Package ‘lulccR’

March 11, 2015

Title Land use change modelling in R

Version 0.1.0

Author Simon Moulds <simonm@riseup.net>

Maintainer Simon Moulds <simonm@riseup.net>

Description Classes and methods for land use change modelling in R

Depends methods,
raster,
R (>= 3.1.0)

License GPL-3

LazyData true

Imports ROCR

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

Collate 'class-ThreeMapComparison.R'
'AgreementBudget.R'
'AgreementBudget.plot.R'
'class-ExpVarMaps.R'
'ExpVarMaps.R'
'FigureOfMerit.R'
'FigureOfMerit.plot.R'
'class-PredModels.R'
'class-ObsLulcMaps.R'
'class-NeighbMaps.R'
'class-ModelInput.R'
'ModelInput.R'
'NeighbMaps.R'
'ObsLulcMaps.R'
'class-Performance.R'
'Performance.R'
'Performance.plot.R'

'PredModels.R'
'class-Prediction.R'
'Prediction.R'
'ThreeMapComparison.R'
'allocate.R'
'allow.R'
'allowNeighb.R'
'approxExtrapDemand.R'
'as.data.frame.R'
'calcProb.R'
'compareAUC.R'
'crossTabulate.R'
'data.R'
'lulccR-package.R'
'partition.R'
'performance.rocr.R'
'resample.R'
'total.R'

R topics documented:

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

Description

The lulccR package is an open and extensible framework for land use change modelling in R.

Details

The aims of the package are as follows:

1. to improve the reproducibility of scientific results and encourage reuse of code within the land use change modelling community
2. to make it easy to directly compare and combine different model structures
3. to allow users to perform every aspect of the modelling process within the same environment

To achieve these aims the package utilises an object-oriented approach based on the S4 system, which provides a formal structure for the modelling framework.

Models are represented by objects inheriting from the superclass `Model`. This class is designed to represent general information required by all models while specific models are represented by its subclasses. Currently the package includes two inductive land use change models: the first is an implementation of the Change in Land Use and its Effects at Small Regional extent (CLUE-S) model (Verburg et al., 2002) (class `CluesModel`), while the second is an ordered procedure based on the algorithm described by Fuchs et al. (2013) but modified to allow stochastic transitions (class `OrderedModel`).

The main input to land use change models is a set of predictive models relating observed land use or land use change to spatially explicit explanatory variables. A predictive model is usually obtained for each category or transition. In lulccR these models are represented by the class `PredModels`. Currently lulccR supports binary logistic regression, provided by base R (`glm`), recursive partitioning and regression trees, provided by package `rpart` and random forest, provided by package `randomForest`. To a large extent the success of the allocation routine depends on the strength of the predictive models: this is one reason why an R package for land use change modelling is attractive.

To validate model output lulccR includes a method developed by Pontius et al. (2011) that simultaneously compares a reference map for time 1, a reference map for time 2 and a simulated map for time 2 at multiple resolutions. In lulccR the results of the comparison are represented by the class `ThreeMapComparison`. From objects of this class it is straightforward to extract information about different sources of agreement and disagreement, represented by the class `AgreementBudget`, which can then be plotted. The results of the comparison are conveniently summarised by the figure of merit, represented by the class`FigureOfMerit`.

In addition to the core functionality described above, lulccR inludes several utility functions to assist with the model building process. Two example datasets are also included.

Author(s)

Simon Moulds

References

Fuchs, R., Herold, M., Verburg, P.H., and Clevers, J.G.P.W. (2013). A high-resolution and harmonized model approach for reconstructing and analysing historic land changes in Europe, Biogeosciences, 10:1543-1559.

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

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

```
## Not run:

## Complete example for Plum Island Ecosystems dataset

## Load observed land use maps
obs <- ObsLulcMaps(x=pie,
                   pattern="lu",
                   categories=c(1,2,3),
                   labels=c("forest","built","other"),
                   t=c(0,6,14))

## Load explanatory variables
ef <- ExpVarMaps(x=pie, pattern="ef")

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

## convert initial land use map to RasterBrick where each layer is a boolean
## map for the respective land use
br <- raster::layerize(obs@maps[[1]])
names(br) <- obs@labels

## create data.frame to fit models
train.df <- raster::extract(x=br, y=part$train, df=TRUE)
train.df <- cbind(train.df, as.data.frame(x=ef, cells=part$train))

## model formulas
forest.formula <- formula(forest ~ 1)
built.formula <- formula(built~ef_001+ef_002+ef_003)
other.formula <- formula(built~ef_001+ef_002)
```

```
## glm models
forest.glm <- glm(formula=forest.formula, family=binomial, data=train.df)
built.glm <- glm(formula=built.formula, family=binomial, data=train.df)
other.glm <- glm(formula=other.formula, family=binomial, data=train.df)

## create PredModels objects
glm.models <- PredModels(models=list(forest.glm, built.glm, other.glm),
                         obs=obs)

## obtain demand scenario
dmd <- approxExtrapDemand(obs=obs, tout=c(0:14))

## create model input object
pie.model.input <- ModelInput(obs=obs,
                              ef=ef,
                              models=glm.models,
                              time=0:14,
                              demand=dmd)

## create ClueModel object
clues.rules <- matrix(data=c(1,1,1,
                             1,1,1,
                             1,1,1), nrow=3, ncol=3, byrow=TRUE)

clues.parms <- list(jitter.f=0.0002,
                    scale.f=0.000001,
                    max.iter=1000,
                    max.diff=50,
                    ave.diff=50)

pie.clues <- CluesModel(x=pie.model.input,
                        elas=c(0.2,0.2,0.2),
                        rules=clues.rules,
                        params=clues.parms)

## create OrderedModel object
## allocate built land first, then forest, then other
pie.ordered <- OrderedModel(x=pie.model.input,
                            order=c(2,1,3))

## perform allocation
pie.clues <- allocate(pie.clues)
pie.ordered <- allocate(pie.ordered)

## validate ordered model input
pie.ordered.tabs <- ThreeMapComparison(x=pie.ordered,
                                       factors=2^(1:9),
                                       timestep=14)

pie.ordered.agr <- AgreementBudget(x=pie.ordered.tabs)
p <- AgreementBudget.plot(x=pie.ordered.agr, from=1, to=2)
print(p)

pie.ordered.fom <- FigureOfMerit(x=pie.ordered.tabs)
p <- FigureOfMerit.plot(x=pie.ordered.fom, from=1, to=2)
print(p)
```

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

`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, ...)

## S4 method for signature ThreeMapComparison
AgreementBudget(x, ...)

## S4 method for signature RasterLayer
AgreementBudget(x, ...)
```

Arguments

`x`	a ThreeMapComparison object or RasterLayer
`...`	additional arguments to ThreeMapComparison. Only required if x is not a ThreeMapComparison object

Details

The types of agreement and disagreement considered are those descibed in Pontius et al. (2011):

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.

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-class`, `AgreementBudget.plot`, `ThreeMapComparison`, `FigureOfMerit`

Examples

```
## see lulccR-package examples
```

`AgreementBudget-class` *Class AgreementBudget*

Description

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

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`

`overall` data.frame containing the overall agreement budget

`category` list of data.frames showing the agreement budget for each category

`transition` list of data.frames showing the agreement budget for all possible transitions

---

`AgreementBudget.plot` *Plot method for AgreementBudget objects*

Description

Plot an `AgreementBudget` object.

Usage

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

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

Arguments

`x`	an AgreementBudget object
`from`	optional numeric value representing a land use category. If provided without `to` the figure of merit for all transitions from this category will be plotted
`to`	similar to `from`. If provided with a valid `from` argument the transition defined by these two arguments (i.e. `from` -> `to`) will be plotted
`col`	character specifying the plotting colour. Default is to use the 'Set2' palette 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
...	additional arguments to lattice::xyplot

Details

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

Value

A trellis object.

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

Examples

```
## see lulccR-package examples
```

---

allocate *Allocate land use change spatially*

Description

Perform spatially explicit allocation of land use change using different models. Currently the function provides an implementation of the Change in Land Use and its Effects at Small regional extent (CLUE-S) model (Verburg et al., 2002) and an ordered procedure based on the algorithm described by Fuchs et al., (2013), modified to allow stochastic transitions.

Usage

```
allocate(model, ...)

## S4 method for signature CluesModel
allocate(model, ...)

## S4 method for signature OrderedModel
allocate(model, stochastic = TRUE, ...)
```

Arguments

`model`	an object inheriting from class `Model`
`stochastic`	logical indicating whether the model should be run stochastically. Only used if `model` is an `OrderedModel` object
`...`	additional arguments for specific methods

Value

An update Model object.

References

Fuchs, R., Herold, M., Verburg, P.H., and Clevers, J.G.P.W. (2013). A high-resolution and harmonized model approach for reconstructing and analysing historic land changes in Europe, Biogeosciences, 10:1543-1559.

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

`ModelInput`, `CluesModel`, `OrderedModel`

Examples

```
## see lulccR-package examples
```

`allow` *Implement decision rules for land use change*

Description

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

Usage

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

Arguments

`x`	numeric vector containing the land use pattern for the current timestep
`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
`hist`	numeric vector containing land use history (values represent the number of timesteps the cell has contained the current land use category). Only required for rules 2 and 3
`...`	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 > 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`
3. rule > 1000: this rule imposes a minimum period of time (rule-1000) before land use is allowed to change

`allow` should be called from `allocate` methods. 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 are removed from the allocation procedure.

Value

A matrix.

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

`allowNeighb`

Examples

```
## Plum Island Ecosystems

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

## get land use values
x <- getValues(obs@maps[[1]])
x <- x[!is.na(x)]

## create vector of arbitrary land use history values
hist <- sample(1:10, length(x), replace=TRUE)

## 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=x,
               hist=hist,
               categories=obs@categories,
               cd=cd,
               rules=rules)

## create raster showing cells allowed to change from forest to built
r <- obs@maps[[1]]
r[!is.na(r)] <- allow[,2]
r[obs@maps[[1]] != 1] <- NA  ## set all cells not belonging to forest to NA
plot(r)

## 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(neighb, x, categories, rules, ...)
```

Arguments

`neighb`	a NeighbMaps object
`x`	a categorical RasterLayer to which neighbourhood rules should be applied. If `neighb` is supplied it is updated with this map
`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 `x@categories`
`...`	additional arguments (none)

Value

A matrix.

See Also

`allow`, `NeighbMaps`

Examples

```
## Plum Island Ecosystems

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

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

## 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(neighb=nb,
                        x=obs@maps[[1]],
                        categories=obs@categories,
                        rules=0.5)

## create raster showing cells allowed to change to forest
r <- obs@maps[[1]]
r[!is.na(r)] <- nb.allow[,1]
plot(r)

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

---

approxExtrapDemand *Extrapolate land use area in time*

Description

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

Usage

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

Arguments

obs	an ObsLulcMaps 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`

Details

Many allocation routines, including the two included with `lulccR`, require non-spatial estimates of land use demand for every timestep in the study period. Some routines are coupled to complex economic models that predict future or past land use demand based on economic considerations; however, linear extrapolation of trends remains a useful technique.

Value

A matrix.

See Also

`Hmisc::approxExtrap`

Examples

```
## Plum Island Ecosystems

## load observed land use maps
obs <- ObsLulcMaps(x=pie,
                    pattern="lu",
                    categories=c(1,2,3),
                    labels=c("forest","built","other"),
                    t=c(0,6,14))

## obtain demand scenario
dmd <- approxExtrapDemand(obs=obs, tout=c(0:14))
```

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

Description

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

Usage

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

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

Arguments

`x`	an ExpVarMaps object
`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

`cells`	index of cells to be extracted
`timestep`	numeric indicating the timestep under consideration. Only relevant if `x@maps` contains dynamic predictor variables
`...`	additional arguments (none)

Value

A data.frame.

See Also

`as.data.frame`, `ExpVarMaps`, `partition`

Examples

```
## Not run:

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

ef <- ExpVarMaps(x=pie, pattern="ef")
part <- partition(x=obs@maps[[1]], size=0.5, spatial=FALSE)
efdf <- as.data.frame(x=ef, cells=part$train)

## End(Not run)
```

`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 PredModels
calcProb(object, newdata, df = FALSE, ...)
```

Arguments

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

Details

This function is usually called from `allocate` to calculate land use suitability at each timestep. However, it may also be used to produce suitability maps (see examples).

Value

A matrix or data.frame.

See Also

`PredModels`, `allocate`, `glm`, `rpart::rpart`, `randomForest::randomForest`

Examples

```
## Not run:

## Sibuyan Island

## load observed land use data
obs <- ObsLulcMaps(x=sibuyan$maps,
                    pattern="lu",
                    categories=c(1,2,3,4,5),
                    labels=c("Forest","Coconut","Grass","Rice","Other"),
                    t=c(0,14))

## load explanatory variables
ef <- ExpVarMaps(x=sibuyan$maps, pattern="ef")
part <- partition(x=obs@maps[[1]], size=0.5, spatial=FALSE)
efdf <- as.data.frame(x=ef, cells=part$all)

## get training data
br <- raster::layerize(obs@maps[[1]])
names(br) <- obs@labels
train.df <- raster::extract(x=br, y=part$train, df=TRUE)
train.df <- cbind(train.df, as.data.frame(x=ef, cells=part$train))

## fit glm models
forest.glm <- glm(Forest ~ ef_001+ef_002+ef_003+ef_004+ef_005
                           +ef_006+ef_007+ef_008+ef_010+ef_012,
                  family=binomial, data=train.df)

coconut.glm <- glm(Coconut ~ ef_001+ef_002+ef_005+ef_007+ef_008
                             +ef_009+ef_010+ef_011+ef_012,
                   family=binomial, data=train.df)

grass.glm <- glm(Grass ~ ef_001+ef_002+ef_004+ef_005+ef_007
                             +ef_008+ef_009+ef_010+ef_011+ef_012+ef_013,
```

```
                   family=binomial, data=train.df)

rice.glm <- glm(Rice~ef_009+ef_010+ef_011,
                family=binomial, data=train.df)

other.glm <- glm(Other~1, family=binomial, data=train.df)

## create PredModels object
glm.models <- PredModels(models=list(forest.glm, coconut.glm, grass.glm,
                                     rice.glm, other.glm),
                         obs=obs)

probmaps <- calcProb(object=glm.models, newdata=efdf, df=TRUE)
points <- rasterToPoints(obs@maps[[1]], spatial=TRUE)
probmaps <- SpatialPointsDataFrame(coords=points, data=probmaps)
r <- rasterize(x=probmaps, y=obs@maps[[1]], field=names(probmaps))
plot(stack(r))

## End(Not run)
```

`CluesModel` *Create a CluesModel object*

Description

Methods to create a `CluesModel` object to supply to `allocate`.

Usage

```
CluesModel(x, ...)

## S4 method for signature ModelInput
CluesModel(x, elas, rules = NULL, nb.rules = NULL,
  params, output = NULL, ...)
```

Arguments

`x`	a ModelInput object
`elas`	numeric indicating the elasticity of each land use category to change. Elasticity varies between 0 and 1, with 0 indicating a low resistance to change and 1 indicating a high resistance to change
`rules`	matrix with land use change decision rules
`nb.rules`	numeric with neighbourhood decision rules
`params`	list with model parameters
`output`	either a RasterStack containing output maps or NULL
`...`	additional arguments (none)

Details

The `params` argument is a list of parameter values which 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` Scale factor which controls the amount by which suitability is increased if demand is not met. 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

Note that, in order to achieve convergence, it is likely that some adjustment of these parameters will be required.

Value

A CluesModel object.

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

`CluesModel-class`, `allocate`

Examples

```
## see lulccR-package examples
```

`CluesModel-class` *Class CluesModel*

Description

An S4 class to represent inputs to the CLUE-S land use change model.

Slots

`obs` an ObsLulcMaps object

`ef` an ExpVarMaps object

`models` a PredModels object

`time` numeric vector of 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

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

`output` RasterStack containing simulated land use maps 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

`params` list with model parameters

`compareAUC` *Calculate the area under the ROC curve (AUC)*

Description

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

Usage

```
compareAUC(pred, ...)

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

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

Arguments

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

Details

The user can compare the performance of different statistical models by providing a list of `Prediction` 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 (Pontius and Parmentier, 2014).

Value

A data.frame.

References

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

Pontius Jr, R. G., & Parmentier, B. (2014). Recommendations for using the relative operating characteristic (ROC). Landscape ecology, 29(3), 367-382.

See Also

Prediction, ROCR::performance

Examples

```
## see PredModels examples
```

---

crossTabulate *Cross tabulate land use transitions*

Description

Cross tabulate land use transitions using `raster::crosstab`. This step should form the basis of 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 ObsLulcMaps,ANY
crossTabulate(x, y, index, ...)
```

Arguments

x	RasterLayer representing land use map from an earlier timestep or an ObsLulcMaps 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 ObsLulcMaps object
categories	numeric vector containing land use categories to consider. Not used if x is an ObsLulcMaps object
labels	character vector (optional) with labels corresponding to `categories`. Not used if x is an ObsLulcMaps object
index	numeric vector with index of land use maps from ObsLulcMaps to use in analysis
...	additional arguments to `raster::crosstab`

Value

A data.frame.

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

ObsLulcMaps, raster::crosstab

Examples

```
## Plum Island Ecosystems

## Load observed land use maps
obs <- ObsLulcMaps(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))

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

ExpVarMaps *Create an ExpVarMaps object*

Description

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

Usage

```
ExpVarMaps(x, pattern, ...)

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

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

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

## S4 method for signature list,character
ExpVarMaps(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

Explanatory variables 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 explanatory variables from other maps in the directory, list or RasterStack. This should be followed by a unique number to differentiate the explanatory variables (note that the order of variables in the ExpVarMaps object is determined by this value) If the variable 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 in the simulation. The different parts should be separated by a period or underscore.

Maps of different explanatory 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 `ObsLulcMaps` object. However, maps for different timesteps of the same dynamic variable should have the same extent and resolution because these are stored as RasterStack objects.

Value

An ExpVarMaps object.

See Also

`raster::stack`

Examples

```
## Plum Island Ecosystems
ef <- ExpVarMaps(x=pie, pattern="ef")

## Sibuyan
ef <- ExpVarMaps(x=sibuyan$maps, pattern="ef")
```

`ExpVarMaps-class` *Class ExpVarMaps*

Description

An S4 class for explanatory variables.

Slots

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

`varnames` character vector with the name of each variable in `maps`

`dynamic` logical indicating whether dynamic variables are present

`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, ...)

## S4 method for signature RasterLayer
FigureOfMerit(x, ...)

## S4 method for signature ThreeMapComparison
FigureOfMerit(x, ...)
```

Arguments

`x`	a ThreeMapComparison object or RasterLayer
`...`	additional arguments to ThreeMapComparison. Only required if `x` is not a ThreeMapComparison object

Details

In land use change modelling 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 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.

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

`FigureOfMerit.plot`, `ThreeMapComparison`

Examples

```
## see lulccR-package examples
```

`FigureOfMerit-class` *Class FigureOfMerit*

Description

An S4 class for different figure of merit scores.

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`
- `overall` list containing the overall figure of merit score for each aggregation factor
- `category` list of numeric vectors containing category specific scores
- `transition` list of matrices containing transition specific scores

`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,
  col = RColorBrewer::brewer.pal(8, "Set2"), type = "b", key, scales, xlab,
  ylab, ...)
```

Arguments

`x`	a FigureOfMerit object
`from`	optional numeric value representing a land use category. If provided without `to` the figure of merit for all transitions from this category will be plotted
`to`	similar to `from`. If provided with a valid `from` argument 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

`col`	character specifying the plotting colour. Default is to use the 'Set2' palette from `RColorBrewer`
`type`	character. See `lattice::panel.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`
`...`	additional arguments to `lattice::xyplot`

Value

A trellis object.

See Also

`FigureOfMerit`, `lattice::xyplot`, `lattice::panel.xyplot`

Examples

```
## see lulccR-package examples
```

`Model-class` *Virtual class Model*

Description

A virtual S4 class to represent land use change models.

Slots

`obs` an ObsLulcMaps object

`ef` an ExpVarMaps object

`models` a PredModels object

`time` numeric vector of 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

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

`output` RasterStack containing simulated land use maps or NULL

ModelInput *Create a ModelInput object*

Description

Methods to combine several object classes that are useful for land use change modelling and perform checks to ensure the objects are compatible in time and space for a model simulation.

Usage

```
ModelInput(obs, ef, models, time, demand, ...)

## S4 method for signature ObsLulcMaps,ExpVarMaps,PredModels,numeric,matrix
ModelInput(obs, ef,
  models, time, demand, hist, mask, neighb = NULL, ...)
```

Arguments

`obs`	an ObsLulcMaps or ModelInput object
`ef`	an ExpVarMaps object
`models`	a PredModels 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)
`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
`...`	additional arguments (none)

Value

A ModelInput object.

See Also

`ModelInput-class`

Examples

```
## see lulccR-package examples
```

`ModelInput-class` *Class ModelInput*

Description

An S4 class to represent land use change model inputs.

Slots

- `obs` an ObsLulcMaps object
- `ef` an ExpVarMaps object
- `models` a PredModels 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
- `categories` numeric vector of land use categories
- `labels` character vector corresponding to `categories`

`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.

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 containing 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.
`fun`	function. Input argument to `focal`. Default is `mean`
`...`	additional arguments to `raster::focal`

Value

A NeighbMaps object.

See Also

`NeighbMaps-class`, `allowNeighb`, `raster::focal`

Examples

```
## observed data
obs <- ObsLulcMaps(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=c(3,3,3))          # 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)

## plot neighbourhood map for forest
plot(nb2@maps[[1]])
```

NeighbMaps-class *Class NeighbMaps*

Description

An S4 class for neighbourhood maps.

Slots

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

`categories` numeric vector of land use categories for which neighbourhood maps exist

`weights` list of weights matrices

`fun` function used to calculate neighbourhood values

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

ObsLulcMaps *Create an ObsLulcMaps object*

Description

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

Usage

```
ObsLulcMaps(x, pattern, ...)

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

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

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

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

## S4 method for signature RasterStack,ANY
ObsLulcMaps(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
`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
`...`	additional arguments to `raster::stack`

Details

Observed land use maps should have the same extent and resolution. The location of non-NA cells in `ObsLulcMaps` objects defines the region for subsequent analysis.

Value

An ObsLulcMaps object.

See Also

`ObsLulcMaps-class`, `raster::stack`

Examples

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

## Sibuyan Island
obs <- ObsLulcMaps(x=sibuyan$maps,
                   pattern="lu",
                   categories=c(1,2,3,4,5),
                   labels=c("forest","coconut","grass","rice","other"),
                   t=c(0,14))
```

`ObsLulcMaps-class` *Class ObsLulcMaps*

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

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

`OrderedModel` *Create an OrderedModel object*

Description

Methods to create a `OrderedModel` object to supply to `allocate`.

Usage

```
OrderedModel(x, ...)

## S4 method for signature ModelInput
OrderedModel(x, rules = NULL, nb.rules = NULL, order,
  params, output = NULL, ...)
```

Arguments

`x`	a ModelInput object
`rules`	matrix with land use change decision rules
`nb.rules`	numeric with neighbourhood decision rules
`order`	numeric vector of land use categories in the order that change should be allocated. See Details
`params`	list with model parameters
`output`	either a RasterStack containing output maps or NULL
`...`	additional arguments (none)

Details

The `params` argument is a list of parameter values which should contain the following components:

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

Value

An OrderedModel object.

References

Fuchs, R., Herold, M., Verburg, P.H., and Clevers, J.G.P.W. (2013). A high-resolution and harmonized model approach for reconstructing and analysing historic land changes in Europe, Biogeosciences, 10:1543-1559.

See Also

OrderedModel-class, allocate

Examples

```
## see luloccR-package examples
```

`OrderedModel-class` *Class OrderedModel*

Description

An S4 class to represent inputs to the Ordered allocation procedure

Slots

- `obs` an ObsLulcMaps object
- `ef` an ExpVarMaps object
- `models` a PredModels object
- `time` numeric vector of 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
- `categories` numeric vector of land use categories
- `labels` character vector corresponding to `categories`
- `output` RasterStack containing simulated land use maps or NULL
- `rules` matrix with land use change decision rules
- `nb.rules` numeric with neighbourhood decision rules
- `order` numeric vector of land use categories in the order that change should be allocated
- `params` list with model parameters

`partition` *Partition raster data*

Description

Divide a categorical raster map into training and testing partitions. A wrapper function for `caret::createDataPartition` (Kuhn, 2008) to 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 value 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

`all` a SpatialPoints object or numeric vector indicating all non-NA cells in the study region

References

Kuhn, M. (2008). Building predictive models in R using the caret package. Journal of Statistical Software, 28(5), 1-26.

See Also

`caret::createDataPartition`

Examples

```
## Plum Island Ecosystems

## Load observed land use maps
obs <- ObsLulcMaps(x=pie,
                   pattern="lu",
                   categories=c(1,2,3),
                   labels=c("forest","built","other"),
                   t=c(0,6,14))

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

`Performance` *Create a Performance object*

Description

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

Usage

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

Arguments

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

Value

A `Performance` object.

References

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

See Also

`performance`, `Prediction`

Examples

```
## see PredModels examples
```

`performance` *Create ROCR performance objects*

Description

A wrapper function for `ROCR::performance` (Sing et al, 2005) 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 `ROCR::prediction` objects
`measure`	performance measure to use for the evaluation. See `ROCR::performance`
`x.measure`	a second performance measure. See `ROCR::performance`
`...`	additional arguments to `ROCR::performance`

Value

A list of `performance` objects.

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

Performance-class *Class Performance*

Description

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

Slots

performance list of ROCR performance objects. Each object is calculated for the corresponding ROCR prediction object held in the Prediction 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 for which performance objects were created

labels character vector with labels corresponding to categories

---

Performance.plot *Plot method for Performance objects*

Description

Plot the the ROC curve for each performance object in a Performance object. If more than one Performance objects are provided ROC curves for the same land use category from different objects are included on the same plot for model comparison.

Usage

```
Performance.plot(x, ...)

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

Arguments

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

Value

A trellis object.

See Also

Performance, lattice::xyplot

Examples

```
## see PredModels examples
```

---

pie *Land use change dataset for Plum Island Ecosystem*

Description

Dataset containing land use maps for 1985, 1991 and 1999 and several explanatory 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

ef_001 RasterLayer showing elevation

ef_002 RasterLayer showing slope

ef_003 RasterLayer showing distance to built land in 1985

References

Pontius Jr, R. G., & Parmentier, B. (2014). Recommendations for using the relative operating characteristic (ROC). Landscape ecology, 29(3), 367-382.

`Prediction` *Create a Prediction object*

Description

This function creates a `ROCR::prediction` object for each predictive model in a `PredModels` object. It should be used with `Performance` to evaluate multiple models with exactly the same criteria while keeping track of which model corresponds to which land use category.

Usage

```
Prediction(models, obs, ef, timestep = 0, partition, ...)
```

Arguments

`models`	a PredModels object
`obs`	an ObsLulcMaps object
`ef`	an ExpVarMaps object
`timestep`	numeric indicating the timestep of the observed map in obs against which the observed map should be tested
`partition`	index of cells for which occurrence should be predicted
`...`	additional arguments to `ROCR::prediction`

Value

A `Prediction` object.

References

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

See Also

`link{Performance}`, `ROCR::prediction`

Examples

```
## see PredModels examples
```

Prediction-class *Class Prediction*

Description

An S4 class that extends ROCR::prediction-class 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 PredModels 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

PredModels *Create a PredModels object*

Description

Methods to create a PredModels-class object.

Usage

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

## S4 method for signature list,ObsLulcMaps,ANY,ANY
PredModels(models, obs, categories, labels,
  ...)

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

Arguments

models	a list of predictive models corresponding to obs@categories or categories
obs	an ObsLulcMaps 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

A PredModels object.

See Also

glm, rpart::rpart, randomForest::randomForest

Examples

```
## Not run:

## Plum Island Ecosystems

## Load observed land use maps
obs <- ObsLulcMaps(x=pie,
                   pattern="lu",
                   categories=c(1,2,3),
                   labels=c("forest","built","other"),
                   t=c(0,6,14))

## Load explanatory variables
ef <- ExpVarMaps(x=pie, pattern="ef")

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

## convert initial land use map to RasterBrick where each layer is a boolean
## map for the respective land use
br <- raster::layerize(obs@maps[[1]])
names(br) <- obs@labels

## create data.frame to fit models
train.df <- raster::extract(x=br, y=part$train, df=TRUE)
train.df <- cbind(train.df, as.data.frame(x=ef, cells=part$train))

## model formulas
forest.formula <- formula(forest ~ 1)
built.formula <- formula(built~ef_001+ef_002+ef_003)
other.formula <- formula(built~ef_001+ef_002)

## glm models
forest.glm <- glm(formula=forest.formula, family=binomial, data=train.df)
built.glm <- glm(formula=built.formula, family=binomial, data=train.df)
other.glm <- glm(formula=other.formula, family=binomial, data=train.df)

## NB rpart and randomForest do not accept null model formula, so only fit
## models for built and other classes

## rpart models
built.rp <- rpart::rpart(formula=built.formula, data=train.df, method="class")
other.rp <- rpart::rpart(formula=other.formula, data=train.df, method="class")

## random forest models (warning: takes a long time!)
built.rf <- randomForest::randomForest(formula=built.formula, data=train.df)
other.rf <- randomForest::randomForest(formula=other.formula, data=train.df)

## create PredModels object
glm.models <- PredModels(models=list(forest.glm, built.glm, other.glm),
                         obs=obs)
```

```
rp.models <- PredModels(models=list(built.rp, other.rp),
                        categories=obs@categories[2:3],
                        labels=obs@labels[2:3])

rf.models <- PredModels(model=list(built.rf, other.rf),
                        categories=obs@categories[2:3],
                        labels=obs@labels[2:3])

## obtain Prediction objects
glm.pred <- Prediction(models=glm.models, obs=obs, ef=ef, partition=part$test)
rp.pred <- Prediction(models=rp.models, obs=obs, ef=ef, partition=part$test)
rf.pred <- Prediction(models=rf.models, obs=obs, ef=ef, partition=part$test)

## quickly compare area under the curve
compareAUC(pred=list(glm=glm.pred, rpart=rp.pred, randomForest=rf.pred))

## obtain Performance objects
glm.perf <- Performance(pred=glm.pred, measure="rch")
rp.perf <- Performance(pred=rp.pred, measure="rch")
rf.perf <- Performance(pred=rf.pred, measure="rch")

## plot ROC curve
p <- Performance.plot(list(glm=glm.perf, rpart=rp.perf, rf=rf.perf),
                      layout=c(3,1),
                      aspect="iso",
                      xlab=list(label=""),
                      ylab=list(label=""),
                      scales=list(cex=0.6),
                      key.args=list(cex=0.4, size=2.5),
                      par.strip.text=list(cex=0.6),
                      par.settings=list(strip.background=
                                          list(col="lightgrey")))

## view plot
print(p)

## End(Not run)
```

PredModels-class *Class PredModels*

Description

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

Slots

`models` list of predictive models

`categories` numeric vector of land use categories

`labels` character vector with labels corresponding to `categories` resample *Resample maps in ExpVarMaps object or list*

Description

A wrapper function for raster::resample to resample raster objects in an ExpVarMaps object or list.

Usage

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

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

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

Arguments

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

Value

An ExpVarMaps object or list, depending on x.

See Also

ExpVarMaps, raster::resample

Examples

```
## Not run:

## create ExpVarMaps object
ef <- ExpVarMaps(x=pie, pattern="ef")

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

## End(Not run)
```

---

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
...	additional arguments (none)

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 the units of demand to number of cells, as required by ModelInput.

Value

A matrix.

Examples

```
## Sibuyan Island

## load demand scenario from data
dmd <- sibuyan$demand$demand1 * runif(1)
ncell <- length(which(!is.na(getValues(sibuyan$maps$lu_sib_1997))))

## recover demand
dmd <- roundSum(dmd, ncell=ncell)
```

---

sibuyan *Land use change dataset for Sibuyan Island*

Description

Dataset containing land use map for 1997 and several explanatory 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:

maps list containing the following RasterLayers:

- **lu_sib_1997** RasterLayer with land use in 1997 (forest, coconut, grassland, rice, other)
- **ef_001** RasterLayer showing distance to sea
- **ef_002** RasterLayer showing mean population density
- **ef_003** RasterLayer showing occurrence of diorite rock
- **ef_004** RasterLayer showing occurrence of ultramafic rock
- **ef_005** RasterLayer showing occurrence of sediments
- **ef_006** RasterLayer showing areas with no erosion
- **ef_007** RasterLayer showing areas with moderate erosion
- **ef_008** RasterLayer showing elevation
- **ef_009** RasterLayer showing slope
- **ef_010** RasterLayer showing aspect
- **ef_011** RasterLayer showing distance to roads in 1997
- **ef_012** RasterLayer showing distance to urban areas in 1997
- **ef_013** RasterLayer showing distance to streams
- **restr1** RasterLayer showing location of current national park
- **restr2** RasterLayer showing location of proposed national park

demand list of matrices with different demand scenarios:

- **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.

`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 (minor allocation disagreement), disagreement between coarse squares (major allocation disagreement), disagreement about the quantity of land use change and agreement.

Usage

```
ThreeMapComparison(x, x1, y1, ...)

## S4 method for signature Model,ANY,ANY
ThreeMapComparison(x, x1, y1, factors, timestep, ...)

## S4 method for signature RasterLayer,RasterLayer,RasterLayer
ThreeMapComparison(x, x1, y1,
  factors, categories, labels, ...)
```

Arguments

`x`	either a RasterLayer of observed land use at time 0 or an object inheriting from class `Model`
`x1`	a RasterLayer of observed land use at a subsequent time. Only required if `x` is also a RasterLayer
`y1`	a RasterLayer of simulated land use corresponding to `x1`. Only required if `x` is also a RasterLayer
`factors`	numeric vector of aggregation factors (equivalent to the 'fact' argument to `raster::aggregate` representing the resolutions at which model performance should be tested
`timestep`	numeric value indicating the timestep of the simulated land use map. Only required if `x` is a `Model` object
`categories`	numeric vector of land use categories in observed maps. Only required if `x` is a RasterLayer
`labels`	character vector (optional) with labels corresponding to `categories`. Only required if `x` is a RasterLayer
`...`	additional arguments to `raster::aggregate`

Value

A `ThreeMapComparison` object.

References

Pontius Jr, R.G., Peethambaram, S., Castella, J.C. (2011). Comparison of three maps at multiple resol utions: 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`, `raster::aggregate`

Examples

```
## see lulccR-package examples
```

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`

References

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

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

Examples

```
## RasterLayer
total(x=sibuyan$maps$lu_sib_1997)

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

Index