Package ‘lulcc’

July 15, 2016

Title Land Use Change Modelling in R

Version 1.0.0

Author Simon Moulds <simon.moulds10@imperial.ac.uk>

Maintainer Simon Moulds <simon.moulds10@imperial.ac.uk>

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-PredictiveModelList.R'
'class-NeighbRasterStack.R'
'class-LulcRasterStack.R'
'class-ExpVarRasterStack.R'
'class-Model.R'
'class-ThreeMapComparison.R'
'AgreementBudget.R'
'ExpVarRasterStack.R'
'FigureOfMerit.R'
'LulcRasterStack.R'
'Model.R'
'NeighbRasterStack.R'
'class-PerformanceList.R'
'PerformanceList.R'
'class-PredictionList.R'
'PredictionList.R'

'as.data.frame.R'
'PredictiveModelList.R'
'ThreeMapComparison.R'
'allocate.R'
'allow.R'
'approxExtrapDemand.R'
'c.PredictiveModelList.R'
'compareAUC.R'
'crop.R'
'index.R'
'crossTabulate.R'
'data.R'
'getPredictiveModelInputData.R'
'length.R'
'lulcc-package.R'
'names.R'
'partition.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 2015-08-19 11:50:07

R topics documented:

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

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 discrete land use change models: an implementation of the Change in Land Use and its Effects at Small Regional extent (CLUE-S) model (Verburg et al., 2002) (class `CluesModel`) and an ordered procedure based on the algorithm described by Fuchs et al. (2013) but modified to allow stochastic transitions (class `OrderedModel`). An implementation of the continuous land use change model CLUE (Veldkamp and Fresco, 1996; Verburg and Bouma, 1999) is also included.

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.

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 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.

Veldkamp, A., & Fresco, L. O. (1996). CLUE-CR: an integrated multi-scale model to simulate land use change scenarios in Costa Rica. Ecological modelling, 91(1), 231-248.

Verburg, P.H., & Bouma, J. (1999). Land use change under conditions of high population pressure: the case of Java. Global environmental change, 9(4), 303-312.

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:

## Plum Island Ecosystems

data(pie)

## Observed maps
lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))
plot(lu)

crossTabulate(x=lu, times=c(0,14))

## Explanatory variables
idx <- data.frame(var=c("ef_001","ef_002","ef_003"),
                  yr=c(0,0,0),
                  dynamic=c(FALSE,FALSE,FALSE))
idx

ef <- ExpVarRasterStack(x=stack(pie[4:6]), index=idx)

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

## predictive modelling
forest.form <- as.formula("Forest ~ ef_001 + ef_002")
built.form <- as.formula("Built ~ ef_001 + ef_002 + ef_003")
other.form <- as.formula("Other ~ ef_001 + ef_002")

library(randomForest)
library(rpart)

forest.glm <- glm(forest.form, family=binomial, data=train.data)
forest.rprt <- rpart(forest.form, data=train.data)
forest.rf <- randomForest(forest.form, method="class", data=train.data)

built.glm <- glm(built.form, family=binomial, data=train.data)
built.rprt <- rpart(built.form, data=train.data)
built.rf <- randomForest(built.form, method="class", data=train.data)

other.glm <- glm(other.form, family=binomial, data=train.data)
other.rprt <- rpart(other.form, data=train.data)
other.rf <- randomForest(other.form, method="class", data=train.data)
```

```
## Binomial logistic regression
glm.mods <- PredictiveModelList(list(forest.glm, built.glm, other.glm),
                                categories=lu@categories,
                                labels=lu@labels)

## Recursive partitioning and regression trees
rprt.mods <- PredictiveModelList(list(forest.rprt, built.rprt, other.rprt),
                                 categories=lu@categories,
                                 labels=lu@labels)

## Random forests
rf.mods <- PredictiveModelList(list(forest.rf, built.rf, other.rf),
                               categories=lu@categories,
                               labels=lu@labels)

test.data <- getPredictiveModelInputData(lu=lu,
                                         ef=ef,
                                         cells=part[["test"]],
                                         t=0)

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

rprt.pred <- PredictionList(models=rprt.mods, newdata=test.data)
rprt.perf <- PerformanceList(pred=rprt.pred, measure="rch")

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

p <- plot(list(glm=glm.perf, rpart=rprt.perf, rf=rf.perf))

## Probability maps
all.data <- as.data.frame(x=ef, cells=part[["all"]])
probmaps <- predict(object=glm.mods,
                    newdata=all.data,
                    data.frame=TRUE)

points <- rasterToPoints(lu[[1]], spatial=TRUE)
probmaps <- SpatialPointsDataFrame(points, probmaps)
probmaps <- rasterize(x=probmaps, y=lu[[1]],
                      field=names(probmaps))

p <- levelplot(probmaps, layout=c(2,2), margin=FALSE)

## Demand scenario
dmd <- approxExtrapDemand(lu=lu, tout=0:14)

## CLUE-S modelling
clues.model <- CluesModel(observed.lulc=lu,
                          explanatory.variables=ef,
                          predictive.models=glm.mods,
                          time=0:14,
                          demand=dmd,
                          history=NULL,
                          mask=NULL,
                          neighbourhood=NULL,
```

```
                          transition.rules=matrix(data=1, nrow=3, ncol=3),
                          neighbourhood.rules=NULL,
                          elasticity=c(0.2,0.2,0.2),
                          iteration.factor=0.00001,
                          max.iteration=1000,
                          max.difference=5,
                          ave.difference=5)

clues.result <- allocate(clues.model)

## Ordered modelling
ordered.model <- OrderedModel(observed.lulc=lu,
                              explanatory.variables=ef,
                              predictive.models=glm.mods,
                              time=0:14,
                              demand=dmd,
                              transition.rules=matrix(data=1, 3, 3),
                              order=c(2,1,3))

ordered.result <- allocate(ordered.model, stochastic=FALSE)

## Validation

clues.tabs <- ThreeMapComparison(x=lu[[1]],
                                   x1=lu[[3]],
                                   y1=clues.result[[15]],
                                   factors=2^(1:8),
                                   categories=lu@categories,
                                   labels=lu@labels)

clues.agr <- AgreementBudget(x=clues.tabs)
clues.fom <- FigureOfMerit(x=clues.tabs)
ordered.tabs <- ThreeMapComparison(x=lu[[1]],
                                   x1=lu[[3]],
                                   y1=ordered.result[[15]],
                                   factors=2^(1:8),
                                   categories=lu@categories,
                                   labels=lu@labels)

ordered.agr <- AgreementBudget(x=ordered.tabs)
ordered.fom <- FigureOfMerit(x=ordered.tabs)

p1 <- plot(clues.agr, from=1, to=2)
p2 <- plot(ordered.agr, from=1, to=2)

agr.p <- c("CLUE-S"=p1, Ordered=p2, layout=c(1,2))
agr.p

p1 <- plot(clues.fom, from=1, to=2)
p2 <- plot(ordered.fom, from=1, to=2)

fom.p <- c("CLUE-S"=p1, Ordered=p2, layout=c(1,2))
fom.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 passed to ThreeMapComparison

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`, `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 spatial allocation of land use change using different models. Currently the function provides an implementation of the Change in Land Use and its Effects (CLUE; Veldkamp and Fresco, 1996, Verburg et al., 1996), CLUE at Small regional extent (CLUE-S; 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 ClueModel
allocate(model, ...)

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

Arguments

`model`	an object inheriting from class `Model`
`stochastic`	logical
`...`	additional arguments for specific methods

Value

LulcRasterStack.

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.

Veldkamp, A., & Fresco, L. O. (1996). CLUE-CR: an integrated multi-scale model to simulate land use change scenarios in Costa Rica. Ecological modelling, 91(1), 231-248.

Verburg, P.H., & Bouma, J. (1999). Land use change under conditions of high population pressure: the case of Java. Global environmental change, 9(4), 303-312.

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`

Examples

```
## 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

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 maps
lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

## get land use values
x <- getValues(lu[[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(lu=lu, 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=lu@categories,
               cd=cd,
               rules=rules)

## create raster showing cells that are allowed to change from forest to built
r <- lu[[1]]
r[!is.na(r)] <- allow[,2]
r[lu[[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

```
## Plum Island Ecosystems

## load observed land use maps
lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              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=lu[[1]], weights=w, categories=1)

## only allow change to forest within neighbourhood of current forest cells
## note that rules can be any value between zero (less restrictive) and one
## (more restrictive)
nb.allow <- allowNeighb(neighb=nb,
                        x=lu[[1]],
                        categories=lu@categories,
                        rules=0.5)

## create raster showing cells allowed to change to forest
r <- lu[[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(lu, ...)

## S4 method for signature LulcRasterStack
approxExtrapDemand(lu, tout, ...)

## S4 method for signature DiscreteLulcRasterStack
approxExtrapDemand(lu, tout, ...)
```

Arguments

`lu`	an LulcRasterStack 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::approxExtrap`

Examples

```
## Plum Island Ecosystems

## load observed land use maps
lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

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

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

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

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

as.data.frame.ExpVarRasterStack
Coerce objects to data.frame

Description

This function extracts data from all raster objects in LulcRasterStack or ExpVarRasterStack objects for a specified timestep.

Usage

```
## S3 method for class ExpVarRasterStack
as.data.frame(x, row.names = NULL,
  optional = FALSE, cells, t, ...)

## S3 method for class DiscreteLulcRasterStack
as.data.frame(x, row.names = NULL,
  optional = FALSE, cells, t, ...)

## S3 method for class ContinuousLulcRasterStack
as.data.frame(x, row.names = NULL,
  optional = FALSE, cells, t, ...)

## S4 method for signature ExpVarRasterStack
as.data.frame(x, row.names = NULL,
  optional = FALSE, cells, t, ...)

## S4 method for signature DiscreteLulcRasterStack
as.data.frame(x, row.names = NULL,
  optional = FALSE, cells, t, ...)

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

Arguments

`x`	an ExpVarRasterStack or LulcRasterStack 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 a DiscreteLulcRasterStack 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, LulcRasterStack, ExpVarRasterStack, partition

Examples

```
## Not run:

## Plum Island Ecosystems

## load observed land use maps
lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

## explanatory variables
idx <- data.frame(var=c("ef_001","ef_002","ef_003"),
                  yr=c(0,0,0),
                  dynamic=c(FALSE,FALSE,FALSE))

ef <- ExpVarRasterStack(x=stack(pie[4:6]), index=idx)

## separate data into training and testing partitions
part <- partition(x=lu[[1]], size=0.1, spatial=TRUE)
df1 <- as.data.frame(x=lu, cells=part[["all"]], t=0)
df2 <- as.data.frame(x=ef, cells=part[["all"]], t=0)
df <- cbind(df1,df2)


## End(Not run)
```

---

c.PredictiveModelList *Merge PredictiveModelList objects*

Description

Combine different PredictiveModelList objects into one

Usage

```
## 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

```
## Not run:

## Plum Island Ecosystems

## load data
data(pie)

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

## explanatory variables
ef <- ExpVarRasterStack(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)
glm.models


## End(Not run)
```

`clue` *CLUE*

Description

Allocate land use change using the CLUE algorithm.

Usage

```
clue(lu0.vals, regr, demand, elasticity, change.rule, min.elasticity,
  max.elasticity, min.change, max.change, min.value, max.value, max.iteration,
  max.difference, cell.area, ncell, ncode)
```

Arguments

`lu0.vals`	matrix containing non-NA values from `lu0`
`regr`	matrix containing...
`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 land use map, the second row should be the number of cells to allocate in the subsequent time point
`elasticity`	Initial elasticity value. Default is 0.1
`change.rule`	numeric vector specifying for each land use whether change is allowed in either direction (0), allowed in the direction of demand only (-1) or not allowed (1)
`min.elasticity`	Minimum elasticity value. Default is 0.001
`max.elasticity`	Maximum elasticity value. Default is 1.5
`min.change`	numeric vector indicating for each land use the minimum amount of change that is allowed to occur in one time step
`max.change`	numeric vector indicating for each land use the maximum amount of change that is allowed to occur in one time step
`min.value`	numeric vector indicating the minimum fraction of each land use in a given cell
`max.value`	numeric vector indicating the maximum fraction of each land use in a given cell
`max.iteration`	The maximum number of iterations allowed at each time step
`max.difference`	The maximum allowable difference between demand and allocated area
`cell.area`	The area of each grid cell in the study region, which should have the same units as the demand
`ncell`	number of cells considered for change (equal to the length of `lu0.vals`
`ncode`	number of land use categories under consideration
`...`	additional arguments (none)

Value

numeric vector with updated land use values.

Examples

```
## See lulcc-package examples
```

ClueModel — *Create a ClueModel object*

Description

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

Usage

```
ClueModel(observed.lulc, explanatory.variables, predictive.models, time, demand,
  elasticity = 0.1, change.rule, min.elasticity = 0.001,
  max.elasticity = 1.5, min.value, max.value, min.change, max.change,
  max.iteration = 1000, max.difference, cell.area)
```

Arguments

`observed.lulc`	an LulcRasterStack
`explanatory.variables`	an ExpVarRasterStack object
`predictive.models`	a PredictiveModelList 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)
`elasticity`	Initial elasticity value. Default is 0.1
`change.rule`	numeric vector specifying for each land use whether change is allowed in either direction (0), allowed in the direction of demand only (-1) or not allowed (1)
`min.elasticity`	Minimum elasticity value. Default is 0.001
`max.elasticity`	Maximum elasticity value. Default is 1.5
`min.value`	numeric vector indicating the minimum fraction of each land use in a given cell
`max.value`	numeric vector indicating the maximum fraction of each land use in a given cell
`min.change`	numeric vector indicating for each land use the minimum amount of change that is allowed to occur in one time step
`max.change`	numeric vector indicating for each land use the maximum amount of change that is allowed to occur in one time step
`max.iteration`	The maximum number of iterations allowed at each time step
`max.difference`	The maximum allowable difference between demand and allocated area
`cell.area`	The area of each grid cell in the study region, which should have the same units as the demand
`...`	additional arguments (none)

Value

A ClueModel object.

References

Veldkamp, A., & Fresco, L. O. (1996). CLUE-CR: an integrated multi-scale model to simulate land use change scenarios in Costa Rica. Ecological modelling, 91(1), 231-248.

Verburg, P.H., & Bouma, J. (1999). Land use change under conditions of high population pressure: the case of Java. Global environmental change, 9(4), 303-312.

See Also

`ClueModel-class`, `allocate`

Examples

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

ClueModel-class *Class ClueModel*

Description

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

Slots

`observed.lulc` a ContinuousLulcRasterStack object

`explanatory.variables` an ExpVarRasterStack object

`predictive.models` a PredictiveModelList object

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

`demand` matrix containing demand scenario

`elasticity` numeric

`change.rule` numeric

`min.elasticity` numeric

`max.elasticity` numeric

`min.value` numeric

`max.value` numeric

`min.change` numeric

`max.change` numeric

`max.iteration` numeric

`max.difference` numeric

`cell.area` numeric

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

clues *CLUE-S*

Description

Allocate land use change using the CLUE-S algorithm.

Usage

```
clues(lu0, lu0.vals, tprob, nb = NULL, nb.rules = NULL,
  transition.rules = NULL, hist.vals = NULL, mask.vals = NULL, demand,
  categories, elasticity, iteration.factor, max.iteration, max.difference,
  ave.difference, ...)
```

Arguments

`lu0`	RasterLayer showing initial land use
`lu0.vals`	numeric containing non-NA values from `lu0`
`tprob`	matrix with land use suitability values. Columns should correspond to `categories`, rows should correspond with `cells`
`nb`	neighbourhood map. See CluesModel
`nb.rules`	neighbourhood rules. See CluesModel documentation
`transition.rules`	transition rules. See CluesModel documentation
`hist.vals`	numeric vector detailing the number of consecutive time steps each cell has been allocated to its current land use
`mask.vals`	numeric vector containing binary values where 0 indicates cells that are not allowed to change
`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 land use map, the second row should be the number of cells to allocate in the subsequent time point
`categories`	numeric vector containing land use categories
`elasticity`	elasticity values. See CluesModel documentation
`iteration.factor`	iteration factor. See CluesModel documentation
`max.iteration`	The maximum number of iterations allowed at each time step
`max.difference`	The maximum allowable difference between demand and allocated area
`ave.difference`	The maximum allowable average difference across all land uses
`...`	additional arguments (none)

Value

numeric vector with updated land use values.

Examples

```
## See lulcc-package examples
```

`CluesModel` *Create a CluesModel object*

Description

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

Usage

```
CluesModel(observed.lulc, explanatory.variables, predictive.models, time,
  demand, history = NULL, mask = NULL, neighbourhood = NULL,
  transition.rules, neighbourhood.rules = NULL, elasticity,
  iteration.factor = 1e-05, max.iteration = 1000, max.difference = 5,
  ave.difference = 5, ...)
```

Arguments

`observed.lulc`	an LulcRasterStack
`explanatory.variables`	an ExpVarRasterStack object
`predictive.models`	a PredictiveModelList 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)
`history`	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
`neighbourhood`	an object of class NeighbRasterStack
`transition.rules`	matrix with land use change decision rules
`neighbourhood.rules`	numeric with neighbourhood decision rules
`elasticity`	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
`iteration.factor`	TODO,
`max.iteration`	The maximum number of iterations allowed at each time step
`max.difference`	The maximum allowable difference between demand and allocated area
`ave.difference`	The maximum allowable average difference across all land uses
`...`	additional arguments (none)

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 lulcc-package examples
```

ClueModel-class *Class ClueModel*

Description

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

Slots

`observed.lulc` an LulcRasterStack object

`explanatory.variables` an ExpVarRasterStack object

`predictive.models` a PredictiveModelList object

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

`demand` matrix containing demand scenario

`history` RasterLayer showing land use history or NULL

`mask` RasterLayer showing masked areas or NULL

`neighbourhood` NeighbRasterStack object or NULL

`transition.rules` matrix with land use change decision rules

`neighbourhood.rules` numeric with neighbourhood decision rules

`elasticity` numeric indicating elasticity to change (only required for

`iteration.factor` TODO

`max.iteration` TODO

`max.difference` TODO

`ave.difference` TODO

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

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

Description

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

Usage

```
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
`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 `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

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

`PredictionList`, `ROCR::performance`

Examples

```
## see PredictiveModelList examples
```

`ContinuousLulcRasterStack-class` *Class ContinuousLulcRasterStack*

Description

A virtual S4 class for observed land use 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`

`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`

`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 DiscreteLulcRasterStack,ANY
crossTabulate(x, y, times, ...)
```

Arguments

`x`	RasterLayer representing land use map from an earlier timestep or an LulcRasterStack 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 LulcRasterStack object
`categories`	numeric vector containing land use categories to consider. Not used if `x` is an LulcRasterStack object
`labels`	character vector (optional) with labels corresponding to `categories`. Not used if `x` is an LulcRasterStack object
`times`	numeric vector representing the time points of two land use maps from LulcRasterStack
`...`	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

LulcRasterStack, raster::crosstab

Examples

```
## Not run:

## Plum Island Ecosystems

## load observed land use maps
lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

crossTabulate(x=lu, times=c(0,6))
crossTabulate(x=lu, times=c(0,14))

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


## End(Not run)
```

DiscreteLulcRasterStack-class
Class DiscreteLulcRasterStack

Description

A virtual S4 class for observed land use 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

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`

`ExpVarRasterStack` *Create an ExpVarRasterStack object*

Description

Methods to create an ExpVarRasterStack object, which may be created from file or an existing Raster* object.

Usage

```
ExpVarRasterStack(x, ...)

## S4 method for signature character
ExpVarRasterStack(x, ...)

## S4 method for signature RasterStack
ExpVarRasterStack(x, index, ...)
```

Arguments

`x`	Raster* object
`index`	data.frame
`...`	additional arguments to `raster::stack`

Details

Inductive and deductive land use change models predict the allocation of change based on spatially explicit biophysical and socioeconomic covariates. These may be static, such as elevation or geology, or dynamic, such as maps of population density or road networks. To identify whether a covariable is static or dynamic a data frame is supplied to the ExpVarRasterStack constructor function with three columns: the first column specifies the name of the variable, the second column specifies the time point for which it is relevant and the third column specifies whether it is dynamic or not. Data frame rows should correspond to the individual layers of the RasterStack object containing the explanatory variables. If dynamic variables are used it is not necessary to supply a map for each time point in the simulation: during allocation the most recent map will automatically be selected.

Value

An ExpVarRasterStack object.

See Also

raster::stack

Examples

```
## Plum Island Ecosystems

idx <- data.frame(var=paste("ef_", formatC(1:3, width=3, flag=0)),
                  yr=rep(0,3),
                  dynamic=rep(FALSE,3))

ef <- ExpVarRasterStack(x=stack(pie[4:6]), index=idx)

## Sibuyan

idx <- data.frame(var=paste("ef_", formatC(1:13, width=3, flag=0)),
                  yr=rep(0,13),
                  dynamic=rep(FALSE,13))

ef <- ExpVarRasterStack(x=stack(sibuyan$maps[3:15]), index=idx)
```

ExpVarRasterStack-class *Class ExpVarRasterStack*

Description

An S4 class for explanatory variables.

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
- `index` data.frame TODO

Extract by index *Extract by index*

Description

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

Usage

```
## S4 method for signature DiscreteLulcRasterStack,ANY,ANY
x[[i, j, ...]]

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

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

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

## S4 method for signature PredictiveModelList,ANY,ANY
x[i, j, ..., drop = FALSE]

## S4 method for signature ContinuousLulcRasterStack,ANY,ANY
x[[i, j, ...]]
```

Arguments

x	An object of class DiscreteLulcRasterStack, ContinuousLulcRasterStack, PredictionList, PerformanceList, PredictiveModelList
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)
drop	logical. If TRUE the result is coerced to the lowest possible dimension

Examples

```
## Plum Island Ecosystems

lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

summary(lu[[1]])
summary(lu[[1:2]])

## Also see lulcc-package
```

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

`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. Column names correspond to the names of lu and ef.

Usage

```
getPredictiveModelInputData(lu, ef, cells, ...)
```

Arguments

`lu`	an LulcRasterStack object
`ef`	an ExpVarRasterStack 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`, `LulcRasterStack`, `ExpVarRasterStack`, `partition`

Examples

```
## Not run:

## Plum Island Ecosystems

lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

idx <- data.frame(var=paste("ef_", formatC(1:3, width=3, flag=0)),
                  yr=rep(0,3),
                  dynamic=rep(FALSE,3))

ef <- ExpVarRasterStack(x=stack(pie[4:6]), index=idx)

part <- partition(x=lu, size=0.1, spatial=TRUE, t=0)

train.data <- getPredictiveModelInputData(lu=lu,
                                          ef=ef,
                                          cells=part[["train"]],
                                          t=0)
dim(train.data)
names(train.data)


## End(Not run)
```

`LulcRasterStack` *Create an LulcRasterStack object*

Description

Methods to create an LulcRasterStack object, which may be created from file or an existing Raster* object

Usage

```
DiscreteLulcRasterStack(x, ...)

## S4 method for signature Raster
DiscreteLulcRasterStack(x, ...)

## S4 method for signature RasterStack
DiscreteLulcRasterStack(x, categories, labels, t)

ContinuousLulcRasterStack(x, ...)

## S4 method for signature Raster
ContinuousLulcRasterStack(x, ...)

## S4 method for signature RasterStack
ContinuousLulcRasterStack(x, 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
`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 and have the same non-NA cells. The location of non-NA cells in `LulcRasterStack` objects defines the region for subsequent analysis.

Value

An LulcRasterStack object.

See Also

`LulcRasterStack-class`, `raster::stack`

Examples

```
## Plum Island Ecosystems

lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))
lu

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

`LulcRasterStack-class` *Class LulcRasterStack*

Description

A virtual S4 class for observed land use 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`

`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`

`Model-class` *Virtual class Model*

Description

A virtual S4 class to represent land use change models.

`NeighbRasterStack` *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

```
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`
- `...` — additional arguments to `raster::focal`

Value

A NeighbRasterStack object.

See Also

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

Examples

```
## Plum Island Ecosystems

lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              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=lu[[1]],
                         categories=c(1,2,3),
                         weights=list(w1,w2,w3))

## update nb2 for 1991
nb2 <- NeighbRasterStack(x=lu[[2]],
                         neighb=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`

`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`

`calls` list containing each call to `raster::focal`

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

ordered *Ordered allocation*

Description

Allocate land use change using the ordered algorithm.

Usage

```
ordered(lu0, lu0.vals, tprob, nb = NULL, nb.rules = NULL,
  transition.rules = NULL, hist.vals = NULL, mask.vals = NULL, demand,
  categories, order, stochastic)
```

Arguments

`lu0`	RasterLayer showing initial land use
`lu0.vals`	numeric containing non-NA values from `lu0`
`tprob`	matrix with land use suitability values. Columns should correspond to `categories`, rows should correspond with `cells`
`nb`	neighbourhood map. See CluesModel

`nb.rules` neighbourhood rules. See CluesModel documentation

`transition.rules` transition rules. See CluesModel documentation

`hist.vals` numeric vector detailing the number of consecutive time steps each cell has been allocated to its current land use

`mask.vals` numeric vector containing binary values where 0 indicates cells that are not allowed to change

`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 land use map, the second row should be the number of cells to allocate in the subsequent time point

`categories` numeric vector containing land use categories

`order` numeric vector of land use categories in the order that change should be allocated

`stochastic` Logical indicating whether or not the allocation routine should be run in stochastic mode

`...` additional arguments (none)

Value

numeric vector with updated land use values.

Examples

```
## See lulcc-package examples
```

`OrderedModel` *Create a OrderedModel object*

Description

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

Usage

```
OrderedModel(observed.lulc, explanatory.variables, predictive.models, time,
  demand, history = NULL, mask = NULL, neighbourhood = NULL,
  transition.rules, neighbourhood.rules = NULL, order, ...)
```

Arguments

`observed.lulc` an LulcRasterStack

`explanatory.variables` an ExpVarRasterStack object

`predictive.models` a PredictiveModelList 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)
`history`	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
`neighbourhood`	an object of class NeighbRasterStack
`transition.rules`	matrix with land use change decision rules
`neighbourhood.rules`	numeric with neighbourhood decision rules
`order`	numeric vector of land use categories in the order that change should be allocated
`...`	additional arguments (none)

Value

A 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 lulcc-package examples
```

`OrderedModel-class` *Class OrderedModel*

Description

An S4 class to represent inputs to the Ordered allocation procedure

Slots

`observed.lulc` an LulcRasterStack object

`explanatory.variables` an ExpVarRasterStack object

`predictive.models` a PredictiveModelList object

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

`demand` matrix containing demand scenario

`history` RasterLayer showing land use history or NULL `mask` RasterLayer showing masked areas or NULL

`neighbourhood` NeighbRasterStack object or NULL

`transition.rules` matrix with land use change decision rules

`neighbourhood.rules` numeric with neighbourhood decision rules

`order` numeric vector of land use categories in the order that change should be allocated

`categories` numeric vector of land use categories

`labels` character vector corresponding to `categories`

`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, ...)

## S4 method for signature RasterLayer
partition(x, size = 0.5, spatial = TRUE, ...)

## S4 method for signature DiscreteLulcRasterStack
partition(x, size = 0.5, spatial = TRUE,
  t, ...)

## S4 method for signature ContinuousLulcRasterStack
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
`t`	numeric corresponding to one of the time points for which a land use map is available.
`...`	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

```
## Not run:

## Plum Island Ecosystems

lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

part <- partition(x=lu, size=0.05, spatial=TRUE, t=0)

plot(lu[[1]])
points(part[["train"]])


## End(Not run)
```

PerformanceList *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

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

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

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.

Examples

```
data(pie)
```

---

plot *Plot method for objects based on Raster* data*

Description

Plot `lulcc` objects based on Raster* data

Usage

```
## S3 method for class ContinuousLulcRasterStack
plot(x, y, ...)

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

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

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

## S4 method for signature DiscreteLulcRasterStack,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
`category`	numeric
`factors`	numeric
`...`	additional arguments to `rasterVis::levelplot`

Value

A trellis object.

See Also

`rasterVis::levelplot`

Examples

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

---

plot.AgreementBudget *Plot method for AgreementBudget objects*

Description

Plot an `AgreementBudget` object.

Usage

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

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

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.

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 lulcc-package examples
```

---

plot.FigureOfMerit *Plot method for FigureOfMerit objects*

Description

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

Usage

```
## 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
`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 lulcc-package examples
```

---

plot.PerformanceList *Plot method for PerformanceList objects*

Description

Plot the 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

```
## 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
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
```

---

predict.PredictiveModelList
Predict allocation suitability

Description

Estimate allocation suitability with predictive models.

Usage

```
## S3 method for class 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

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

`predict`, `allocate`

Examples

```
## Not run:

## Plum Island Ecosystems

lu <- DiscreteLulcRasterStack(x=stack(pie[1:3]),
                              categories=c(1,2,3),
                              labels=c("Forest","Built","Other"),
                              t=c(0,6,14))

idx <- data.frame(var=c("ef_001","ef_002","ef_003"),
                  yr=c(0,0,0),
                  dynamic=c(FALSE,FALSE,FALSE))
ef <- ExpVarRasterStack(x=stack(pie[4:6]), index=idx)
```

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

forest.form <- as.formula("Forest ~ ef_001 + ef_002")
built.form <- as.formula("Built ~ ef_001 + ef_002 + ef_003")
other.form <- as.formula("Other ~ ef_001 + ef_002")

forest.glm <- glm(forest.form, family=binomial, data=train.data)
built.glm <- glm(built.form, family=binomial, data=train.data)
other.glm <- glm(other.form, family=binomial, data=train.data)
glm.mods <- PredictiveModelList(list(forest.glm, built.glm, other.glm),
                                categories=lu@categories,
                                labels=lu@labels)

all.data <- as.data.frame(x=ef, cells=part[["all"]])
probmaps <- predict(object=glm.mods,
                    newdata=all.data,
                    data.frame=TRUE)

points <- rasterToPoints(lu[[1]], spatial=TRUE)
probmaps <- SpatialPointsDataFrame(points, probmaps)
probmaps <- rasterize(x=probmaps, y=lu[[1]],
                      field=names(probmaps))
plot(probmaps)


## End(Not run)
```

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

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

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` *Create PredictiveModelList object*

Description

Crete an object of class PredictiveModelList.

Usage

```
PredictiveModelList(models, categories, labels, ...)
```

Arguments

`models`	list containing predictive models
`categories`	numeric vector of land use categories in observed maps
`labels`	character vector with labels corresponding to `categories`
`...`	additional arguments (none)

Value

A PredictiveModelList object

Examples

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

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,ExpVarRasterStack,Raster-method

Resample maps in ExpVarRasterStack object or list

Description

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

Usage

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

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

Arguments

`x`	an ExpVarRasterStack 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 ExpVarRasterStack object or list, depending on `x`.

See Also

`ExpVarRasterStack`, `raster::resample`

Examples

```
## see lulcc-examples
```

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, n, digits = 0, ...)
```

Arguments

`x`	matrix or data.frame
`n`	numeric specifying the target sum for each row in `x`
`digits`	integer indicating the number of decimal places to be used
`...`	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

```
## Not run:

## Sibuyan Island

## load observed land use data and create demand scenario
obs <- LulcRasterStack(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
dmd <- roundSum(dmd, ncell=ncell)
```

```
apply(dmd, 1, sum)

## End(Not run)
```

---

show,PredictiveModelList-method *Show objects*

Description

Show lulcc objects

Usage

```
## S4 method for signature PredictiveModelList
show(object)

## S4 method for signature PredictionList
show(object)

## S4 method for signature PerformanceList
show(object)

## S4 method for signature Model
show(object)

## S4 method for signature ThreeMapComparison
show(object)
```

Arguments

`object`	an object belonging to one of the classes in `lulcc`

Value

Null

---

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.

Examples

```
data(sibuyan)
```

`subset,PredictiveModelList-method`
Subset objects

Description

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

Usage

```
## S4 method for signature PredictiveModelList
subset(x, subset, drop = FALSE, ...)

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

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

Arguments

x	an object of class `ExpVarRasterStack`, `PredictiveModelList`, `PredictionList` or `PerformanceList`
subset	integer or character indicating the objects to be extracted
drop	logical
...	additional arguments (none)

Value

Subsetted object, possibly simplified

Examples

```
## Sibuyan Island

lu <- DiscreteLulcRasterStack(x=stack(sibuyan$maps[1:2]),
                              categories=c(1,2,3,4,5),
                              labels=c("forest","coconut","grass","rice","other"),
                              t=c(0,14))


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

## load explanatory variables
idx <- data.frame(var=paste("ef_", formatC(1:13, width=3, flag=0)),
                  yr=rep(0,13),
                  dynamic=rep(FALSE,13))

ef <- ExpVarRasterStack(x=stack(sibuyan$maps[3:15]), index=idx)

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

summary — *Summary*

Description

Summarise lulcc objects

Usage

```
summary(object, ...)

## S4 method for signature LulcRasterStack
summary(object, ...)

## S4 method for signature ExpVarRasterStack
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

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

## 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
`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 lulcc-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

`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

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 Raster* object*

Description

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

Usage

```
total(x, ...)

## S4 method for signature DiscreteLulcRasterStack
total(x, categories, ...)

## S4 method for signature ContinuousLulcRasterStack
total(x, categories, ...)

## S4 method for signature Raster
total(x, categories, ...)
```

Arguments

`x`	Raster* object
`categories`	numeric vector containing land use categories. Only cells belonging to these categories will be counted
`...`	additional arguments (none)

Details

If x is a DiscreteLulcRasterStack object this function returns the number of cells belonging to each category. If x is a ContinuousLulcRasterStack object the function returns the sum of the fractions of the various land use categories.

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

```
## Sibuyan Island

lu <- DiscreteLulcRasterStack(x=stack(sibuyan$maps[1:2]),
                              categories=c(1,2,3,4,5),
                              labels=c("forest","coconut","grass","rice","other"),
                              t=c(0,14))

total(x=lu)
total(x=lu[[1]])
total(x=lu[[2]])
```

`updateDataFrame` *Update data frame*

Description

Function to update a data frame holding model variables with values of dynamic covariables for a new time point. This function is used internally by allocation routines.

Usage

```
updateDataFrame(x, ...)

## S4 method for signature ExpVarRasterStack
updateDataFrame(x, y, cells, time, ...)
```

Arguments

`x`	ExpVarRasterStack object
`y`	data.frame to update
`cells`	index of cells to be extracted and added to the data frame (see `as.data.frame`)
`time`	numeric indicating the time for which the data frame should be updated
`...`	additional arguments (none)

Value

data.frame

Index