Package 'lulccR'

January 7, 2015

Title Land use change modelling in R

Version 0.0.0.9000

Author Simon Moulds <simonm@riseup.net>

Maintainer Simon Moulds <simonm@riseup.net>

Description Functions for land use change modelling in R

Depends methods, R (>= 3.1.0)

License GPL-3

LazyData true

Imports raster, ROCR,

Suggests caret, gsubfn, Hmisc, knitr, lattice, plyr, RColorBrewer,

VignetteBuilder knitr

Collate 'AgreementBudget.R' 'class-AgreementBudget.R'
'AgreementBudget.plot.R' 'FigureOfMerit.R' 'class-FigureOfMerit.R' 'FigureOfMerit.plot.R'
'class-StatModels.R' 'class-PredictorMaps.R' 'class-ObservedMaps.R' 'class-NeighbMaps.R'
'class-ModelInput.R' 'ModelInput.R' 'NeighbMaps.R'
'ObservedMaps.R' 'class-PerformanceMulti.R' 'PerformanceMulti.R' 'PerformanceMulti.plot.R'
'class-PredictionMulti.R' 'PredictionMulti.R' 'PredictorMaps.R'
'StatModels.R' 'ThreeMapComparison.R' 'allocate.R' 'allow.R'
'allowNeighb.R' 'approxExtrapDemand.R' 'as.data.frame.R'
'calcProb.R' 'class-ThreeMapComparison.R' 'compareAUC.R'
'crossTabulate.R' 'data.R' 'lulccR-package.R' 'partition.R' 'performance.R' 'resample.R' 'total.R'

R topics documented:

`lulccR-package` *lulccR: land use change modelling in R*

Description

lulccR provides a framework for spatially explicit land use change modelling in R. The main goal of the package is to allow users to perform every stage of the modelling process in the same environment. It therefore includes functions to process and explore model input, fit and evaluate predictive models, estimate future demand for different land uses, allocate land use change spatially, calibrate and validate the model, and, finally, visualise model output.

Author(s)

Simon Moulds

`AgreementBudget` *Create an AgreementBudget object*

Description

This function quantifies sources of agreement and disagreement between a reference map for time 1, a reference map for time 2 and a simulated map for time 2 to provide meaningful information about the performance of land use change simulations.

Usage

```
AgreementBudget(x, from, to, ...)
```

Arguments

`x`	a `ThreeMapComparison` object
`from`	numeric (optional). A single value representing a land use category. Results will be restricted to transitions from this category
`to`	numeric (optional). Similar to `from`. If provided with a valid `from` argument the result will be restricted to the transition defined by these two arguments (i.e. `from -> to`)
`...`	additional arguments (none)

Details

The types of agreement and disagreement considered are as follows:

1. Persistence simulated correctly (agreement)
2. Persistence simulated as change (disagreement)
3. Change simulated incorrectly (disagreement)
4. Change simulated correctly (agreement)
5. Change simulated as persistence (disagreement)

Value

an `AgreementBudget` object

Author(s)

Simon Moulds

See Also

`link{ThreeMapComparison}`, `link{FigureOfMerit}`

AgreementBudget-class *Class AgreementBudget*

Description

An S4 class for information about sources of agreement and disagreement between three maps.

Slots

`agreement` data.frame containing sources of agreement and disagreement. Rows represent different resolutions and columns represent the different types of agreement and disagreement

`factors` numeric vector of aggregation factors

`type` character. Either 'overall' (considering all possible transitions), 'category' (all transitions from one category) or 'transition' (specific transition)

`categories` numeric vector of land use categories in the three maps considered

`labels` character vector with labels corresponding to `categories`

`from,to` numeric vectors that together define land use transitions for which agreement was calculated

Author(s)

Simon Moulds

AgreementBudget.plot *Plot method for AgreementBudget objects*

Description

Plot sources of agreement and disagreement between three maps at multiple resolutions

Usage

```
AgreementBudget.plot(x, ...)

## S4 method for signature AgreementBudget
AgreementBudget.plot(x,
  col = RColorBrewer::brewer.pal(5, "Set2"), key, scales, xlab, ylab, ...)
```

Arguments

`x`	object of class `AgreementBudget`
`...`	additional arguments to `lattice::xyplot`
`col`	character. A colour palette. Default is 'Set2' from RColorBrewer
`key`	list. See `lattice::xyplot`
`scales`	list. See `lattice::xyplot`
`xlab`	character or expression. See `lattice::xyplot`
`ylab`	character or expression. See `lattice::xyplot`

Details

The plot layout is based on work presented in Pontius et al. (2011)

Value

a trellis object

Author(s)

Simon Moulds

References

Pontius Jr, R.G., Peethambaram, S., Castella, J.C. (2011). Comparison of three maps at multiple resolutions: a case study of land change simulation in Cho Don District, Vietnam. Annals of the Association of American Geographers 101(1): 45-62.

See Also

AgreementBudget, lattice::xyplot

`allocate` *Allocate land use change spatially Perform spatially explicit allocation of land use change using different methods. Currently the function provides an algorithm based on the CLUE-S model (Verburg et al., 2002) and a novel stochastic procedure.* `allocate` *is designed to be easily extensible. To write a new allocation method (apart from writing the algorithm itself) one simply needs to create a new class inheriting from* `ModelInput` *and an* `allocate` *method for this class.*

Description

Allocate land use change spatially

Perform spatially explicit allocation of land use change using different methods. Currently the function provides an algorithm based on the CLUE-S model (Verburg et al., 2002) and a novel stochastic procedure.

`allocate` is designed to be easily extensible. To write a new allocation method (apart from writing the algorithm itself) one simply needs to create a new class inheriting from `ModelInput` and an `allocate` method for this class.

Usage

```
allocate(input, ...)

## S4 method for signature CluesModelInput
allocate(input, ...)

## S4 method for signature OrderedModelInput
allocate(input, ...)
```

Arguments

input	an object inheriting from class ModelInput
...	additional arguments for specific methods

Author(s)

Simon Moulds

References

Verburg, P.H., Soepboer, W., Veldkamp, A., Limpiada, R., Espaldon, V., Mastura, S.S. (2002). Modeling the spatial dynamics of regional land use: the CLUE-S model. Environmental management, 30(3):391-405.

See Also

link{ModelInput}

---

allow *Implement decision rules for land use change*

Description

Identify legitimate transitions based on land use history and specific transition rules.

Usage

```
allow(x, hist, categories, cd, rules, ...)
```

Arguments

x	numeric vector containing the land use pattern for the current timestep
hist	numeric vector containing land use history (values represent the number of years the cell has contained the current land use category)
categories	numeric vector containing land use categories in the study region
cd	numeric vector indicating the direction of change for each land use category. A value of 1 means demand is increasing (i.e. the number of cells belonging to the category must increase), -1 means decreasing demand and 0 means demand is static
rules	matrix. See details
...	additional arguments (none)

Details

Decision rules are based on those described by Verburg et al. (2002). The `rules` input argument is a square matrix with dimensions equal to the number of land use categories in the study region where rows represent the current land use and columns represent future transitions. The value of each element should represent a rule from the following list:

1. (rule == 0 | rule == 1): this rule concerns specific land use transitions that are allowed (1) or not (0)
2. (rule == -1): this rule prevents transitions unless demand for the present land use category is decreasing
3. (rule == -2): this rule prevents transitions to land use categories with decreasing or static demand. Note that by combining rule 3 and rule 2 it is possible to prevent simultaneous expansion and contraction
4. (rule > 100 & rule < 1000): this rule imposes a time limit (rule-100) on land use transitions, after which land use change is not allowed. Time is taken from `hist`
5. (rule > 1000): this rule imposes a minimum period of time (rule-1000) before land use is allowed to change

`allow` should be called from methods in `allocate`. The output is a matrix with the same dimensions as the matrix used internally by allocation functions to store land use suitability. Thus, by multiplying the two matrices together, disallowed transitions can be removed from the allocation procedure.

Value

a matrix with values of 1 (change allowed) or NA (change not allowed)

Author(s)

Simon Moulds

References

Verburg, P.H., Soepboer, W., Veldkamp, A., Limpiada, R., Espaldon, V., Mastura, S.S. (2002). Modeling the spatial dynamics of regional land use: the CLUE-S model. Environmental management, 30(3):391-405.

Examples

```
obs <- ObservedMaps(x=pie,
                    pattern="lu",
                    categories=c(1,2,3),
                    labels=c("forest","built","other"),
                    t=c(0,6,14))

# create arbitrary land use history raster
hist <- raster(obs@maps[[1]])
vals <- sample(1:10, ncell(hist))
hist <- setValues(hist, vals[which(!is.na(getValues(obs@maps[[1]])))])

# calculate demand and get change direction for first timestep
dmd <- approxExtrapDemand(obs=obs, tout=0:14)
cd <- dmd[2,] - dmd[1,]
```

```
# create rules matrix, only allowing forest to change if the cell has
# belonged to forest for more than 8 years
rules <- matrix(data=c(1,1008,1008,
                       1,1,1,
                       1,1,1), nrow=3, ncol=3, byrow=TRUE)

allow <- allow(x=obs@maps[[1]],
               hist=hist,
               categories=obs@categories,
               cd=cd,
               rules=rules)

# NB output is only useful when used within an allocation routine
```

`allowNeighb` *Implement neighbourhood decision rules*

Description

Identify legitimate transitions for each cell according to neighbourhood decision rules.

Usage

```
allowNeighb(x, cells, categories, rules, ...)
```

Arguments

`x`	a `NeighbMaps` object
`cells`	index of non-NA cells in the study region
`categories`	numeric vector containing land use categories. If `allowNeighb` is called from an allocation model this argument should contain all categories in the simulation, regardless of whether they're associated with a neighbourhood decision rule
`rules`	a numeric vector with neighbourhood decision rules. Each rule is a value between 0 and 1 representing the threshold neighbourhood value above which change is allowed. Rules should correspond with `neighb@categories`
`...`	additional arguments (none)

Details

See `allow`

Value

a matrix. See `allow`

Examples

```
# observed data
obs <- ObservedMaps(x=pie,
                    pattern="lu",
                    categories=c(1,2,3),
                    labels=c("forest","built","other"),
                    t=c(0,6,14))

# create a NeighbMaps object
nb <- NeighbMaps(x=obs@maps[[1]],
                 categories=1,
                 weights=3,
                 fun=mean)

# index of non-NA cells in study region
na.ix <- which(!is.na(getValues(obs@maps[[1]])))

# only allow change to forest within neighbourhood of current forest cells
# note that rules can be any value between zero (less restrictive) and one
# (more restrictive)
nb.allow <- allowNeighb(x=nb,
                        cells=na.ix,
                        categories=obs@categories,
                        rules=0.5)

# NB output is only useful when used within an allocation routine
```

---

approxExtrapDemand *Extrapolate land use area*

Description

Extrapolate land use area from two observed land maps to provide a valid (although not necessarily realistic) demand scenario.

Usage

```
approxExtrapDemand(obs, tout, ...)
```

Arguments

obs	an ObservedMaps object containing at least two maps
tout	numeric vector specifying the timesteps where interpolation is to take place. Comparable to the xout argument of Hmisc::approxExtrap
...	additional arguments to Hmisc::approxExtrap

Value

a matrix

Author(s)

Simon Moulds

Examples

```
obs <- ObservedMaps(x=pie,
                    pattern="lu",
                    categories=c(1,2,3),
                    labels=c("forest","built","other"),
                    t=c(0,6,14))
dmd1 <- approxExtrapDemand(obs=obs, tout=c(0:14))
```

`as.data.frame` *Coerce* `PredictorMaps` *object to data.frame*

Description

This function extracts data from all rasters in a `PredictorMaps` object for a specified timestep (if dynamic variables are present).

Usage

```
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S4 method for signature PredictorMaps
as.data.frame(x, row.names = NULL,
  optional = FALSE, cells, timestep = 0, ...)
```

Arguments

`x`	an object of class `PredictorMaps`
`row.names`	NULL or a character vector giving the row.names for the data.frame. Missing values are not allowed
`optional`	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional
`...`	additional arguments (none)
`cells`	index of cells to be extracted
`timestep`	numeric indicating the timestep under consideration. Only relevant if x@maps contains dynamic predictor variables

Value

data.frame

Author(s)

Simon Moulds

See Also

`PredictorMaps`, `partition`

`calcProb` *Estimate location suitability*

Description

Estimate location suitability with predictive models.

Usage

```
calcProb(object, ...)

## S4 method for signature glm
calcProb(object, newdata, ...)

## S4 method for signature rpart
calcProb(object, newdata, ...)

## S4 method for signature randomForest
calcProb(object, newdata, ...)

## S4 method for signature StatModels
calcProb(object, newdata, df = FALSE, ...)
```

Arguments

`object`	a `StatModels` object or a model of any class for which a predict method exists (currently, `glm`, `rpart::rpart` and `randomForest::randomForest`
`...`	additional arguments to `predict`
`newdata`	data.frame containing new data
`df`	logical indicating whether the function should return a data.frame or matrix (default)

Value

a data.frame or matrix

Author(s)

Simon Moulds

See Also

`StatModels`, `predict`

compareAUC *Compare the area under the ROC curve (AUC) for different models*

Description

Estimate the AUC for each ROCR::prediction object in a PredictionMulti object.

Usage

```
compareAUC(pred, ...)

## S4 method for signature PredictionMulti
compareAUC(pred, digits = 4, ...)

## S4 method for signature list
compareAUC(pred, digits = 4, ...)
```

Arguments

pred	a PredictionMulti object or a list of these
...	additional arguments (none)
digits	numeric indicating the number of digits to be displayed after the decimal point for AUC values

Details

The user can compare the performance of different statistical models by providing a list of PredictionMulti objects. Note that compareAUC should be used in conjunction with other comparison methods because the AUC does not contain as much information as, for instance, the ROC curve itself

Value

data.frame containing AUC values

Author(s)

Simon Moulds

References

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T. (2005). ROCR: visualizing classifier performance in R. Bioinformatics 21(20):3940-3941.

See Also

PredictionMulti, ROCR::performance

crossTabulate *Cross tabulate land use transitions*

Description

Cross tabulate land use transitions using `raster::crosstab`. This should be the basis for further research into the processes driving the most important transitions in the study region (Pontius et al., 2004).

Usage

```
crossTabulate(x, y, ...)

## S4 method for signature RasterLayer,RasterLayer
crossTabulate(x, y, categories,
  labels = as.character(categories), ...)

## S4 method for signature ObservedMaps,ANY
crossTabulate(x, y, index, ...)
```

Arguments

`x`	RasterLayer representing land use map from an earlier timestep or an ObservedMaps object containing at least two land use maps for different points in time
`y`	RasterLayer representing land use map from a later timestep. Not used if `x` is an ObservedMaps object
`...`	additional arguments to `raster::crosstab`
`categories`	numeric vector containing land use categories to consider. Not used if `x` is an ObservedMaps object
`labels`	character vector (optional) with labels corresponding to `categories`. Not used if `x` is an ObservedMaps object
`index`	numeric vector with index of land use maps from ObservedMaps to use in analysis

Value

a data.frame

Author(s)

Simon Moulds

References

Pontius Jr, R.G., Shusas, E., McEachern, M. (2004). Detecting important categorical land changes while accounting for persistence. Agriculture, Ecosystems & Environment 101(2):251-268.

See Also

`ObservedMaps`, `raster::crosstab`

Examples

```
# RasterLayer input
crossTabulate(x=pie$lu_pie_1985,
              y=pie$lu_pie_1999,
              categories=c(1,2,3),
              labels=c("forest","built","other"))

# ObservedMaps input
obs <- ObservedMaps(x=pie,
                    pattern="lu",
                    categories=c(1,2,3),
                    labels=c("forest","built","other"),
                    t=c(0,6,14))

crossTabulate(x=obs, index=c(1,3))
```

`FigureOfMerit` *Create a FigureOfMerit object*

Description

Calculate the figure of merit at different levels and at different resolutions for a reference map at time 1, a reference map at time 2 and a simulated map at time 2.

Usage

```
FigureOfMerit(x, ...)
```

Arguments

`x`	a `ThreeMapComparison` object
`...`	additional arguments (none)

Details

In land use change modelling terms, the figure of merit is the intersection of observed change and simulated change divided by the union of these, with a range of 0 (perfect disagreement) to 1 (perfect agreement). It is useful to to calculate the figure of merit at three levels: (1) considering all possible transitions from all land use categories, (2) considering all transitions from specific land use categories and (3) considering a specific transition from one land use category to another.

Value

a `FigureOfMerit` object

Author(s)

Simon Moulds

References

Pontius Jr, R.G., Peethambaram, S., Castella, J.C. (2011). Comparison of three maps at multiple resolutions: a case study of land change simulation in Cho Don District, Vietnam. Annals of the Association of American Geographers 101(1): 45-62.

See Also

ThreeMapComparison, AgreementBudget

FigureOfMerit-class *Class FigureOfMerit*

Description

An S4 class for different figure of merit scores.

Slots

`overall` list containing the overall figure of merit score for each resolution

`category` list of numeric vectors containing category specific scores

`transition` list of matrices containing transition specific scores

`factors` numeric vector of aggregation factors

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

Author(s)

Simon Moulds

FigureOfMerit.plot *Plot method for FigureOfMerit objects*

Description

Plot the overall, category-specific or transition-specific figure of merit at different resolutions

Usage

```
FigureOfMerit.plot(x, ...)

## S4 method for signature FigureOfMerit
FigureOfMerit.plot(x, from, to, type = "b", key,
  scales, xlab, ylab, ...)
```

Arguments

x	a FigureOfMerit object
...	additional arguments to xyplot
from	numeric (optional). A single value corresponding to a land use category. If provided without 'to' the figure of merit for all transitions from this category will be plotted
to	numeric (optional). Similar to 'from'. If provided with a valid 'from' argument the figure of merit for the transition defined by these two arguments (i.e. from -> to) will be plotted. It is possible to include more than one category in which case the different transitions will be included on the same plot
type	character. See lattice::xyplot
key	list. See lattice::xyplot
scales	list. See lattice::xyplot
xlab	character or expression. See lattice::xyplot
ylab	character or expression. See lattice::xyplot

Value

a trellis object

Author(s)

Simon Moulds

See Also

FigureOfMerit

ModelInput *Create a ModelInput object*

Description

Methods to combine several object classes that are useful for land use change modelling and perform a series of checks to ensure the objects are compatible in time and space for a model simulation. If they are, a ModelInput object is created which may be supplied to allocate.

Usage

```
ModelInput(x, pred, models, time, demand, ...)

CluesModelInput(x, elas, ...)

OrderedModelInput(x, ...)

## S4 method for signature ModelInput,ANY,ANY,ANY,ANY
ModelInput(x, pred, models, time, demand,
  ...)
```

```
## S4 method for signature
## ObservedMaps,PredictorMaps,StatModels,numeric,matrix
ModelInput(x,
  pred, models, time, demand, hist, mask, neighb = NULL, ...)

## S4 method for signature ModelInput,numeric
CluesModelInput(x, elas, rules = NULL,
  nb.rules = NULL, params, ...)

## S4 method for signature ObservedMaps,numeric
CluesModelInput(x, elas, rules = NULL,
  nb.rules = NULL, params, ...)

## S4 method for signature ModelInput
OrderedModelInput(x, rules = NULL, nb.rules = NULL,
  params, ...)

## S4 method for signature ObservedMaps
OrderedModelInput(x, rules = NULL, nb.rules = NULL,
  params, ...)
```

Arguments

x	an ObservedMaps object or a ModelInput object
pred	a PredictorMaps object
models	a StatModels object
time	numeric vector containing timesteps over which simulation will occur
demand	matrix with demand for each land use category in terms of number of cells to be allocated. The first row should be the number of cells allocated to the initial observed land use map (i.e. the land use map for time 0)
...	additional arguments (none)
elas	numeric indicating elasticity of each land use category to change (only required for CluesModelInput objects
hist	RasterLayer containing land use history (values represent the number of years the cell has contained the current land use category)
mask	RasterLayer containing binary values where 0 indicates cells that are not allowed to change
neighb	an object of class NeighbMaps
rules	matrix with land use change decision rules
nb.rules	numeric with neighbourhood decision rules
params	list with model parameters

Details

The params argument is a list of parameter values. For CluesModelInput it should contain the following components:

jitter.f Parameter controlling the amount of perturbation applied to the probability surface prior to running the CLUE-S iterative algorithm. Higher values result in more perturbation. Default is 0.0001

`scale.f` In CLUE-S the suitability of land use types that do not meet demand is increased by an amount obtained by multiplying this parameter by the difference between allocated and demanded area. Default is 0.0005

`max.iter` The maximum number of iterations in the simulation

`max.diff` The maximum allowed difference between allocated and demanded area of any land use type. Default is 5

`ave.diff` The average allowed difference between allocated and demanded area. Default is 5

When `params` is passed to `OrderedModelInput` it should only contain `max.diff`, which has the same meaning as for `CluesModelInput`.

Value

a ModelInput object

Author(s)

Simon Moulds

See Also

`allocate`

`ModelInput-class` *Class ModelInput*

Description

An S4 class to represent common inputs to land use change models.

Slots

`obs` RasterLayer showing initial land use

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

`pred` a PredictorMaps object

`models` a StatModels object

`time` numeric vector containing timesteps over which simulation will occur

`demand` matrix containing demand scenario or NULL

`hist` RasterLayer showing land use history or NULL

`mask` RasterLayer showing masked areas or NULL

`neighb` NeighbMaps object or NULL

`rules` matrix with land use change decision rules

`nb.rules` numeric with neighbourhood decision rules

`elas` numeric indicating elasticity to change (only required for `CluesModelInput` objects

`params` list with model parameters

Author(s)

Simon Moulds

NeighbMaps *Create a NeighbMaps object*

Description

Methods to calculate neighbourhood values for cells in raster maps using `raster::focal`. By default the fraction of non-NA cells within the moving window (i.e. the size of the weights matrix) devoted to each land use category is calculated. This behaviour can be changed by altering the weights matrix or providing an alternative function. The resulting object can be used as the basis of neighbourhood decision rules or as predictor variables in statistical models.

Usage

```
NeighbMaps(x, categories, weights, neighb, ...)

## S4 method for signature RasterLayer,numeric,list,ANY
NeighbMaps(x, categories, weights,
  neighb, fun = mean, ...)

## S4 method for signature RasterLayer,numeric,numeric,ANY
NeighbMaps(x, categories, weights,
  neighb, fun = mean, ...)

## S4 method for signature RasterLayer,ANY,ANY,NeighbMaps
NeighbMaps(x, categories, weights,
  neighb)
```

Arguments

`x`	RasterLayer with categorical data
`categories`	numeric vector containing land use categories for which neighbourhood values should be calculated
`weights`	list containing a matrix of weights (the 'w' argument in `focal`) for each land use category or a numeric vector specifying the size of each weights matrix. In the latter case only square matrices are possible and all weights are given a value of 1. The order of list or vector elements should correspond to the order of land use categories in `categories`
`neighb`	NeighbMaps object. Only used if `categories` and `weights` are not provided. This option can be useful when existing NeighbMaps objects need to be updated because a new land use map is available, such as during the allocation procedure. In this case `categories` and `weights` are set to `neighb@categories` and `neighb@weights` respectively
`...`	additional arguments to `focal`
`fun`	function. Input argument to `focal`. Default is `mean`

Value

a NeighbMaps object

Author(s)

Simon Moulds

See Also

allowNeighb

Examples

```
## observed data
obs <- ObservedMaps(x=pie,
                    pattern="lu",
                    categories=c(1,2,3),
                    labels=c("forest","built","other"),
                    t=c(0,6,14))

## create a NeighbMaps object for 1985 land use map
nb1 <- NeighbMaps(x=obs@maps[[1]],
                  categories=c(1,2,3), # all land use categories
                  weights=3)           # 3*3 neighbourhood

w1 <- matrix(data=c(1,1,1,
                    1,1,1,
                    1,1,1), nrow=3, ncol=3, byrow=TRUE)

w2 <- matrix(data=c(1,1,1,
                    1,1,1,
                    1,1,1), nrow=3, ncol=3, byrow=TRUE)

w3 <- matrix(data=c(1,1,1,
                    1,1,1,
                    1,1,1), nrow=3, ncol=3, byrow=TRUE)

nb2 <- NeighbMaps(x=obs@maps[[1]],
                  categories=c(1,2,3),
                  weights=list(w1,w2,w3))

## update nb2 for 1991
nb2 <- NeighbMaps(x=obs@maps[[2]],
                  neighb=nb2)
```

NeighbMaps-class *Class NeighbMaps*

Description

An S4 class for neighbourhood maps. Objects contain all the information used in the original call to the constructor function NeighbMaps so they can easily be updated for a new land use map.

Slots

`maps` list of RasterLayers showing neighbourhood values for each land use in `categories`

`weights` list of weights matrices

`fun` function used to calculate neighbourhood values

`focal.args` list of all other arguments supplied to `raster::focal`

`categories` numeric vector of land use categories for which neighbourhood maps were calculated

Author(s)

Simon Moulds

`ObservedMaps` *Create an ObservedMaps object*

Description

Methods to create an ObservedMaps object, which may be created from file, an existing Raster* object or a list of Raster* objects.

Usage

```
ObservedMaps(x, pattern, ...)

## S4 method for signature missing,character
ObservedMaps(x, pattern, ...)

## S4 method for signature character,character
ObservedMaps(x, pattern, ...)

## S4 method for signature list,character
ObservedMaps(x, pattern, ...)

## S4 method for signature RasterLayer,ANY
ObservedMaps(x, pattern, ...)

## S4 method for signature RasterStack,ANY
ObservedMaps(x, pattern, categories, labels, t)
```

Arguments

`x`	path (character), Raster* object or list of Raster* objects. Default behaviour is to search for files in the working directory
`pattern`	regular expression (character). Only filenames (if `x` is a path) or Raster* objects (if `x` is a list) matching the regular expression will be returned. See `raster::raster` for more information about supported filetypes
`...`	additional arguments to `raster::stack`
`categories`	numeric vector of land use categories in observed maps
`labels`	character vector (optional) with labels corresponding to `categories`
`t`	numeric vector containing the timestep of each observed map. The first timestep must be 0

Details

Observed maps should have the same extent and resolution. Note that the study region is defined by the location of non-NA cells in observed maps.

Value

an ObservedMaps object

Author(s)

Simon Moulds

Examples

```
## Plum Islands Ecosystem
obs <- ObservedMaps(x=pie,
                    pattern="lu",
                    categories=c(1,2,3),
                    labels=c("forest","built","other"),
                    t=c(0,6,14))

## Sibuyan Island
obs <- ObservedMaps(x=sibuyan,
                    pattern="lu",
                    categories=c(1,2,3,4,5),
                    labels=c("forest","coconut","grass","rice","other"),
                    t=c(0))
```

ObservedMaps-class *Class ObservedMaps*

Description

An S4 class for observed land use maps.

Slots

`maps` RasterStack containing observed land use maps

`t` numeric vector with timesteps corresponding to each observed map

`total` matrix with number of cells belonging to each land use category in the observed maps. Rows represent different maps and columns represent land land use categories

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

Author(s)

Simon Moulds

partition *Partition raster data*

Description

Divide a categorical raster map into training and testing partitions.

Usage

```
partition(x, size = 0.5, spatial = TRUE, ...)
```

Arguments

x	RasterLayer with categorical data
size	numeric between zero and one indicating the proportion of non-NA cells that should be included in the training partition. Default is 0.5, which results in equally sized partitions
spatial	logical. If TRUE, the function returns a SpatialPoints object with the coordinates of cells in each partition. If FALSE, the cell numbers are returned
...	additional arguments (none)

Value

a list containing the following components:

`train` a SpatialPoints object or numeric vector indicating the cells in the training partition

`test` a SpatialPoints object or numeric vector indicating the cells in the testing partition

Author(s)

Simon Moulds

Examples

```
obs <- ObservedMaps(x=sibuyan,
                    pattern="lu",
                    categories=c(1,2,3,4,5),
                    labels=c("forest","coconut","grass","rice","other"),
                    t=c(0))

## create equally sized training and testing partitions
partition(x=obs@maps[[1]], size=0.5, spatial=FALSE)
```

performance *Create* performance *objects*

Description

A wrapper function for ROCR::performance to create performance objects from a list of prediction objects.

Usage

```
performance(prediction.obj, ...)

## S4 method for signature list
performance(prediction.obj, measure, x.measure = "cutoff",
  ...)
```

Arguments

prediction.obj	a list of prediction objects
...	additional arguments to ROCR::performance
measure	performance measure to use for the evaluation. See ROCR::performance
x.measure	a second performance measure. See ROCR::performance

Value

a list of performance objects

Author(s)

Simon Moulds

References

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T. (2005). ROCR: visualizing classifier performance in R. Bioinformatics 21(20):3940-3941.

See Also

ROCR::prediction, ROCR::performance

PerformanceMulti *Create a PerformanceMulti object*

Description

This function uses different measures to evaluate multiple ROCR::prediction objects stored in a PredictionMulti object.

Usage

```
PerformanceMulti(pred, measure, x.measure = "cutoff", ...)
```

Arguments

pred	an object of class PredictionMulti
measure	performance measure to use for the evaluation. See ROCR::performance
x.measure	a second performance measure. See ROCR::performance
...	additional arguments to performance

Value

a PerformanceMulti object

Author(s)

Simon Moulds

References

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T. (2005). ROCR: visualizing classifier performance in R. Bioinformatics 21(20):3940-3941.

See Also

performance, PredictionMulti

PerformanceMulti-class *Class PerformanceMulti*

Description

An S4 class that extends ROCR::performance-class to hold the results of multiple model evaluations.

Slots

`performance` list of `performance` objects. Each object is calculated for the corresponding `prediction` object held in the `PredictionMulti` object supplied to the constructor function

`auc` numeric vector containing the area under the ROC curve for each performance object

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

Author(s)

Simon Moulds

`PerformanceMulti.plot` *Plot methods for PerformanceMulti objects*

Description

Plot the the ROC curve for each `performance` object in a `PerformanceMulti` object. If a list of `PerformanceMulti` objects is provided, with each list element representing a different type of mathematical model, ROC curves for different models are included on the same plot for model comparison

Usage

```
PerformanceMulti.plot(x, ...)

## S4 method for signature list
PerformanceMulti.plot(x, multipanel = TRUE, type = "l",
  abline = list(c(0, 1), col = "grey"), col = RColorBrewer::brewer.pal(9,
  "Set1"), key.args = list(), ...)
```

Arguments

`x`	either a single `PerformanceMulti` object or a list of these. If a list is provided it must be named.
`...`	additional arguments to `lattice::xyplot`
`multipanel`	logical. If `TRUE`, create a trellis plot where the number of panels equals the number of `PerformanceMulti` objects. Otherwise, create a single plot for each `PerformanceMulti` object
`type`	character. See `lattice::panel.xyplot`
`abline`	list. See `lattice::panel.xyplot`
`col`	character. Plotting colour
`key.args`	list. TODO

Value

a single plot (if `multipanel=FALSE`) or a list of plots.

Author(s)

Simon Moulds

See Also

PerformanceMulti

---

pie *Land use change dataset for Plum Island Ecosystem*

Description

Dataset containing land use maps for X and Y and several predictor variables derived from Pontius and Parmentier (2014).

Usage

```
pie
```

Format

A list containing the following elements:

lu_pie_1985 RasterLayer showing land use in 1985 (forest, built, other)

lu_pie_1991 RasterLayer showing land use in 1991

lu_pie_1999 RasterLayer showing land use in 1999

pred_001 RasterLayer showing elevation

pred_002 RasterLayer showing slope

pred_003 RasterLayer showing distance to built land in 1985

References

Pontius, R.G., and Parmentier, B. (2014). Recommendations for using the relative operating characteristic (ROC). Landscape Ecology doi:10.1007/s10980-013-9984-8.

PredictionMulti *Create a PredictionMulti object*

Description

This function creates a ROCR::prediction object for each statistical model in a StatModels object. It should be used with PerformanceMulti to evaluate multiple models with exactly the same criteria while keeping track of which model corresponds to which land use category. This makes it easier to write functions that compare different model types for the same land use category, such as PerformanceMulti.plot.

Usage

```
PredictionMulti(models, obs, pred, timestep = 0, partition, ...)
```

Arguments

models	a StatModels object
obs	an ObservedMaps object
pred	a PredictorMaps object
timestep	numeric indicating the timestep of the observed map in obs against which the observed map should be tested
partition	character. Either 'train', 'test' or 'none', indicating whether to use the training or testing partition or the full dataset for model evaluation. Default is 'test'
...	additional arguments to ROCR::prediction

Value

a PredictionMulti object

Author(s)

Simon Moulds

References

Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T. (2005). ROCR: visualizing classifier performance in R. Bioinformatics 21(20):3940-3941.

See Also

ROCR::prediction

PredictionMulti-class *Class PredictionMulti*

Description

An S4 class that extends ROCR::prediction-class in order to hold the results of multiple model predictions.

Slots

prediction a list of ROCR::prediction-class objects. These objects are calculated for each statistical model in the StatModels object supplied to the constructor function

categories numeric vector of land use categories for which prediction objects were created

labels character vector with labels corresponding to categories

Author(s)

Simon Moulds

PredictorMaps *Create a PredictorMaps object*

Description

Methods to load maps of predictor variables, which may be created from file, an existing Raster* object or a list of Raster* objects.

Usage

```
PredictorMaps(x, pattern, ...)

## S4 method for signature missing,character
PredictorMaps(x, pattern, ...)

## S4 method for signature character,character
PredictorMaps(x, pattern, ...)

## S4 method for signature RasterStack,character
PredictorMaps(x, pattern, ...)

## S4 method for signature list,character
PredictorMaps(x, pattern, ...)
```

Arguments

x	path (character) to directory containing observed land use maps, a Raster* object or a list of Raster* objects.
pattern	regular expression (character). Only filenames (if x is a path) or Raster* objects (if x is a list) matching the regular expression will be returned. See raster::raster for more information about supported filetypes
...	additional arguments to raster::stack

Details

Predictor maps should follow a naming convention to identify them as static (one map provided for the study period) or dynamic (one map provided for each year of the study period). The name should consist of two (static) or three (dynamic) parts: firstly, the prefix should differentiate predictor maps from other maps in the directory, list or RasterStack. This should be followed by a unique number to differentiate the predictor maps (note that the order of predictor variables in the PredictorMaps object is determined by this value) If the predictor is dynamic this number should be followed by a second number representing the timestep to which the map applies. Dynamic variables should include a map for time 0 (corresponding to the initial observed map) and every subsequent timestep. The different parts should be separated by a period or underscore.

Maps of different predictor variables should have the same coordinate reference system but do not have to have the same extent and resolution as long as the minimum extent is that of the study region defined by an ObservedMaps object. However, maps for different timesteps of the same dynamic predictor variable should have the same extent and resolution because these are stored as RasterStack objects.

Value

a PredictorMaps object

Author(s)

Simon Moulds

Examples

```
## Plum Island Ecosystem
pred.maps <- PredictorMaps(x=pie, pattern="pred")

## Sibuyan
pred.maps <- PredictorMaps(x=sibuyan, pattern="pred")
```

PredictorMaps-class *Class PredictorMaps*

Description

An S4 class for predictor variables.

Slots

`maps` list of RasterStacks. The length of the list corresponds to the number of predictor variables and the number of layers in each RasterStack represents time

`map.names` character vector of the name of each variable in `maps`

`dynamic` logical indicating whether dynamic variables are present

Author(s)

Simon Moulds

---

resample *Resample maps in PredictorMaps object or list*

Description

A wrapper function for `raster::resample` to resample raster objects in a PredictorMaps-class object or list.

Usage

```
resample(x, y, ...)

## S4 method for signature PredictorMaps,Raster
resample(x, y, method = "ngb", ...)

## S4 method for signature list,Raster
resample(x, y, method = "ngb", ...)
```

Arguments

x	a PredictorMaps object or list of Raster* maps to be resampled
y	Raster* object with parameters that x should be resampled to
...	additional arguments to raster::resample
method	method used to compute values for the new RasterLayer, should be "bilinear" for bilinear interpolation, or "ngb" for nearest neighbour

Value

PredictorMaps object or list, depending on x

Author(s)

Simon Moulds

See Also

PredictorMaps

Examples

```
## create PredictorMaps object
pred <- predictorMaps(x=pie, pattern="pred")
pred <- PredictorMaps(maps=pred)

## resample to ensure maps have same characteristics as observed maps
pred <- resample(x=pred, y=pie$lu_pie_1985, method="ngb")
```

---

roundSum *Round elements in matrix or data.frame rows*

Description

Round all numbers in a matrix or data.frame while ensuring that all rows sum to the same value

Usage

```
roundSum(x, ncell)
```

Arguments

x	matrix or data.frame
ncell	numeric specifying the target sum for each row in x

Details

The main application of roundSum is to ensure that each row in the demand matrix specifies exactly the number of cells to be allocated to each land use category for the respective timestep. It may also be used to convert demand from area to number of cells

Value

a matrix

Author(s)

Simon Moulds

`sibuyan` *Land use change dataset for Sibuyan Island*

Description

Dataset containing land use map for 1997 and several predictor variables for Sibuyan Island derived from Verburg et al. (2002). Data are modified by Peter Verburg to demonstrate the CLUE-s model; as such the dataset should not be used for purposes other than demonstration.

Usage

```
sibuyan
```

Format

A list containing the following components:

lu_sib_1997 RasterLayer with land use in 1997 (forest, coconut, grassland, rice, other)

pred_001 RasterLayer showing distance to sea

pred_002 RasterLayer showing mean population density

pred_003 RasterLayer showing occurrence of diorite rock

pred_004 RasterLayer showing occurrence of ultramafic rock

pred_005 RasterLayer showing occurrence of sediments

pred_006 RasterLayer showing areas with no erosion

pred_007 RasterLayer showing areas with moderate erosion

pred_008 RasterLayer showing elevation

pred_009 RasterLayer showing slope

pred_010 RasterLayer showing aspect

pred_011 RasterLayer showing distance to roads in 1997

pred_012 RasterLayer showing distance to urban areas in 1997

pred_013 RasterLayer showing distance to streams

restr1 RasterLayer showing location of current national park

restr2 RasterLayer showing location of proposed national park

demand1 data.frame with demand scenario representing slow growth scenario

demand2 data.frame with demand scenario representing fast growth scenario

demand3 data.frame with demand scenario representing land use change primarily for food production

References

Verburg, P.H., Soepboer, W., Veldkamp, A., Limpiada, R., Espaldon, V., Mastura, S.S (2002). Modeling the Spatial Dynamics of Regional Land Use: The CLUE-S Model. Environmental Management 30(3): 391-405.

`StatModels` *Create a StatModels object*

Description

Methods to create a StatModels-class object.

Usage

```
StatModels(models, obs, categories, labels, ...)

## S4 method for signature list,ObservedMaps,ANY,ANY
StatModels(models, obs, categories,
  labels, ...)

## S4 method for signature list,ANY,numeric,character
StatModels(models, obs, categories,
  labels, ...)
```

Arguments

`models`	a list of mathematical models
`obs`	an ObservedMaps object
`categories`	numeric vector of land use categories in observed maps. Only required if `obs` is not provided
`labels`	character vector (optional) with labels corresponding to `categories`. Only required if `obs` is not provided
`...`	additional arguments (none)

Value

an object of class `StatModels`

Author(s)

Simon Moulds

StatModels-class *Class StatModels*

Description

An S4 class to hold multiple mathematical models for different land use categories belonging to the same map.

Slots

`models` list of mathematical models

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

Author(s)

Simon Moulds

ThreeMapComparison *Evaluate allocation performance with three maps*

Description

An implementation of the method described by Pontius et al. (2011), which compares a reference map at time 1, a reference map at time 2 and a simulated map at time 2 to evaluate allocation performance at multiple resolutions while taking into account persistence. The method quantifies disagreement within coarse squares, disagreement between coarse squares, disagreement about the quantity of land use change and agreement.

Usage

```
ThreeMapComparison(rt1, rt2, st2, categories, labels, factors, ...)
```

Arguments

`rt1`	RasterLayer. Observed land use map at time 1
`rt2`	RasterLayer. Observed land use map at time 2
`st2`	RasterLayer. Simulated land use map at time 2
`categories`	numeric vector of land use categories in observed maps
`labels`	character vector (optional) with labels corresponding to `categories`
`factors`	numeric vector of aggregation factors (equivalent to the 'fact' argument to `raster::aggregate` representing the resolutions at which model performance should be tested
`...`	additional arguments to `raster::aggregate`

Value

a `ThreeMapComparison` object

Author(s)

Simon Moulds

References

Pontius Jr, R.G., Peethambaram, S., Castella, J.C. (2011). Comparison of three maps at multiple resolutions: a case study of land change simulation in Cho Don District, Vietnam. Annals of the Association of American Geographers 101(1): 45-62.

See Also

AgreementBudget, FigureOfMerit

ThreeMapComparison-class *Class ThreeMapComparison*

Description

An S4 class to hold results of a comparison between a reference map for time 1, a reference map for time 2 and a simulation map for time 2 using the the method described by Pontius et al. (2011).

Slots

`tables` list of data.frames that depict the three dimensional table described by Pontius et al. (2011) at different resolutions

`factors` numeric vector of aggregation factors

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

Author(s)

Simon Moulds

---

total *Total number of cells in a categorical Raster* object*

Description

Count the number of cells belonging to each category in a Raster* object.

Usage

```
total(x, categories)
```

Arguments

`x`	Raster* object
`categories`	numeric vector containing land use categories. Only cells belonging to these categories will be counted

Value

A list containing the following components:

`total` a matrix containing the total number of cells belonging to each category. Rows represent layers in the input Raster* object

`categories` the categories included in the calculation

Author(s)

Simon Moulds

Examples

```
# RasterLayer
total(x=sib$lu_sib_1997)

# RasterStack
total(x=stack(pie$lu_pie_1985, pie$lu_pie_1991, pie$lu_pie_1999))
```

Index