Package ‘lulcc’

March 1, 2019

Title Land Use Change Modelling in R
Version 1.0.4
Author Simon Moulds <sim.moulds@gmail.com>
Maintainer Simon Moulds <sim.moulds@gmail.com>
Description Classes and methods for spatially explicit land use change modelling in R.
Depends methods, raster, R (>= 3.1.0)
License GPL (>= 2)
LazyData true
Imports ROCR, lattice, rasterVis
Suggests caret, rpart, randomForest, gsubfn, Hmisc, plyr, RColorBrewer,
Collate 'class-CategoryLabel.R' 'class-ObsLulcRasterStack.R'
 'class-PredictiveModelList.R' 'class-ExpVarRasterList.R'
 'class-NeighbRasterStack.R' 'class-Model.R'
 'class-ThreeMapComparison.R' 'AgreementBudget.R'
 'ExpVarRasterList.R' 'FigureOfMerit.R' 'Model.R'
 'NeighbRasterStack.R' 'ObsLulcRasterStack.R'
 'class-PerformanceList.R' 'PerformanceList.R'
 'class-PredictionList.R' 'PredictionList.R'
 'class-PredictiveModelList.R' 'Interest.R'
 'ThreeMapComparison.R' 'allocate.R' 'allow.R' 'allowNeighb.R'
 'approxExtrapDemand.R' 'as.data.frame.R'
 'class-PredictiveModelList.R' 'index.R' 'coerce.R' 'compareAUC.R'
 'crossTabulate.R' 'data.R' 'getPredictiveModelInputData.R'
 'length.R' 'lulcc-package.R' 'models.R' 'names.R' 'partition.R'
 'performance.rocr.R' 'plot.AgreementBudget.R'
 'plot.FigureOfMerit.R' 'plot.Performance.R' 'plot.R'
 'predict.R' 'resample.R' 'show.R' 'subset.R' 'summary.R'
 'total.R'
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-03-01 10:50:04 UTC
RoxygenNote 6.0.1
R topics documented:

lulcc-package ........................................... 3
AgreementBudget ......................................... 7
AgreementBudget-class .................................... 8
allocate ..................................................... 9
allow ....................................................... 10
allowNeigh ............................................... 12
approxExtrapDemand ..................................... 13
as.data.frame.ExpVarRasterList ......................... 14
c.PredictiveModelList .................................. 16
CategoryLabel-class .................................... 17
CluesModel ............................................... 17
CluesModel-class ....................................... 19
compareAUC .............................................. 20
crossTabulate ............................................ 21
ExpVarRasterList ........................................ 22
ExpVarRasterList-class .................................. 24
Extract by index ........................................ 24
FigureOfMerit ........................................... 25
FigureOfMerit-class .................................... 26
getPredictiveModelInputData ............................. 27
Model fitting ............................................. 27
Model-class ............................................... 28
NeighbRasterStack ....................................... 29
NeighbRasterStack-class ................................ 30
ObsLulcRasterStack ...................................... 31
ObsLulcRasterStack-class ................................. 32
OrderedModel ............................................ 33
OrderedModel-class ..................................... 35
partition .................................................. 35
performance .............................................. 37
PerformanceList ......................................... 38
PerformanceList-class ................................... 39
pie ......................................................... 39
plot ......................................................... 40
plot.AgreementBudget .................................. 41
plot.FigureOfMerit ...................................... 42
plot.PerformanceList .................................... 43
predict.PredictiveModelList ............................. 44
PredictionList ........................................... 46
PredictionList-class ..................................... 47
PredictiveModelList-class ............................... 47
resample.ExpVarRasterList,Raster-method ............... 47
roundSum ............................................... 49
show.ExpVarRasterList-method .......................... 50
sibuyan .................................................. 50
subset.ExpVarRasterList-method ........................ 52
**Description**

The `lulcc` 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 several aspects 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. Generic methods implemented for the `lulcc` classes include `summary`, `show`, and `plot`.

Land use change 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 inductive 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 `lulcc` these models are represented by the class `PredictiveModelList`. Currently `lulcc` 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 `lulcc` 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 `lulcc` 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, `lulcc` includes several utility functions to assist with the model building process. Two example datasets are also included.

**Author(s)**

Simon Moulds

**References**


**Examples**

```r
## Not run:

## Plum Island Ecosystems

## load data
data(pie)

## observed maps
obs <- ObsLulcRasterStack(x=pie,
                           pattern="lu",
                           categories=c(1,2,3),
                           labels=c("Forest","Built","Other"),
                           t=c(0,6,14))

obs

plot(obs)

crossTabulate(obs, times=c(0,14))

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

ef

part <- partition(x=obs[[1]], size=0.1, spatial=TRUE)
train.data <- getPredictiveModellInputData(obs=obs, ef=ef, cells=part[["train"]])
```
forms <- list(Built = ef_001+ef_002+ef_003,
Forest = ef_001+ef_002,
Other = ef_001+ef_002)

glm.models <- glmModels(formula=formats, family=binomial, data=train.data, obs=obs)
rpart.models <- rpartModels(formula=formats, data=train.data, obs=obs)
rf.models <- randomForestModels(formula=formats, data=train.data, obs=obs)

## test ability of models to predict allocation of forest, built and other land uses in testing partition
test.data <- getPredictiveModelInputData(obs=obs, ef=ef, cells=part[["test"]])

glm.pred <- PredictionList(models=glm.models, newdata=test.data)
glm.perf <- PerformanceList(pred=glm.pred, measure="rch")
rpart.pred <- PredictionList(models=rpart.models, newdata=test.data)
rpart.perf <- PerformanceList(pred=rpart.pred, measure="rch")
rf.pred <- PredictionList(models=rf.models, newdata=test.data)
rf.perf <- PerformanceList(pred=rf.pred, measure="rch")

plot(list(glm=glm.perf, rpart=rpart.perf, rf=rf.perf))

## test ability of models to predict location of urban gain 1985 to 1991
test.data <- getPredictiveModelInputData(obs=obs, ef=ef, cells=part, t=6)

glm.pred <- PredictionList(models=glm.models[[2]], newdata=test.data)
glm.perf <- PerformanceList(pred=glm.pred, measure="rch")

plot(list(glm=glm.perf))

## obtain demand scenario
dmd <- approxExtrapDemand(obs=obs, tout=0.14)

matplot(dmd, type="l", ylab="Demand (no. of cells)", xlab="Time point",
1ty=1, col=col("Green", "Red", "Blue"))
legend("topleft", legend=obs@labels, col=col("Green", "Red", "Blue"), lty=1)

## get neighbourhood values
w <- matrix(data=1, nrow=3, ncol=3)

nb <- NeighRasterStack(x=obs[[1]], weights=w, categories=2)

## create CLUE-S model object
c1ues.rules <- matrix(data=1, nrow=3, ncol=3, byrow=TRUE)

c1ues.parms <- list(jitter.f=0.0002,
  scale.f=0.000001,
  max.iter=1000,
  max.diff=50,
  ave.diff=50)

c1ues.model <- CluesModel(obs=obs,
  ef=ef,
models = glm.models,
time = 0:14,
demand = dmd,
elas = c(0.2, 0.2, 0.2),
rules = clues.rules,
params = clues.parms)

## Create Ordered model
ordered.model <- OrderedModel(obs = obs,
                              ef = ef,
                              models = glm.models,
                              time = 0:14,
                              demand = dmd,
                              order = c(2, 1, 3))

## perform allocation
clues.model <- allocate(clues.model)
ordered.model <- allocate(ordered.model, stochastic = TRUE)

## pattern validation
## CLUE-S
clues.tabs <- ThreeMapComparison(x = clues.model,
                                  factors = 2^(1:8),
                                  timestep = 14)
plot(clues.tabs)
plot(clues.tabs, category = 1, factors = 2^(1:8)[c(1, 3, 5, 7)])

## Ordered
ordered.tabs <- ThreeMapComparison(x = ordered.model,
                                    factors = 2^(1:8),
                                    timestep = 14)
plot(ordered.tabs)
plot(ordered.tabs, category = 1, factors = 2^(1:8)[c(1, 3, 5, 7)])

## calculate agreement budget and plot
## CLUE-S
clues.agr <- AgreementBudget(x = clues.tabs)
plot(clues.agr, from = 1, to = 2)

## Ordered
ordered.agr <- AgreementBudget(x = ordered.tabs)
plot(ordered.agr, from = 1, to = 2)

## calculate Figure of Merit and plot
## CLUE-S
clues.fom <- FigureOfMerit(x = clues.tabs)
p1 <- plot(clues.fom, from = 1, to = 2)

## Ordered
ordered.fom <- FigureOfMerit(x = ordered.tabs)
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 described 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


See Also

AgreementBudget-class, plot.AgreementBudget, ThreeMapComparison, FigureOfMerit

Examples

## see lulcc-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
maps  list of RasterStack objects containing land use maps at different resolutions
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
**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**

```r
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`
- `...`:
  - additional arguments for specific methods
- `stochastic`:
  - logical indicating whether the model should be run stochastically. Only used if `model` is an `OrderedModel` object

**Value**

An updated Model object.

**References**


**See Also**

- `CluesModel`
- `OrderedModel`

**Examples**

```r
## see lulcc-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


See Also

allowNeighb

Examples

```r
## Plum Island Ecosystems

## load observed land use data
obs <- ObsLulcRasterStack(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[[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 that are allowed to change from forest to built
r <- obs[[1]]
r[!is.na(r)] <- allow[,2]
r[obs[[1]] != 1] <- NA
plot(r)

## NB output is only useful when used within 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 NeighbRasterStack 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, NeighbRasterStack

Examples

```r
## Plum Island Ecosystems

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

## create a NeighbRasterStack object for forest only
w <- matrix(data=1, nrow=3, ncol=3)
nb <- NeighbRasterStack(x=obs[,1], weights=w, categories=1)
```
## 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

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

### Arguments

- **obs**: an ObsLulcRasterStack 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 `lulcc`, 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::approxExtrp`

Examples

```r
## Plum Island Ecosystems

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

## obtain demand scenario by interpolating between observed maps
dmd <- approxExtrpDemand(obs=obs, tout=c(0:14))

## plot
demand <- approxExtrpDemand(obs=obs, tout=c(0:14))

matplot(dmd, type="l", ylab="Demand (no. of cells)", xlab="Time point",
    lty=1, col=c("Green","Red","Blue"))
legend("topleft", legend=obs$labels, col=c("Green","Red","Blue"), lty=1)

## linear extrapolation is also possible
dmd <- approxExtrpDemand(obs=obs, tout=c(0:50))

## plot
demand <- approxExtrpDemand(obs=obs, tout=c(0:50))

matplot(dmd, type="l", ylab="Demand (no. of cells)", xlab="Time point",
    lty=1, col=c("Green","Red","Blue"))
legend("topleft", legend=obs$labels, col=c("Green","Red","Blue"), lty=1)
```

as.data.frame.ExpVarRasterList

`Coerce objects to data.frame`

Description

This function extracts data from all raster objects in `ObsLulcRasterStack` or `ExpVarRasterList` objects for a specified timestep.

Usage

```r
## S3 method for class 'ExpVarRasterList'
as.data.frame(x, row.names = NULL,
    optional = FALSE, cells, t = 0, ...)

## S3 method for class 'ObsLulcRasterStack'
as.data.frame(x, row.names = NULL,
    ...)
 Arguments

x an ExpVarRasterList or ObsLulcRasterStack 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, which may be a SpatialPoints* object or a numeric vector representing cell numbers (see raster::extract)
t numeric indicating the time under consideration
... additional arguments (none)

 Details

If x is an ObsLulcRasterStack object the raster corresponding to t is first transformed to a RasterBrick with a boolean layer for each class with raster::layerize.

 Value

A data.frame.

 See Also

as.data.frame, ObsLulcRasterStack, ExpVarRasterList, partition

 Examples

## Not run:

## Plum Island Ecosystems

## observed maps
obs <- ObsLulcRasterStack(x=pie,
   pattern="lu",
categories=c(1,2,3),
labels=c("Forest","Built","Other"),
t=c(0,6,14))
## c.PredictiveModelList

### Merge PredictiveModelList objects

**Description**

Combine different PredictiveModelList objects into one

**Usage**

```r
## S3 method for class 'PredictiveModelList'
c(..., recursive = FALSE)
```

**Arguments**

- `...` two or more PredictiveModelList objects
- `recursive` for consistency with generic method (ignored)

**Value**

a PredictiveModelList object

**Examples**

```r
## Not run:

## Plum Island Ecosystems

## load data
data(pie)

## observed maps
obs <- ObsLulcRasterStack(x=pie,
                           pattern="lu",
                           categories=c(1,2,3),
                           labels=c("Forest","Built","Other"),
                           t=c(0,6,14)))
```
## CategoryLabel-class

`CategoryLabel` is a virtual S4 class to represent information about categorical Raster objects. It contains slots for `categories` and `labels`.

### Slots
- **categories**: numeric vector of land use categories
- **labels**: character vector corresponding to categories

## CluesModel

`CluesModel` is a virtual class that represents model objects created with the `clues` function.

### Description

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

```r
## explanatory variables
ef <- ExpVarRasterList(x=pie, pattern="ef")

part <- partition(x=obs[[1]], size=0.1, spatial=TRUE)
train.data <- getPredictiveModelInputData(obs=obs, ef=ef, cells=part["train"], t=0)

forms <- list(Built ~ ef_001+ef_002+ef_003,
              Forest ~ 1,
              Other ~ ef_001+ef_002)

glm.models <- glmModels(formula=forms, family=binomial, data=train.data, obs=obs)

glm.models

## separate glm.models into two PredictiveModelList objects
mod1 <- glm.models[[1]]
mod2 <- glm.models[[2:3]]

## put them back together again
glm.models <- c(mod1, mod2)

## End(Not run)
```
Usage

CluesModel(obs, ef, models, ...)

## S4 method for signature
## 'ObsLulcRasterStack,ExpVarRasterList,PredictiveModelList'
CluesModel(obs,
    ef, models, time, demand, hist, mask, neighb = NULL, elas, rules = NULL,
    nb.rules = NULL, params, output = NULL, ...)

Arguments

- **obs**: an ObsLulcRasterStack
- **ef**: an ExpVarRasterList object
- **models**: a PredictiveModelList object
- **...**: additional arguments (none)
- **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 NeighbRasterStack
- **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

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

See Also
CluesModel-class, allocate

Examples

## see lulcc-package examples

---

CluesModel-class  Class CluesModel

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

Slots
obs  an ObsLulcRasterStack object
ef  an ExpVarRasterList object
models  a PredictiveModelList object
time numeric vector of timesteps over which simulation will occur
demand matrix containing demand scenario
hist RasterLayer showing land use history or NULL
mask RasterLayer showing masked areas or NULL
neighb NeighbRasterStack object or NULL
categories numeric vector of land use categories
labels character vector corresponding to categories
compareAUC

rules matrix with land use change decision rules
nb.rules numeric with neighbourhood decision rules
elas numeric indicating elasticity to change (only required for
dparams list with model parameters
output RasterStack containing simulated land use maps or NULL

calculate the area under the ROC curve (AUC)

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

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

Arguments
- `pred` a PredictionList 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 PredictionList 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
crosTabulate

See Also

PredictionList, ROCR::performance

Examples

## see PredictiveModellist examples

crosTabulate(x, y, ...)  

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

## S4 method for signature 'ObsLulcRasterStack,ANY'
crosTabulate(x, y, times, ...)

Arguments

x RasterLayer representing land use map from an earlier timestep or an ObsLulcRasterStack 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 ObsLulcRasterStack object

... additional arguments to raster::crosstab

categories numeric vector containing land use categories to consider. Not used if x is an ObsLulcRasterStack object

labels character vector (optional) with labels corresponding to categories. Not used if x is an ObsLulcRasterStack object

times numeric vector representing the time points of two land use maps from ObsLulcRasterStack

Value

A data.frame.
References


See Also

ObsLulcRasterStack, raster::crosstab

Examples

```r
## Plum Island Ecosystems

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

## RasterLayer input
crossTabulate(x=obs[[1]],
    y=obs[[3]],
    categories=c(1,2,3),
    labels=c("forest","built","other"))
```

---

**ExpVarRasterList**

*Create an ExpVarRasterList 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

`ExpVarRasterList(x, ...)`

```r
## S4 method for signature 'missing'
ExpVarRasterList(x, pattern = NULL, ...)

## S4 method for signature 'character'
ExpVarRasterList(x, pattern = NULL, ...)

## S4 method for signature 'RasterStack'
```
ExpVarRasterList

ExpVarRasterList(x, pattern = NULL, ...)

## S4 method for signature 'list'
ExpVarRasterList(x, pattern = NULL, ...)

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

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 ExpVarRasterList 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 ObsLulcRasterStack 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 ExpVarRasterList object.

See Also

raster::stack

Examples

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

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

Class ExpVarRasterList

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

names character vector with the name of each variable in maps
dynamic logical indicating whether dynamic variables are present

Extract by index

Description

object[[i]] can be used to extract individual objects from container classes such as ExpVarRasterList,
PredictiveModellist, PredictionList and PerformanceList.

Usage

## S4 method for signature 'ExpVarRasterList,ANY,ANY'
x[[i, j, ...]]

## S4 method for signature 'CategoryLabel,ANY,ANY'
x[[i, j, ...]]

Arguments

x an object of class ExpVarRasterList or any object inheriting from the virtual
class CategoryLabel

i layer number (if 'x' inherits from a RasterStack) or list index (if 'x' stores data
as a list)

j numeric (not used)

... additional arguments (none)
### Examples

```r
## Plum Island Ecosystems

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

summary(obs[[1]])
summary(obs[[1:2]])
```

---

**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**

```r
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


See Also

plot.FigureOfMerit, ThreeMapComparison

Examples

## see lulcc-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
maps  list of RasterStack objects containing land use maps at different resolutions
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
getPredictiveModelInputData

Extract data to fit predictive models

Description

Extract a data.frame containing variables required for fitting predictive models.

Usage

generatePredictiveModelInputData(obs, ef, cells, ...)

Arguments

- obs: an ObsLulcRasterStack object
- ef: an ExpVarRasterList object
- cells: index of cells to be extracted, which may be a SpatialPoints* object or a numeric vector representing cell numbers (see raster::extract)
- ...: additional arguments to as.data.frame

Value

A data.frame.

See Also

  as.data.frame, ObsLulcRasterStack, ExpVarRasterList, partition

Examples

  # TODO

Model fitting

Fit predictive models

Description

These functions fit parametric and non-parametric models to data.
Usage

glmModels(formula, family = binomial, model = FALSE, ..., obs, categories = NA, labels = NA)

randomForestModels(formula, ..., obs, categories = NA, labels = NA)

rpartModels(formula, ..., obs, categories = NA, labels = NA)

Arguments

formula         list containing formula objects
family          see glm. Default is 'binomial'. Only used by glmModels
model           see glm. Default is FALSE. Only used by glmModels
...             additional arguments to specific functions
obs             an ObsLulcRasterStack object
categories      numeric vector of land use categories in observed maps. Only required if 'obs' is missing
labels          character vector (optional) with labels corresponding to categories. Only required if 'obs' is missing

Value

A PredictiveModelList object.

See Also

glm, rpart::rpart, randomForest::randomForest

Examples

## see lulcc-package examples

---

Model-class  Virtual class Model

Description

A virtual S4 class to represent land use change models.

Slots

output  RasterStack containing simulated land use maps or NULL
Create a NeighbRasterStack 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

```r
NeighbRasterStack(x, weights, neighb, ...)  
## S4 method for signature 'RasterLayer,list,ANY'
NeighbRasterStack(x, weights, neighb,  
  categories, fun = mean, ...)
## S4 method for signature 'RasterLayer,matrix,ANY'
NeighbRasterStack(x, weights, neighb,  
  categories, fun = mean, ...)
## S4 method for signature 'RasterLayer,ANY,NeighbRasterStack'
NeighbRasterStack(x, weights,  
  neighb)
```

Arguments

- **x**: RasterLayer containing categorical data
- **weights**: list containing a matrix of weights (the `w` argument in `raster::focal`) for each land use category. The order of list or vector elements should correspond to the order of land use categories in `categories`
- **neighb**: NeighbRasterStack object. Only used if `categories` and `weights` are not provided. This option can be useful when existing NeighbRasterStack objects need to be updated because a new land use map is available, such as during the allocation procedure.
- **categories**: numeric vector containing land use categories for which neighbourhood values should be calculated
- **fun**: function. Input argument to `focal`. Default is `mean`

Value

A NeighbRasterStack object.
See Also

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

Examples

```r
## Plum Island Ecosystems

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

## create a NeighbRasterStack object for 1985 land use map
w1 <- matrix(data=1, nrow=3, ncol=3, byrow=TRUE)
w2 <- w1
w3 <- w1

nb1 <- NeighbRasterStack(x=obs[[1]],
    categories=c(1,2,3),
    weights=list(w1,w2,w3))

## update nb2 for 1991
nb2 <- NeighbRasterStack(x=obs[[2]],
    neigh=nb1)

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

---

### NeighbRasterStack-class

**Class NeighbRasterStack**

**Description**

An S4 class for neighbourhood maps.

**Slots**

- `filename` see `raster::Raster-class`
- `layers` see `raster::Raster-class`
- `title` see `raster::Raster-class`
- `extent` see `raster::Raster-class`
- `rotated` see `raster::Raster-class`
ObsLulcRasterStack

Create an ObsLulcRasterStack object

Description

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

Usage

ObsLulcRasterStack(x, pattern, ...)

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

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

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

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

## S4 method for signature 'RasterStack,ANY'
ObsLulcRasterStack(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
ObsLulcRasterStack-class

Description

An S4 class for observed land use maps.

details

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

Value

An ObsLulcRasterStack object.

See Also

ObsLulcRasterStack-class, raster::stack

Examples

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

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

Slots

filename see raster::Raster-class
layers see raster::Raster-class
title see raster::Raster-class
extent see raster::Raster-class
rotated see raster::Raster-class
rotation see raster::Raster-class
ncols see raster::Raster-class
nrows see raster::Raster-class
crs see raster::Raster-class
history see raster::Raster-class
z see raster::Raster-class

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(obs, ef, models, ...)

## S4 method for signature
## 'ObsLulcRasterStack,ExpVarRasterList,PredictiveModelList'
OrderedModel(obs,
  ef, models, time, demand, hist, mask, neighb = NULL, rules = NULL,
  nb.rules = NULL, order, params, output = NULL, ...)

Arguments

obs an ObsLulcRasterStack object
ef an ExpVarRasterList object
models a PredictiveModelList object
... additional arguments (none)
time numeric vector containing timesteps over which simulation will occur
**OrderedModel**

```r
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)
```

```r
hist RasterLayer containing land use history (values represent the number of years the cell has contained the current land use category)
```

```r
mask RasterLayer containing binary values where 0 indicates cells that are not allowed to change
```

```r
neighb an object of class NeighbRasterStack
```

```r
rules matrix with land use change decision rules
```

```r
nb.rules numeric with neighbourhood decision rules
```

```r
order numeric vector of land use categories in the order that change should be allocated. See Details
```

```r
params list with model parameters
```

```r
output either a RasterStack containing output maps or NULL
```

**Details**

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

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

**Value**

An OrderedModel object.

**References**


**See Also**

`OrderedModel-class`, `allocate`

**Examples**

```r
## see lulcc-package examples
```
OrderedModel-class

Class OrderedModel

Description

An S4 class to represent inputs to the Ordered allocation procedure

Slots

- `obs` an ObsLulcRasterStack object
- `ef` an ExpVarRasterList object
- `models` a PredictiveModelList object
- `time` numeric vector of timesteps over which simulation will occur
- `demand` matrix containing demand scenario
- `hist` RasterLayer showing land use history or NULL
- `mask` RasterLayer showing masked areas or NULL
- `neighb` NeighbRasterStack object or NULL
- `categories` numeric vector of land use categories
- `labels` character vector corresponding to categories
- `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
- `output` RasterStack containing simulated land use maps or NULL

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


See Also

caret::createDataPartition

Examples

```r
## Not run:

## Plum Island Ecosystems

## Load observed land use maps
obs <- ObsLULCRasterStack(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[[1]], size=0.1, spatial=FALSE)
names(part)

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

References


See Also

ROCR::prediction, ROCR::performance
Create a PerformanceList object

Description

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

Usage

PerformanceList(pred, measure, x.measure = "cutoff", ...)

Arguments

- pred: an object of class PredictionList
- 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 PerformanceList object.

References


See Also

performance, PredictionList

Examples

## see lulcc-package examples
PerformanceList-class

Class PerformanceList

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 PredictionList object supplied to the constructor function.
- auc: numeric vector containing the area under the 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.

pie

Land use change dataset for Plum Island Ecosystem

Description


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

plot

*Plot method for objects based on Raster* data

**Description**

Plot `lulcc` objects based on Raster* data

**Usage**

```r
## S3 method for class 'ObsLulcRasterStack'
plot(x, y, ...)

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

## S3 method for class 'ThreeMapComparison'
plot(x, y, category, factors, ...)

## S4 method for signature 'ObsLulcRasterStack,ANY'
plot(x, y, ...)

## S4 method for signature 'Model,ANY'
plot(x, y, ...)

## S4 method for signature 'ThreeMapComparison,ANY'
plot(x, y, category, factors, ...)
```

**Arguments**

- `x`: an object from `lulcc` containing Raster data
- `y`: not used
- `...`: additional arguments to `rasterVis::levelplot`
- `category`: numeric
- `factors`: numeric

**Value**

A trellis object.

**See Also**

`rasterVis::levelplot`
Examples

```r
## see lulcc-package examples
```

---

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

---

**Description**

Plot an AgreementBudget object.

**Usage**

```r
## S3 method for class 'AgreementBudget'
plot(x, y, from, to,
     col = RColorBrewer::brewer.pal(5, "Set2")(5, "Set2")(5, "Set2")(5, "Set2"),
     key = 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
```

**Arguments**

- `x`    an AgreementBudget object
- `y`    not used
- `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.
plot.FigureOfMerit

References


See Also

AgreementBudget, lattice::xyplot

Examples

```r
## see lulcc-package examples
```

---

Plot method for FigureOfMerit objects

Description

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

Usage

```r
## S3 method for class 'FigureOfMerit'
plot(x, y, ..., from, to,
    col = RColorBrewer::brewer.pal(8, "Set2"), type = "b", key, scales, xlab, ylab)

## S4 method for signature 'FigureOfMerit,ANY'
plot(x, y, ..., from, to,
    col = RColorBrewer::brewer.pal(8, "Set2"), type = "b", key, scales, xlab, ylab)
```

Arguments

- `x` a FigureOfMerit object
- `y` not used
- `...` additional arguments to lattice::xyplot
- `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
Plot method for PerformanceList objects

Description

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

Usage

```r
## S3 method for class 'PerformanceList'
plot(x, y, multipanel = TRUE, type = "l",
     abline = list(c(0, 1), col = "grey"),
     col = RColorBrewer::brewer_pal(9, "Set1"),
     key.args = NULL, ...)

## S4 method for signature 'list,ANY'
plot(x, y, 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 PerformanceList object or a list of these. If a list is provided it must be named.
- **y**: not used

Value

A trellis object.

See Also

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

Examples

```r
## see lulcc-package examples
```
predict.PredictiveModelList

multipanel logical. If TRUE, create a trellis plot where the number of panels equals the number of PerformanceList objects. Otherwise, create a single plot for each PerformanceList 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

PerformanceList, lattice::xyplot

Examples

## see lulcc-package examples

```r
predict.PredictiveModelList

Predict location suitability
```

Description

Estimate location suitability with predictive models.

Usage

```r
## S3 method for class 'PredictiveModelList'
predict(object, newdata, data.frame = FALSE, ...)

## S4 method for signature 'PredictiveModelList'
predict(object, newdata, data.frame = FALSE, ...)
```

Arguments

- object a PredictiveModelList object
- newdata data.frame containing new data
- data.frame logical indicating whether the function should return a matrix (default) or data.frame
- ... additional arguments to predict methods
predict.PredictiveModelList

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.

See Also

Model fitting, allocate

Examples

```
## Not run:

## Sibuyan Island

## load observed land use data
obs <- ObsLulcRasterStack(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 <- ExpVarRasterList(x=sibuyan$maps, pattern="ef")

## separate data into training and testing partitions
part <- partition(x=obs[[1]], size=0.1, spatial=TRUE)
train.data <- getPredictiveModelInputData(obs=obs, ef=ef, cells=part["train"])
all.data <- getPredictiveModelInputData(obs=obs, ef=ef, cells=part["all"])

## get glm.models from data
forms <- list(Forest ~ ef_001+ef_002+ef_003+ef_004+ef_005+ef_006+ef_007+ef_008+ef_010+ef_012,
  Coconut ~ ef_001+ef_002+ef_005+ef_007+ef_008+ef_009+ef_010+ef_011+ef_012,
  Grass~ef_001+ef_002+ef_004+ef_005+ef_007+ef_008+ef_009+ef_010+ef_011+ef_012+ef_013,
  Rice~ef_009+ef_010+ef_011,
  Other=1)

glm.models <- glmModels(formula=forms, family=binomial, data=train.data, obs=obs)

## create suitability maps
suitability.maps <- predict(object=glm.models, newdata=all.data, data.frame=TRUE)
points <- rasterToPoints(obs[[1]], spatial=TRUE)
suitability.maps <- SpatialPointsDataFrame(coords=points, data=suitability.maps)
r <- stack(rasterize(x=suitability.maps, y=obs[[1]], field=names(suitability.maps)))
plot(r)

## library(rasterVis)
## levelplot(r)
```
PredictionList

Create a PredictionList object

Description

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

Usage

PredictionList(models, newdata, ...)

Arguments

models a PredictiveModelList object
newdata a data.frame containing new data
... additional arguments to ROCR::prediction

Value

A PredictionList object.

References


See Also

link{PerformanceList}, ROCR::prediction

Examples

## see lulcc-package examples
PredictionList-class

Class PredictionList

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 PredictiveModelList 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

PredictiveModelList-class

Class PredictiveModelList

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,ExpVarRasterList,Raster-method

Resample maps in ExpVarRasterList object or list

Description

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

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

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

Arguments

- `x`: an `ExpVarRasterList` 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 `ExpVarRasterList` object or list, depending on `x`.

See Also

`ExpVarRasterList`, `raster::resample`

Examples

```r
## Not run:

## Plum Island Ecosystems

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

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

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

## End(Not run)
```
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.

Value
A matrix.

Examples
```r
## Sibuyan Island

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

dmd <- approxExtrapDemand(obs, tout=0:14)
apply(dmd, 1, sum)

## artificially perturb for illustration purposes
dmd <- dmd * runif(1)
apply(dmd, 1, sum)

## use roundSum to correct demand scenario
ncell <- length(which(!is.na(getValues(sibuyan$maps$lu_sib_1997)))
ncell
```
```r

```dmd <- roundSum(dmd, ncell=ncell)
apply(dmd, 1, sum)
```
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


Examples

data(sibuyan)
**Description**

Extract a subset of objects from container classes such as `ExpVarRasterList`, `PredictiveModellist`, `PredictionList` and `PerformanceList`.

**Usage**

```r
## S4 method for signature 'ExpVarRasterList'
subset(x, subset, ...)

## S4 method for signature 'PredictiveModellist'
subset(x, subset, ...)

## S4 method for signature 'PerformanceList'
subset(x, subset, ...)

## S4 method for signature 'PredictionList'
subset(x, subset, ...)
```

**Arguments**

- `x`: an object of class `ExpVarRasterList`, `PredictiveModellist`, `PredictionList` or `PerformanceList`
- `subset`: integer or character indicating the objects to be extracted
- `...`: additional arguments (none)

**Examples**

```r
## Sibuyan Island

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

summary(obs)
obs <- subset(obs, subset=names(obs)[1])
summary(obs)

## load explanatory variables
ef <- ExpVarRasterList(x=sibuyan$maps, pattern="ef")
```
summary

summary(ef)
  ef <- subset(ef, subset=1:5)
  summary(ef)

summary

Summary

Description

Summarise lulcc objects containing Raster* data or predictive models

Usage

summary(object, ...)

## S4 method for signature 'ObsLulcRasterStack'
summary(object, ...)

## S4 method for signature 'ExpVarRasterList'
summary(object, ...)

## S4 method for signature 'NeighbRasterStack'
summary(object, ...)

## S4 method for signature 'PredictiveModellist'
summary(object, ...)

## S4 method for signature 'Model'
summary(object, ...)

Arguments

object  an object belonging to one of the classes in lulcc
...
additional arguments (none)

Value

A matrix, data.frame or list
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**

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

## S4 method for signature 'Model,ANY,ANY'

```r
ThreeMapComparison(x, x1, y1, factors, timestep, ...)  
```

## S4 method for signature 'RasterLayer,RasterLayer,RasterLayer'

```r
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.
- **...**: additional arguments to `raster::aggregate`.
- **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.

**Value**

A `ThreeMapComparison` object.
References


See Also

AgreementBudget, FigureOfMerit, raster::aggregate

Examples

```r
## see lulcc-package examples
```

---

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
- `maps`: list of RasterStack objects containing land use maps at different resolutions
- `categories`: numeric vector of land use categories
- `labels`: character vector corresponding to categories

References

**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**

```r
## Sibuyan Island

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

total(x=obs)
total(x=obs[[1]])
total(x=obs[[2]])
```
Index

*Topic datasets

pie, 39
sibuyan, 50

[[,CategoryLabel,ANY,ANY-method (Extract by index), 24
[[,ExpVarRasterList,ANY,ANY-method (Extract by index), 24

aggregate, 54, 55
AgreementBudget, 7, 41, 42, 55
AgreementBudget, RasterLayer-method (AgreementBudget), 7
AgreementBudget, ThreeMapComparison-method (AgreementBudget), 7
AgreementBudget-class, 8
allocate, 9, 10, 17, 19, 33, 34, 45
allocate, CluesModel-method (allocate), 9
allocate, OrderedModel-method (allocate), 9
allow, 10, 12
allowNeighb, 11, 12, 30
approxExtrap, 13, 14
approxExtrapDemand, 13
as.data.frame, 15, 27
as.data.frame, ExpVarRasterList-method (as.data.frame.ExpVarRasterList), 14
as.data.frame, ObsLulcRasterStack-method (as.data.frame.ExpVarRasterList), 14
as.data.frame, ExpVarRasterList, 14
as.data.frame, ObsLulcRasterStack (as.data.frame.ExpVarRasterList), 14

c.PredictiveModelList, 16
c.CategoryLabel, 17
CluesModel, 9, 17
CluesModel, ObsLulcRasterStack, ExpVarRasterList, PredictiveModelList-method (CluesModel), 17

CluesModel-class, 19
compareAUC, 20
compareAUC, list-method (compareAUC), 20
compareAUC, PredictionList-method (compareAUC), 20
createDataPartition, 35, 36
crosstab, 21, 22
crossTabulate, 21
crossTabulate, ObsLulcRasterStack, ANY-method (crossTabulate), 21
crossTabulate, RasterLayer, RasterLayer-method (crossTabulate), 21
ExpVarRasterList, 14, 15, 22, 27, 48
ExpVarRasterList, character, character-method (ExpVarRasterList), 22
ExpVarRasterList, character-method (ExpVarRasterList), 22
ExpVarRasterList, list, character-method (ExpVarRasterList), 22
ExpVarRasterList, list-method (ExpVarRasterList), 22
ExpVarRasterList, missing, character-method (ExpVarRasterList), 22
ExpVarRasterList, missing-method (ExpVarRasterList), 22
ExpVarRasterList, RasterStack, character-method (ExpVarRasterList), 22
ExpVarRasterList, RasterStack-method (ExpVarRasterList), 22
ExpVarRasterList-class, 24
extract, 15, 27
Extract by index, 24

FigureOfMerit, 8, 25, 43, 55
FigureOfMerit, RasterLayer-method (FigureOfMerit), 25
FigureOfMerit, ThreeMapComparison-method (FigureOfMerit), 26

57
INDEX

focal, 29–31
getPredictiveModelInputData, 27
glm, 28
glmModels (Model fitting), 27
layerize, 15
levelplot, 40
lulcc-package, 3
Model fitting, 27
Model-class, 28
NeighbRasterStack, 12, 29
NeighbRasterStack, RasterLayer, ANY, NeighbRasterStack-method (NeighbRasterStack), 29
NeighbRasterStack, RasterLayer, list, ANY-method (NeighbRasterStack), 29
NeighbRasterStack, RasterLayer, matrix, ANY-method (NeighbRasterStack), 29
NeighbRasterStack-class, 30
ObsLulcRasterStack, 14, 15, 22, 27, 31
ObsLulcRasterStack, character, character-method (ObsLulcRasterStack), 31
ObsLulcRasterStack, list, character-method (ObsLulcRasterStack), 31
ObsLulcRasterStack, missing, character-method (ObsLulcRasterStack), 31
ObsLulcRasterStack, RasterLayer, ANY-method (ObsLulcRasterStack), 31
ObsLulcRasterStack, RasterStack, ANY-method (ObsLulcRasterStack), 31
ObsLulcRasterStack-class, 32
OrderedModel, 9, 33
OrderedModel, ObsLulcRasterStack, ExpVarRasterList-method (OrderedModel), 33
OrderedModel-class, 35
panel.xyplot, 43, 44
partition, 15, 27, 35
performance, 21, 37, 37, 38
performance, list-method (performance), 37
PerformanceList, 38, 43, 44, 46
PerformanceList-class, 39
pie, 39
plot, 40
plot, AgreementBudget, ANY-method (plot.AgreementBudget), 41
plot, FigureOfMerit, ANY-method (plot.FigureOfMerit), 42
plot, list, ANY-method (plot.PerformanceList), 43
plot, Model, ANY-method (plot), 40
plot, ObsLulcRasterStack, ANY-method (plot), 40
plot, ThreeMapComparison, ANY-method (plot), 40
plot, AgreementBudget, 8, 41
plot, FigureOfMerit, 26, 42
plot, Model (plot), 40
plot, ObsLulcRasterStack (plot), 40
plot, PerformanceList, 43
plot, ThreeMapComparison (plot), 40
predict, PredictiveModelList-method (predict.PredictiveModelList), 44
predict, PredictiveModelList, 44
prediction, 20, 37, 38, 46
PredictionList, 20, 21, 38, 46
PredictionList-class, 47
PredictiveModelList-class, 47
randomForest, 28
randomForestModels (Model fitting), 27
raster, 23, 31
resample, 47, 48
resample, ExpVarRasterList, Raster-method, 47
resample, list, Raster-method (resample, ExpVarRasterList, Raster-method), 47
roundSum, 49
rpart, 28
rpartModels (Model fitting), 27
show, ExpVarRasterList-method, 50
show, Model-method (show, ExpVarRasterList-method), 50
show, PerformanceList-method (show, ExpVarRasterList-method), 50
show, PredictionList-method (show, ExpVarRasterList-method), 50
show, PredictiveModelList-method (show, ExpVarRasterList-method), 50
INDEX

50
show, ThreeMapComparison-method
  (show, ExpVarRasterList-method), 50
sibuyan, 50
stack, 23, 32
subset, ExpVarRasterList-method, 52
subset, PerformanceList-method
  (subset, ExpVarRasterList-method), 52
subset, PredictionList-method
  (subset, ExpVarRasterList-method), 52
subset, PredictiveModellist-method
  (subset, ExpVarRasterList-method), 52
summary, 53
summary, ExpVarRasterList-method
  (summary), 53
summary, Model-method (summary), 53
summary, NeighbRasterStack-method
  (summary), 53
summary, ObsLulcRasterStack-method
  (summary), 53
summary, PredictiveModellist-method
  (summary), 53

ThreeMapComparison, 8, 26, 54
ThreeMapComparison, Model, ANY, ANY-method
  (ThreeMapComparison), 54
ThreeMapComparison, RasterLayer, RasterLayer, RasterLayer-method
  (ThreeMapComparison), 54
ThreeMapComparison-class, 55
total, 56

xyplot, 41–44