting the isoscape using plot.isoscape and isorixpalette for plotting the assignment outcoume using plot.isorix.

Usage

```
data(isoscapepalette)
data(isorixpalette)
```

Format

A vector of colours

Details

Colour palettes can be created by using the function colorRamp that interpolates colours between a set of given colours. One can also use colorRampPalette to create functions providing colours. Also interesting, the function colorspace::choose_palette offers a GUI interface allowing to create and save a palette in a hexadecimal format (which can later on be imported into R). This latter function is however limited to a maximum of 50 colours. You can also use R colour palettes already available such as terrain.colors or others available (see examples below). Alternatively, you can design your own colour palette by writing standard hexadecimal code of colours into a vector.

Note

We use the package **rasterVis** for plotting. Instead of using colour palettes directly, one can also use any "Theme" designed for the lattice graphic environment (see source for details).

Source

For information on how to use themes, check:

https://oscarperpinan.github.io/rastervis/#themes

See Also

rainbow for information about R colour palettes

colorRamp and colorspace::choose_palette to create your own palettes

Examples

```
### A COMPARISON OF SOME COLOUR PALETTES
data(isoscapepalette)
data(isorixpalette)

par(mfrow=c(2, 3))
pie(rep(1, length(isoscapepalette)), col = isoscapepalette,
border=NA, labels=NA, clockwise=TRUE, main="isoscapepalette")
pie(rep(1, length(isorixpalette)), col = isorixpalette,
border=NA, labels=NA, clockwise=TRUE, main="isorixpalette")
pie(rep(1, 100), col = terrain.colors(100), border=NA, labels=NA,
    clockwise=TRUE, main="terrain.colors")
pie(rep(1, 100), col = rainbow(100), border=NA, labels=NA,
    clockwise=TRUE, main="rainbow")
pie(rep(1, 100), col = topo.colors(100), border=NA, labels=NA,
    clockwise=TRUE, main="topo.colors")
pie(rep(1, 100), col = heat.colors(100), border=NA, labels=NA,
    clockwise=TRUE, main="heat.colors")

### CREATING YOUR OWN COLOUR PALETTE
my.palette  <- colorRampPalette(c("blue", "green", "red"), bias=0.7)
par(mfrow=c(1, 1))
pie(1:100, col=my.palette(100), border=NA, labels=NA,
    clockwise=TRUE, main="a home-made palette")
```

Isoscape   *Predicts the spatial distribution of isotopic values*

Description

This function produces the isoscape, i.e. a spatial prediction (i.e. map) of the distribution of isotopic delta values. Predictions are computed using the fitted isoscape for each raster cell of the elevation raster. All shape files can be exported and loaded into any Geographic Information System (GIS) if needed.

Usage

```
Isoscape(
    elevation.raster,
    isofit,
    verbose=interactive()
)
```

Arguments

elevation.raster

        The elevation raster (*RasterLayer*) created by RElevate

isofit   The fitted isoscape created by Isofit

verbose   A *logical* indicating whether information about the progress of the procedure should be displayed or not while the function is running. By default verbose is *TRUE* if users use an interactive R session and *FALSE* otherwise.

Details

This function computes the predictions (mean), prediction variances (mean.predVar), residual variances (mean.residVar) and response variances (mean.respVar) for the isotopic values at a resolution equal to the one of the elevation raster. It also computes the same information for the residual dispersion variance (disp.pred, disp.predVar, disp.residVar, or disp.respVar).

The predictions of isotopic values across the landscape are performed by calling the function predict from the package **spaMM** on the fitted isoscape produced by Isofit.

Let us detail the meaning of mean, mean.predVar and mean.respVar:

Our model assumes that that there is a single true unknown isoscape, which is fixed but which is represented by the mixed-effect model as a random draw from possible realizations of isoscapes (random draws of the Matern-correlated process and of the uncorrelated random effects if considered). We infer this realized isoscape by fitting the model to a limited amount of data, with some uncertainty since different random draws of the unknown isoscape may give the same observed data. There is thus a conditional distribution of possible true isoscapes given the data. For linear mixed-effects models, the mean prediction, technically called the best linear unbiased prediction (BLUP), is the mean of this conditional distribution. The prediction variance is ideally the mean square difference between the true unknown value of the linear predictor and the BLUP at a given location. The response variance has a different meaning. It estimates the variance of new observations drawn from the true unknown isoscape at a given location. The response variance is simply equal to the sum of the prediction variance and the residual variance (note that the residual variance considered assume that a single observation is being observed per location).

The isoscape can be plotted using the function plot.isoscape (see examples).

Value

This function returns a *list* of class *isoscape* containing a stack of all 8 raster layers mentioned above (all being of class *RasterLayer*), and the location of the sources as spatial points.

See Also

Isofit for the function fitting the isoscape

plot.isoscape for the function plotting the isoscape model

IsoriX for the complete work-flow

Examples

```
## The following display takes a few seconds too many and will therefore not
## be run unless you type: example(Isoscape, run.dontrun=TRUE)

## Not run:

data(elevationrastersmall)
data(GNIP_Model)
data(worldcountries)
data(oceanmask)
data(isoscapepalette)

## We crop the elevation raster to the extent of GNIP_Model
elevationraster <- RElevate(
    elevation.raster=elevationrastersmall,
    isofit=GNIP_Model)

## We build the isoscape
isoscape <- Isoscape(
    elevation.raster=elevationraster,
    isofit=GNIP_Model)

isoscape

plot.mean <- plot(
    x=isoscape,
    which="mean",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette,
    plot=FALSE)

plot.mean.predVar <- plot(
    x=isoscape,
    which="mean.predVar",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette,
    plot=FALSE)

plot.mean.residVar <- plot(
    x=isoscape,
    which="mean.residVar",
    borders=list(borders=worldcountries),
```

```
        mask=list(mask=oceanmask),
        palette=isoscapepalette,
        plot=FALSE)

plot.mean.respVar <- plot(
    x=isoscape,
    which="mean.respVar",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette,
    plot=FALSE)

if(require(lattice)) {
    print(plot.mean,          split=c(1, 1, 2, 2), more=TRUE)
    print(plot.mean.predVar,  split=c(2, 1, 2, 2), more=TRUE)
    print(plot.mean.residVar, split=c(1, 2, 2, 2), more=TRUE)
    print(plot.mean.respVar,  split=c(2, 2, 2, 2), more=FALSE)
}

plot.disp <- plot(
    x=isoscape,
    which="disp",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette,
    plot=FALSE)

plot.disp.predVar <- plot(
    x=isoscape,
    which="disp.predVar",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette,
    plot=FALSE)

plot.disp.residVar <- plot(
    x=isoscape,
    which="disp.residVar",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette,
    plot=FALSE)

plot.disp.respVar <- plot(
    x=isoscape,
    which="disp.respVar",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette,
    plot=FALSE)

if(require(lattice)) {
    print(plot.disp, split=c(1, 1, 2, 2), more=TRUE)
```

```
    print(plot.disp.predVar,   split=c(2, 1, 2, 2), more=TRUE)
    print(plot.disp.residVar,  split=c(1, 2, 2, 2), more=TRUE)
    print(plot.disp.respVar,   split=c(2, 2, 2, 2), more=FALSE)
}


## End(Not run)
```

Isosim   *Simulate isotopic values*

Description

This function allows for the simulation of isoscapes. Both partial or complete (i.e. maps) isoscapes can be simulated.

Usage

```
Isosim(
    simu.data,
    mean.model.fix.coef=c(intercept=64, elev=-0.01,
        lat.abs=-2.3, lat.2=0, long=0, long.2=0),
    disp.model.fix.coef=c(intercept=5.8, elev=0,
        lat.abs=0, lat.2=0, long=0, long.2=0),
    mean.model.matern.coef=c(nu=0.35, rho=5e-5, lambda=899),
    disp.model.matern.coef=c(nu=3.2e-01, rho=1.5e-05, lambda=5),
    mean.model.uncorr.coef=c(nugget=0, lambda=0),
    disp.model.uncorr.coef=c(nugget=0, lambda=0),
    dist.method="Earth",
    seed=NULL,
    save.dataframe=FALSE,
    verbose=interactive()
)
```

Arguments

simu.data   A *data.frame* containing the covariates needed for the simulation, or alterna-
                     tively an elevation raster of class *RasterLayer*

mean.model.fix.coef
                     A *vector* of coefficients for fixed-effects

disp.model.fix.coef
                     A *vector* of coefficients for fixed-effects

mean.model.matern.coef
                     A *vector* of coefficients for the spatial random effect

disp.model.matern.coef
                     A *vector* of coefficients for the spatial random effect

```
mean.model.uncorr.coef
```
A *vector* of coefficients for the uncorrelated random effect
```
disp.model.uncorr.coef
```
A *vector* of coefficients for the uncorrelated random effect

`dist.method`   A *string* indicating the distance method

`seed`   An *integer* used to set the seed of the random generator

`save.dataframe` A *logical* indicating whether the detailed *data.frame* containing the simulated data should be saved

`verbose`   A *logical* indicating whether information about the progress of the procedure should be displayed or not while the function is running. By default verbose is *TRUE* if users use an interactive R session and *FALSE* otherwise.

Details

This function takes as inputs the values for all covariates matching a series of locations (which can be provided as an elevation raster or as a *data.frame*), as well as the parameters of the isoscape model. The function is not required to fit an isoscape, nor to perform assignments. It is an additional function that can be useful to test the method, and to study the effect of different parameters on isoscapes. We chose default values for parameters inspired to model fits in order to simulate a relatively realistic isoscape. We precised 'relatively' because, in the simulations, the Matern realizations for the mean and the dispersion are drawn independently, which is not the case in nature. Note that extra parameters present in the input lists will not make the function crash but won't be considered during computations either.

Value

This function returns a *list* of class *isoscape* containing a stack of raster and an optional *data.frame*. The stack contains the raster `mean.raster` storing the mean isotopic value, and the raster `disp.raster` storing the residual dispersion variance. The optional *data.frame* contains the simulated data and details of the computation in an object called `data`.

Note

The spatial autocorrelation and the Nugget are computed by the functions [RMwhittle](#) and [RMnugget](#), respectively. These two functions are part of the powerful package **[RandomFields](#)**.

See Also

[Isofit](#) for the function fitting the isoscape model [IsoriX](#) for the complete work-flow

Examples

```
## This example takes couple of minutes and will therefore not be run
## unless you type: example(Isosim, run.dontrun=TRUE)

## Not run:
## We load an elevation raster
data(elevationrastersmall)
```

```
data(worldcountries)
data(oceanmask)
data(isoscapepalette)

elevationraster <- RElevate(elevationrastersmall,
    manual.crop=c(-30, 60, 30, 70))

## We simulate data under default settings
simu <- Isosim(simu.data=elevationraster,
    save.dataframe=TRUE, seed=2)

simu

## We build the plots of the outcome using IsoriX
plot.mean.simu <- plot(
    x=simu,
    which="mean",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette)

plot.disp.simu <- plot(
    x=simu,
    which="disp",
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette)


## We fit the simulated data by sampling 200 locations

set.seed(123)
newdat <- simu$data[sample(1:nrow(simu$data), 200), ]

isoscapemodel <- Isofit(iso.data=newdat,
    mean.model.fix=list(elev=TRUE, lat.abs=TRUE))

isoscape <- Isoscape(elevationraster, isoscapemodel)

plot.mean.fitted <- plot(
    x=isoscape,
    which="mean",
    sources=list(draw=FALSE),
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
    palette=isoscapepalette)

plot.disp.fitted <- plot(
    x=isoscape,
    which="disp",
    sources=list(draw=FALSE),
    borders=list(borders=worldcountries),
    mask=list(mask=oceanmask),
```

```
      palette=isoscapepalette)

## We compare simulated and fitted data visually
if(require(lattice)){
    print(plot.mean.simu, split=c(1, 1, 2, 2), more=TRUE)
    print(plot.disp.simu, split=c(2, 1, 2, 2), more=TRUE)
    print(plot.mean.fitted, split=c(1, 2, 2, 2), more=TRUE)
    print(plot.disp.fitted, split=c(2, 2, 2, 2), more=FALSE)
}

## We compare simulated and fitted data numerically
## Note that differences are expected, as the geographic
##   area is small compared to the scale at which
##   spatial auto-correlation occurs
round(cbind(
    simulated=c(
        intercept=64,
        lat.abs=-2.3,
        elev=-0.01,
        nu=0.35,
        rho=5e-5,
        rho_div_nu = 5e-5/0.35,
        lambda.matern=899,
        intercept.disp=5.8,
        nu.disp=3.2e-01,
        rho.disp=1.5e-05,
        lambda.matern.disp=5),
    fitted=c(
        intercept=isoscapemodel$mean.fit$fixef[1],
        lat.abs=isoscapemodel$mean.fit$fixef[2],
        elev=isoscapemodel$mean.fit$fixef[3],
        nu=isoscapemodel$mean.fit$ranFix$nu,
        rho=isoscapemodel$mean.fit$ranFix$rho,
        rho_div_nu=with(isoscapemodel$mean.fit$ranFix,rho/nu),
        lambda.matern=isoscapemodel$mean.fit$lambda,
        intercept.disp=isoscapemodel$disp.fit$fixef[1],
        nu.disp=isoscapemodel$disp.fit$ranFix$nu,
        rho.disp=isoscapemodel$disp.fit$ranFix$rho,
        lambda.matern.disp=isoscapemodel$disp.fit$lambda)), 4)


## End(Not run)
```

---

oceanmask   *Mask of world oceans*

Description

This dataset contains a polygon shapefile that can be used to mask large bodies of water.

Usage

```
data(oceanmask)
```

Format

A *SpatialPolygons* object

Source

This *SpatialPolygons* is derived from the worldcountries. See example for details on how we created the dataset.

See Also

worldcountries for another polygon shapefile used to embellish the plots

Examples

```
data(oceanmask)
if(require(sp))
    plot(oceanmask, col='blue')

## HOW DID WE CREATE THIS FILE?
## (This example takes some time and will therefore not be run
##  unless you type: example(oceanmask, run.dontrun=TRUE) )

## Not run:
if(require(raster) & require(rgeos)){
    worldlimit <- as(extent(worldcountries), "SpatialPolygons")
    proj4string(worldlimit) <- crs(worldcountries)
    oceanmask <- gDifference(worldlimit, worldcountries)
    oceanmask
    ## uncomment the following to store the file:
    #save(oceanmask, file="oceanmask.rda", compress="xz")
}

## End(Not run)
```

---

plot.methods   *Plotting functions for IsoriX*

Description

These functions plot objects created by **IsoriX**.

Usage

```
## S3 method for class 'isofit'
plot(
    x,
    cex.scale=0.2,
    ...
)

## S3 method for class 'calibfit'
plot(
    x,
    ...
)

## S3 method for class 'isoscape'
plot(
    x,
    which="mean",
    sources=list(draw=TRUE, cex=0.5, pch=2, lwd=1, col="red"),
    borders=list(borders=NULL, lwd=0.5, col="black"),
    mask=   list(mask=NULL, col="black"),
    palette=terrain.colors(20),
    plot=TRUE,
    ...
)

## S3 method for class 'isorix'
plot(
    x,
    who="group",
    what="pv",
    cutoff= list(draw=TRUE, level=0.05, col="#909090"),
    sources=list(draw=TRUE, cex=0.5, pch=2, lwd=1, col="red"),
    calib=  list(draw=TRUE, cex=0.5, pch=4, lwd=1, col="blue"),
    borders=list(borders=NULL, lwd=0.5, col="black"),
    mask=   list(mask=NULL, col="black"),
    palette=rev(terrain.colors(100)),
    plot=TRUE,
    ...
)
```

Arguments

x	The return object of an Isofit, Isoscape, Calibfit, or Isorix call
cex.scale	a *numeric* giving a scalling factor for the points in the plots
which	A *string* indicating the name of the raster to be plotted (see details)

who	Either "group", or a vector of indices (e.g. 1:3) or names of the individuals (e.g. c("Mbe_1", "Mbe_3")) to be considered in assignment plots
what	A *string* indicating the name of the raster to be plotted (should remain "pv" if who="group", otherwise could be "stat", "stat.var", or "pv")
cutoff	A *list* containing information for the display of the region outside the prediction interval (see details)
sources	A *list* containing information for the display of the location of the sources (see details)
calib	A *list* containing information for the display of the location of the calibration sampling area (see details)
borders	A *list* containing information for the display of borders (e.g. country borders) (see details)
mask	A *list* containing information for the display of a mask (e.g. an ocean mask) (see details)
palette	A vector of colours (*character*)
plot	A *logical* indicating whether the plot shall be plotted or just returned
...	Additional arguments (not in use)

Details

When called upon an object of class isofit, the plot function draws diagnostic information for the fits of the isoscape geostatistical model.

When called upon an object of class calibfit, the plot function draws the fitted calibration function.

When called upon an object of class isoscape, the plot function draws a fine-tuned plot of the isoscape. When used on a fitted isoscape, the user can choose between plotting the predictions (which = "mean"; default), the prediction variance (which = "mean.predVar"), the residual variance (which = "mean.residVar"), or the response variance (which = "mean.respVar") for the mean model; or the corresponding information for the residual dispersion variance model ("disp", "disp.predVar", "disp.residVar", or "disp.respVar"). When used on a simulated isoscape, the user can choose between plotting the mean isotopic value (which = "mean") or the dispersion (which = "disp").

When called upon an object of class isorix, the plot function draws a fine-tuned plot of the assignment. You can use the argument "who" to choose between plotting the assignment for the group or for some individuals (check the vignette "Workflow" for examples).

The arguments "cutoff", "sources", "calib", "borders" and "mask" are used to fine-tune additional layers that can be added to the main plot to embellish it. These arguments must be lists that provide details on how to draw, respectively, the area outside the prediction interval (for assignment plots), the locations of sources (for both isoscape and assignment plots), the locations of the calibration sampling area (for assignment plots, the borders (for both types of plots) and a mask (again, for both)). Within these lists, the element "col", "cex", "pch" and "lwd" influences their respective objects as in traditional R plotting functions (see [par](#) for details). The element "draw" should be a *logical* that indicates whether the layer must be created or not. The argument "borders" (within the list borders) expects an object of the class *SpatialPolygons* such as the object "worldcountries" provided with this package. The argument "mask" (within the list maks) expects an object of the class *SpatialPolygons* such as the object oceanmask provided with this package (see examples).

See Also

Isofit for the function fitting the isoscape

Isoscape for the function building the isoscape

Calibfit for the function fitting the calibration function

Isorix for the function performing the assignment

IsoriX for the complete work-flow

Examples

```
## See ?Isoscape or ?Isorix for examples
```

---

print.isofit   *Print and summary methods for IsoriX*

Description

These functions print summary information of objects of class *isofit*, *isoscape* or *isorix*.

Usage

```
## S3 method for class 'isofit'
print(x, ...)
## S3 method for class 'isoscape'
print(x, ...)
## S3 method for class 'calibfit'
print(x, ...)
## S3 method for class 'isorix'
print(x, ...)

## S3 method for class 'isofit'
summary(object, ...)
## S3 method for class 'isoscape'
summary(object, ...)
## S3 method for class 'calibfit'
summary(object, ...)
## S3 method for class 'isorix'
summary(object, ...)
```

Arguments

object	The return object of an Isofit, Isoscape, Calibfit, or Isorix call
x	The return object of an Isofit, Isoscape, Calibfit, or Isorix call
...	Additional arguments

Value

These functions return nothing

See Also

[Isofit](#) for the function: fitting the isoscape

[Isoscape](#) for the function: building the isoscape

[Calibfit](#) for the function: fitting the calibration function

[Isorix](#) for the function: making the assignment

[IsoriX](#) for the complete workflow

QueryGNIP   *Filter the dataset to create an isoscape*

Description

This function prepares the worldwide GNIP data (*GNIP_World*) to be used for creating the isoscape. This function allows the trimming of data by months, years and location, and for the aggregation of selected data per weather station. The function can also be used to randomly exclude some observations.

Usage

```
QueryGNIP(
    data,
    month.min = 1,
    month.max = 12,
    year.min,
    year.max,
    long.min ,
    long.max,
    lat.min,
    lat.max,
    prop.random = 0,
    random.level = "station"
    )
```

Arguments

data	A *dataframe* containing original isotopic measurements similar in structure to [GNIP_World](#)
month.min	A *numeric* indicating the minimum month to select from. Should be a round number between 1 and 12. The default value is 1 (January).
month.max	A *numeric* indicating the maximum month to select from. Should be a round number between 1 and 12. The default value is 12 (December).

year.min	A *numeric* indicating the oldest year to select from. If not provided, the oldest year of GNIP_World will be considered
year.max	A *numeric* indicating the most recent year to select from. If not provided, the most recent year of [GNIP_World](#) will be considered
long.min	A *numeric* indicating the minimum longitude to select from. Should be a number between -180 and 180. If not provided, -180 will be considered.
long.max	A *numeric* indicating the maximal longitude to select from. Should be a number between -180 and 180. If not provided, 180 will be considered.
lat.min	A *numeric* indicating the minimum latitude to select from. Should be a number between -90 and 90. If not provided, -90 will be considered.
lat.max	A *numeric* indicating the maximal latitude to select from. Should be a number between -90 and 90. If not provided, 90 will be considered.
prop.random	A *numeric* indicating the proportion of observations or weather stations (depending on the argument for random.level) that will be kept. If prop.random is greater than 0, then the function will return a list containing two dataframes: one containing the selected data, called selected.data, and one containing the remaining data, called remaining.data.
random.level	A *string* indicating the level at which random draws can be performed. The two possibilities are "obs", which indicates that observations are randomly drawn taken independently of their location, or "station" (default), which indicates that observations are randomly drawn at the level of weather stations.

Details

This functions allows the user to filter the world-wide weather station data (*GNIP_World*) based on time (years and/ or months) and space (locations given in geographic coordinates, i.e. longitude and latitude) to calculate tailored isoscapes matching e.g. the time of sampling and speeding up the model fit by cropping/ clipping a certain area. The dataframe produced by this function can be used as input to fit the isoscape (see [Isofit](#)).

Value

This function returns a *dataframe* containing the filtered data aggregated by weather station, or a *list*, see above argument prop.random. For each weather station the mean and variance sample estimates are computed.

Note

If one wants e.g. to select the winter months November to February, one has to select 11 as minimum and 2 as maximum and not the opposite.

See Also

[IsoriX](#) for the complete work-flow [GNIP_World](#) for the complete dataset [GNIP_Europe](#) for an example of a dataset produced by this function

Examples

```
data(GNIP_World)

### HOW DID WE CREATE THE OBJECT GNIP_Europe?

## The following example require to have downloaded
## a large elevation raster with the function GetElev()
## and will therefore not run unless you type:
## example(QueryGNIP, run.dontrun=TRUE)

## we first correct the elevations in GNIP_World:
## Not run:
if(require(raster)) {
    elevationrasterbig <- raster("elevation_raster_gmted2010_30mn.tif")
          GNIP_World$elev <- extract(
              elevationrasterbig,
              cbind(GNIP_World$long, GNIP_World$lat))
    }

## End(Not run)

## then we used our function:
GNIP_Europe <- QueryGNIP(
    data=GNIP_World,
    long.min = -27.34,
    long.max = 57.1,
    lat.min = 31.62,
    lat.max = 68.97)
head(GNIP_Europe)
## uncomment the following to store the file:
# save(GNIP_Europe, file="GNIP_Europe.rda", compress="xz")


### CREATE ISOSCAPE-DATASET FOR WARM MONTHS IN EUROPE
GNIP_Europe_Warm <- QueryGNIP(
    data=GNIP_World,
    month.min = 5,
    month.max = 8,
    year.min =  1960,
    year.max =  2013,
    long.min = -27.34,
    long.max = 57.1,
    lat.min = 31.62,
    lat.max = 68.97)

head(GNIP_Europe_Warm)


### CREATE A DATASET WITH 90% OF OBS
GNIP_World_90pct <- QueryGNIP(
    data=GNIP_World,
    prop.random=0.9,
```

```
    random.level = "obs")

lapply(GNIP_World_90pct, head) # show beginnng of both datasets

### CREATE A DATASET WITH HALF THE WEATHER STATIONS
GNIP_World_50pctStations <- QueryGNIP(
    data=GNIP_World,
    prop.random=0.5,
    random.level = "station")

lapply(GNIP_World_50pctStations, head)
```

RElevate   *Prepare the elevation raster*

Description

This function prepares the elevation raster for the follow-up analyses. The size and extent of the elevation raster defines the resolution at which the isoscape and the origin assignment are defined.

Usage

```
RElevate(
    elevation.raster,
    isofit=NULL,
    aggregation.factor=0L,
    aggregation.fun=mean,
    manual.crop=NULL,
    verbose=interactive()
)
```

Arguments

elevation.raster

          The elevation raster (*RasterLayer*)

isofit   The fitted isoscape model returned by the function [Isofit](#)

aggregation.factor

          The number of neighbouring cells (*integer*) to merge during aggregation

aggregation.fun

          The *function* used to aggregate cells

manual.crop   A vector of four coordinates (*numeric*) for manual cropping, e.g. the spatial extent

verbose   A *logical* indicating whether information about the progress of the procedure should be displayed or not while the function is running. By default verbose is *TRUE* if users use an interactive R session, and *FALSE* otherwise.

Details

This functions allows the user to crop an elevation raster according to either the extent of the isoscape or manually. If a fitted isoscape object is provided (see `Isofit`), the function extracts the observed locations of isotopic sources from the model object and crops the elevation raster accordingly. Alternatively, `manual.crop` allows you to crop the elevation raster to a desired extent. If no model and no coordinates for manual cropping are provided, no crop will be performed. Importantly, cropping is recommended as it prevents extrapolations outside the latitude/longitude range of the source data. Predicting outside the range of the source data may lead to highly unreliable predictions.

Aggregation changes the spatial resolution of the raster, making computation faster and using less memory (this can affect the assignment; see note below). An aggregation factor of zero (or one) keeps the resolution constant (default).

Value

The fine-tuned elevation raster of class *RasterLayer*

Note

Aggregating the raster may lead to different results for the assignment, because the elevation of raster cells changes depending on the aggregation function (see example below), which in turn affects model predictions.

See Also

`elevationrastersmall` for information on elevation rasters

`IsoriX` for the complete workflow

Examples

```
### DIFFERENCES IN AGGREGATION

## Loading objects
data(elevationrastersmall)
data(GNIP_Model)
data(worldcountries)
data(oceanmask)

## We aggregate and crop using different settings
elevation.raster1 <- RElevate(
    elevation.raster=elevationrastersmall,
    isofit=GNIP_Model,
    aggregation.factor=0)

elevation.raster2 <- RElevate(
    elevation.raster=elevationrastersmall,
    isofit=GNIP_Model,
    aggregation.factor=5)
```

```
elevation.raster3 <- RElevate(
    elevation.raster=elevationrastersmall,
    isofit=GNIP_Model,
    aggregation.factor=5, aggregation.fun=max)


## The following plot takes a few second too many and will
## therefore not be run unless you type:
## example(RElevate, run.dontrun=TRUE)

## Not run:
    ## We plot the outcome of the 3 different aggregation schemes
if(require(rasterVis)) {
    data(worldcountries)
    data(oceanmask)
    plot.aggregation1 <- levelplot(elevation.raster1,
            margin=FALSE, main="Original small raster") +
        layer(sp.polygons(worldcountries)) +
        layer(sp.polygons(oceanmask, fill="blue"))
    plot.aggregation2 <- levelplot(elevation.raster2,
            margin=FALSE, main="Small raster aggregated (by mean)") +
        layer(sp.polygons(worldcountries)) +
        layer(sp.polygons(oceanmask, fill="blue"))
    plot.aggregation3 <- levelplot(elevation.raster3,
            margin=FALSE, main="Small raster aggregated (by max)") +
        layer(sp.polygons(worldcountries)) +
        layer(sp.polygons(oceanmask, fill="blue"))
    ## panel using lattice syntax:
    print(plot.aggregation1, split=c(1, 1, 1, 3), more=TRUE)
    print(plot.aggregation2, split=c(1, 2, 1, 3), more=TRUE)
    print(plot.aggregation3, split=c(1, 3, 1, 3))
}

## End(Not run)
```

---

worldcountries	*Borders of world countries*

Description

This dataset contains a polygon shapefile that can be used to plot the borders of countries.

Usage

```
data(worldcountries)
```

Format

A *SpatialPolygons*

Source

This *SpatialPolygons* is derived from the world of the package **maps**. Please refer to this other package for description and sources of this dataset. See example for details on how we created the dataset.

See Also

oceanmask for another polygon shapefile used to embellish the plots

Examples

```
data(worldcountries)

if(require(sp))
  plot(worldcountries, border="red", col="darkgrey")


### HOW DID WE CREATE THIS FILE?
### (This example takes some time and will therefore not be run
###  unless you type: example(worldcountries, run.dontrun=TRUE) )

## Not run:
if(require(maps) & require(maptools) & require(raster) & require(rgeos)){
    worldmap <- map("world", fill=TRUE, plot=FALSE)
    worldcountries <- map2SpatialPolygons(worldmap, IDs=worldmap$names)
    worldcountries <- gBuffer(worldcountries, byid=TRUE, width=0)
    proj4string(worldcountries) <- CRS("+proj=longlat +datum=WGS84")
    worldcountries
    ## uncomment the following to store the file:
    #save(worldcountries, file="worldcountries.rda", compress="xz")
}

## End(Not run)
```

Index