Package: luscale (via r-universe)

March 3, 2025

Type Package

Date 2024-11-03

Title PIK Landuse Group Data Scaling Tools

Version 3.1.3

Description A collection of tools which allow to aggregate and
disaggregate data in various ways.

License BSD_2_clause + file LICENSE

URL https://github.com/pik-piam/luscale,
https://doi.org/10.5281/zenodo.1158584

BugReports https://github.com/pik-piam/luscale/issues

Depends magclass, methods, R (>= 2.10.0)

Imports madrat, spam, utils

Suggests covr

Encoding UTF-8

LazyData no

RoxygenNote 7.3.2

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev

Repository https://pik-piam.r-universe.dev

RemoteUrl https://github.com/pik-piam/luscale

RemoteRef HEAD

RemoteSha 0bc5b075c85f0a6e94163e58de31906ad3314c46

Contents

`luscale-package` *luscale: PIK Landuse Group Data Scaling Tools*

Description

A collection of tools which allow to aggregate and disaggregate data in various ways.

Author(s)

Maintainer: Jan Philipp Dietrich `<dietrich@pik-potsdam.de>`

Authors:

- Benjamin Leon Bodirsky `<bodirsky@pik-potsdam.de>`
- Markus Bonsch
- Patrick, von Jeetze
- Ulrich Kreidenweiss
- Roman Julius Hennig
- Florian Humpenoeder `<humpenoeder@pik-potsdam.de>`

See Also

Useful links:

- `https://github.com/pik-piam/luscale`
- doi:10.5281/zenodo.1158584
- Report bugs at `https://github.com/pik-piam/luscale/issues`

`AutomaticMapping` *AutomaticMapping*

Description

Function automatically finds fitting mapping for provided MAgPIE object for a given target aggregation.

Usage

```
AutomaticMapping(x,mapping=NULL,from=NULL,to=NULL)
```

Arguments

`x`	MAgPIE object
`mapping`	A array or data.frame containing a mapping query or mapping name
`from`	Only required if query is not NULL. Column of the query with original dimnames of the incoming dataset
`to`	Only required if query is not NULL. Column of the query with the target dimnames of the outcoming dataset

Author(s)

Benjamin Leon Bodirsky

`getAggregationMatrix` *getAggregationMatrix*

Description

Function which converts the supplied regionmapping file to a transformation matrix which then can be used for aggregation with `speed_aggregate`.

Usage

```
getAggregationMatrix(rel, from=NULL, to=NULL)
```

Arguments

`rel`	file name of a region mapping (".csv" as file ending, ";" used as separator and with columns containing the region names) or a mapping matrix (matrix with region names in columns)
`from`	Name of the first column to be used (if not set the first or second column will be used)
`to`	Name of the second column to be used (if not set the second or third column will be used)

Value

A matrix nregions1 x nregions2 with 1s and 0s showing the mapping of countries to regions.

Author(s)

Jan Philipp Dietrich

Examples

```
## Not run:
  x <- cbind(reg=c("REG1","REG2","REG2","REG3"),country=c("C1","C2","C3","C4"))

  getAggregationMatrix(x)

  getAggregationMatrix(x,from="reg",to="reg")

## End(Not run)
```

`groupAggregate` *groupAggregate*

Description

Function which applies aggregation functions over a subset of an array or magpie object.

Usage

```
groupAggregate(data, vectorfunction=function(x){sum(x,na.rm=TRUE)},
dim=3, query=NULL, from=NULL, to=NULL, ...)
```

Arguments

`data`	A MAgPIE object or array
`vectorfunction`	Aggregation Type. Can be any vector function, and will be applied over the sub-vectors of "dim". E.g. the function is applied to the vector of temperate cereals in each region and each timestep.
`dim`	The dimension over which is aggregated. Can be 1,3 or higher. This dimension will be replaced by the larger categories specified in the query.
`query`	Query assigns the name dimension into categories. If Null, query is automatically searched for. If no query can be found, you have to manually select a query (csv or array), as well as specifiy "from" and "to"
`from`	Only needed if query is not null. From indicates the column within the query where the dimnames of data can be found.
`to`	Only needed if query is not null. To indicates the column within the query into which the dimnames shall be grouped.
`...`	additional arguments for the vectorfunction.

Value

Returns a magpie object. The name dimension "dim" is grouped according to the query, and the vectorfunction is applied on this groups to aggregate them to each on column.

Author(s)

Benjamin Bodirsky

See Also

colSums, superAggregate

Examples

```
## Not run:
a <- new.magpie(cells_and_regions = c("ZAF.1","ZAR.2","VEN.3"),
                years = c("y1995","y2005"),
                names = c("Wheat","Barley","Sugar_cane","Sugar_beet"),
                fill=1:30)

#> a
#An object of class "magpie"
#, , Wheat
#
#      y1995 y2005
#ZAF.1     1     4
#ZAR.2     2     5
#VEN.3     3     6
#
#, , Barley
#
#      y1995 y2005
#ZAF.1     7    10
#ZAR.2     8    11
#VEN.3     9    12
#
#, , Sugar_cane
#
#      y1995 y2005
#ZAF.1    13    16
#ZAR.2    14    17
#VEN.3    15    18
#
#, , Sugar_beet
#
#      y1995 y2005
#ZAF.1    19    22
#ZAR.2    20    23
#VEN.3    21    24

b<-groupAggregate(a,dim=3)
#> b
```

```
#An object of class "magpie"
#, , tece
#
#      y1995 y2005
#ZAF.1     8    14
#ZAR.2    10    16
#VEN.3    12    18
#
#, , sugr_cane
#
#      y1995 y2005
#ZAF.1    13    16
#ZAR.2    14    17
#VEN.3    15    18
#
#, , sugr_beet
#
#      y1995 y2005
#ZAF.1    19    22
#ZAR.2    20    23
#VEN.3    21    24

c<-groupAggregate(a,dim=1)
#> c
#An object of class "magpie"
#, , Wheat
#
#      y1995 y2005
#AFR.1     3     9
#LAM.2     3     6
#
#, , Barley
#
#      y1995 y2005
#AFR.1    15    21
#LAM.2     9    12
#
#, , Sugar_cane
#
#      y1995 y2005
#AFR.1    27    33
#LAM.2    15    18
#
#, , Sugar_beet
#
#      y1995 y2005
#AFR.1    39    45
#LAM.2    21    24

## End(Not run)
```

`interpolate2` *Interpolate 2*

Description

Disaggregates a cellular MAgPIE output to 0.5 degree based on the given mapping and information about the initial 0.5 degree distribution.

Usage

```
interpolate2(x, x_ini, map, x_ini_lr = NULL)
```

Arguments

`x`	The object to be disaggregated. See details for further important information.
`x_ini`	The initial distribution of x in high resolution.
`map`	A relation map between low and high resolution
`x_ini_lr`	Low resolution version of x_ini. Will be calculated automatically, if not provided. Can speed up computation, if provided.

Details

There was a now deleted function called interpolate in this package before, hence the name interpolate2.

The function is based on the following assumption: `x` is an object in low resolution with more than one data dimension and the sum over the data dimensions is constant over time. Example: One column cropland, the other one (cell size - cropland). `x_ini` provides the same type of data as `x` but in high resolution and for the time step previous the initial time step of `x` (e.g. if `x` goes from t=a to t=a+10, `x_ini` must be provided for t=a-1). The function calculates the amount by which the individual data columns of `x` change in each timestep. The output is based on `x_ini` and only the differences in later timesteps to ths starting point are disaggregated by the given mapping. This assures that as little information as possible is lost from the original dataset `x_ini`.

The disaggregation procedure itself works as follows:

1. Differences in distribution between years are derived for the low resolution data set.

2. Based on these differences extension and reduction shares are calculated for the different pools. Reduction shares are calculated relative to the pool itself (e.g. a reduction in a cropland pool from 10ha to 6ha leads to a reduction share of (10ha-6ha)/10ha = 40%). At the same time extension shares are calculated relative to the pool which was made available by reductions of the other pools (e.g. cropland is reduced from 10ha to 6ha, forest area is reduced from 2ha to 1ha, but pastureland increases from 20ha to 25ha. In this case the extension share of pasture will be (25ha-20ha)/(10ha-6ha+2ha-1ha)=5ha/5ha=100%). This difference in calculation of reduction and extension share is crucial for the application at the high resolution level because otherwise the calculation will not add up.

3. Reduction and extension shares are disaggregated to the high resolution level by just assigning the same low resolution shares to all belonging cells at the higher resolution.

4.Starting with the provided high resolution pool data set for the initial year reduction shares are applied on all pools in all cells. The pool which is made available for expansions is calculated by summing up all values which were released by the pool reductions.

5. Pool expansions are calculated based on the pool made available in 4 for the first time step.

6. Steps 4 and 5 are repeated for all the following years based on the newly created high resolution data.

Applying this procedure makes sure that relative pool reductions are identical for the low resolution cell and all belonging high resolutions cells whereas the extension shares relative to the areas made available per cell are identical between low resolution cell and belonging high resolution cells.

Value

The disaggregated MAgPIE object containing x_ini as first timestep

Author(s)

Jan Philipp Dietrich

See Also

`toolAggregate`

`interpolateAvlCroplandWeighted` *interpolateAvlCroplandWeighted*

Description

Disaggregates a modelled time series of land pools after optimisation from the model resolution (low resolution) to the resolution of the land initialisation data set (high resolution), based on a relation map and available cropland.

Usage

```
interpolateAvlCroplandWeighted(
  x,
  x_ini_lr,
  x_ini_hr,
  avl_cropland_hr,
  map,
  urban_land_hr = "static",
  marginal_land = "all_marginal",
  land_consv_hr = NULL,
  peat_hr = NULL,
  snv_pol_shr = 0,
  snv_pol_fader = NULL,
  year_ini = "y1985",
  unit = "Mha"
)
```

Arguments

`x`	Time series of land pools (model output) to be disaggregated.
`x_ini_lr`	The low resolution distribution of x before model optimization.
`x_ini_hr`	The initial high resolution distribution of x (land-use initialisation) before model optimization.
`avl_cropland_hr`	The area of available cropland at the high resolution.
`map`	A relation map between low and high resolution
`urban_land_hr`	Either a magpie object of the cellular urban input data, or "static" string
`marginal_land`	Depending on the cropland suitability data, standard options are • `"all_marginal"`: Cropland can be allocated to marginal land • `"q33_marginal"`: The bottom tertile of the marginal land area is excluded • `"no_marginal"`: Marginal land is fully excluded from cropland
`land_consv_hr`	magpie object containing conservation land, e.g. `cell.conservation_land_0.5.mz` in the output folder
`peat_hr`	Disaggregated peatland with MAgPIE peatland pools
`snv_pol_shr`	Share of available cropland that is witheld for other land cover types. Can be supplied as a single value or as a magpie object containing different values in each iso country.
`snv_pol_fader`	Fader for share of set aside policy.
`year_ini`	Timestep that is assumed for the initial distributions `x_ini_hr` and `x_ini_lr`.
`unit`	Unit of the output. "Mha", "ha" or "share"

Details

The function requires the following input data:

- `x` is an object containing a time series of land pools (model output). The sum over all land pools is constant over time.
- `x_ini_lr` and `x_ini_hr` provide the initial land pools (Mha) at high (hr) and low resolution (lr) before the optimisation. They only contain the initial time step, but share the three-dimensional structure with `x`.
- `avl_cropland_hr` provides information about the amount (Mha) of available cropland at high resolution.
- `map` relation map containing information about cell belongings between the high and low resolution.

The weighted disaggregation works as follows:

1. The share of cropland in terms of total available cropland is calculated at the previous time step and then multiplied by the available cropland at the current time step (as available cropland can

change over time - e.g. by policy restriction as can be specified in `snv_pol_shr`). This temporary cropland pool is then compared to the low resolution cropland pool and the residual area of cropland expansion and reduction is determined.

2. In order to allocate residual area of cropland expansion and reduction, for each grid cell at high resolution expansion and reduction weights are calculated and multiplied by the residual area:

- The reduction weight is given by the ratio between the amount of cropland per grid cell and the total area of the temporary cropland at the low resolution spatial unit. This assumes that the cropland reduction is equally distributed among all high resolution grid cells.
- The expansion weight is calculated as the ratio between the remaining cropland at the grid cell level (high resolution) and the overall remaining cropland at the low resolution spatial unit in the current time step. The remaining cropland given by the difference between the available cropland and the temporaryl cropland pool minus urban land, since it assumed that cropland cannot be allocated to urban land.

3. Following the cropland allocation, the land area for the remaining non-cropland vegetation pools is calculated by substracting the allocated cropland and urban land areas from the total land area in each grid cell.

4. The non-cropland vegetation pool at the high resolution (except of primary forest), calculated in step 3., is then multiplied by the respective shares of the remaining non-cropland vegetation pools at the previous time step (temporary allocation). Similar to the cropland allocation, is not sufficient to also account for changes within these land pools. Therefore, the temporarily allocated non-cropland pools are, once again, compared with the pools at low resolution. The residual area of land expansion and reduction is then allocated by based on reduction and expansion weights, similar as in 2.. The reduction weight is calculated as the ratio between the given temporary land pool at high resolution and total temporary land pool at low resolution. The expansion weight is calculated as the ratio between the remaining land to be filled in each land pool and the total amount of residual land to be allocated in the current time step.

5. Primary forest is treated in a slightly different way, as primary forest cannot be expanded over time. In cropland cells with no cropland expansion, primary forest is, at first, assumed to remain constant and transferred from the previous time step to the current time step. Once again, the sum of the temporarary allocation is compared to the sum of primary forest at low resolution to determine the residual primary forest land, which still needs to be allocated. Where there is an surplus of primary forest, the reduction weight is calculated similarly as in 5., the land area is reduced accordingly. In areas where the temprorily allocated primary forest falls short, the allocation weight is calculated as a function of the difference in primary land between the previous time step and in the current time step. This makes sure that there is no expansion of primary forest.

7. Urban land is assumed to be constant over time.

Value

The disaggregated MAgPIE object containing x_ini_hr as first timestep

Author(s)

Patrick von Jeetze, David Chen

See Also

interpolate2 toolAggregate

Examples

```
## Not run:
a <- interpolateAvlCroplandWeighted(
  x = land,
  x_ini_lr = land_ini_lr,
  x_ini_hr = land_ini_hr,
  avl_cropland_hr = "avl_cropland_0.5.mz",
  map = "clustermap_rev4.59_c200_h12.rds",
  marginal_land = "all_marginal"
)

sf <- read.magpie("f30_scenario_fader.csv")[, , "by2030"]

b <- interpolateAvlCroplandWeighted(
  x = land,
  x_ini_lr = land_ini_lr,
  x_ini_hr = land_ini_hr,
  avl_cropland_hr = "avl_cropland_0.5.mz",
  map = "clustermap_rev4.59_c200_h12.rds",
  marginal_land = "all_marginal",
  snv_pol_shr = 0.2,
  snv_pol_fader = sf
)

iso <- readGDX(gdx, "iso")
set_aside_iso <- readGDX(gdx, "policy_countries30")
set_aside_select <- readGDX(gdx, "s30_snv_shr")
set_aside_noselect <- readGDX(gdx, "s30_snv_shr_noselect")
snv_pol_shr <- new.magpie(iso, fill = snv_noselect)
snv_pol_shr[set_aside_iso, , ] <- set_aside_select

c <- interpolateAvlCroplandWeighted(
  x = land,
  x_ini_lr = land_ini_lr,
  x_ini_hr = land_ini_hr,
  avl_cropland_hr = "avl_cropland_0.5.mz",
  map = "clustermap_rev4.59_c200_h12.rds",
  marginal_land = "all_marginal",
  snv_pol_shr = snv_pol_shr,
  snv_pol_fader = saf
)

## End(Not run)
```

`luqueries` *luqueries*

Description

Land use queries

Format

This datafile contains a list of queries (arrays with at least two columns, where in one column stands a word that is translated in the second column).

Details

Standardqueries: spatial: iso_reg: ISO_3166_1_alpha_3 country code to MAgPIE regions code fao_iso: faostat country name to ISO_3166_1_alpha_3 country code fbs_iso: similar to fao_iso, but with slightly different names used in the food balance sheets goods: faostat_kcr: faostat product names of crops to MAgPIE crop product categories faostat_kli: faostat product names of crops to MAgPIE livestock product categories

query is made with the script "luquery.R" in the SVN folder tools/queries. Please use this script to extend the query. Please do not use NAs in queries, as some scripts might have problems handling it.

Author(s)

Benjamin Leon Bodirsky

See Also

`rename_dimnames`,`groupAggregate`

`rename_dimnames` *rename_dimnames*

Description

Renames the dimnames of an array or MAgPIE object after a query.

Usage

```
rename_dimnames(data,dim=1,query=NULL,from=NULL,to=NULL)
```

Arguments

`data`	Array
`dim`	The dimension to be renamed.
`query`	If NULL, query is automatically searched for. Otherwhise an array, data.frame or the path of a csv with at least two columns. One column has to have the name of "from", the other one the name of "to". Some queries can be found in the svn-folder tools/queries.
`from`	Only required if query is not NULL. Column of the query with original dimnames of the incoming dataset
`to`	Only required if query is not NULL. Column of the query with the target dimnames of the outcoming dataset

Value

An array with different dimnames

Note

translate.with.query has the same functionality, is more efficient, yet more complicated to use.

Author(s)

Benjamin Bodirsky, Ulrich Kreidenweis

`speed_aggregate` *Speed Aggregate*

Description

Aggregates a MAgPIE object from one resolution to another based on a relation matrix

Usage

```
speed_aggregate(x,rel,fname=NULL,weight=NULL,from=NULL,to=NULL,
dim=1, partrel=FALSE)
```

Arguments

`x`	MAgPIE object or file name of a MAgPIE object
`rel`	relation matrix, file name of a spam relation matrix (".spam" as file ending), or file name of a region mapping (".csv" as file ending, ";" used as separator and with 3 columns: Country Name, Country code and region code)
`fname`	file name of a file the output should be written to. If fname=NULL the aggregated data is returned by the function

`weight`	MAgPIE object containing weights which should be considered for a weighted aggregation. Please see the "details" section below for more information
`from`	Name of the first column to be used in rel if it is a region mapping (if not set the first or second column will be used). This setting is ignored for spam files.
`to`	Name of the second column to be used in rel if it is a region mapping (if not set the second or third column will be used). This setting is ignored for spam files.
`dim`	Specifiying the dimension of the MAgPIE object that should be (dis-)aggregated. Either specified as an integer (1=spatial,2=temporal,3=data) or if you want to specify a subdimension specified by name of that dimension or position within the given dimension (e.g. 3.2 means the 2nd data dimension, 3.8 means the 8th data dimension).
`partrel`	If set to TRUE allowes that the relation matrix does contain less entries than x and vice versa. These values without relation are lost in the output.

Details

Basically speed_aggregate is doing nothing more than a normal matrix multiplication which is taking into account the 3 dimensional structure of MAgPIE objects. So, you can provide any kind of relation matrix you would like. However, for easier usability it is also possible to provide weights for a weighted (dis-)aggregation as a MAgPIE object. In this case rel must be a 1-0-matrix containing the mapping between both resolutions. The weight needs to be provided in the higher spatial aggregation, meaning for aggregation the spatial resolution of your input data and in the case of dissaggregation the spatial resolution of your output data. The temporal and data dimension must be either identical to the resolution of the data set that should be (dis-)aggregated or 1. If the temporal and/or data dimension is 1 this means that the same transformation matrix is applied for all years and/or all data columns. In the case that a column should be just summed up instead of being calculated as a weighted average you either do not provide any weight (than all columns are just summed up) or your set this specific weighting column to NA.

Value

If fname=NULL, the aggregated data, otherwise nothing is returned.

Author(s)

Jan Philipp Dietrich, Ulrich Kreidenweis

Examples

```
## Not run: low_res <- speed_aggregate(high_res,rel)
```

`superAggregate` *superAggregate*

Description

Function which applies aggregation functions over a subset of an magpie or lpj object.

Usage

```
superAggregate(data, aggr_type, level="reg", weight=NULL, na.rm=TRUE,
crop_aggr=FALSE, ...)
```

Arguments

`data`	An MAgPIE or LPJ object
`aggr_type`	Aggregation Type. Can be any function for one or two objects (data and weight) of the same size. Currently pre-supported functions: "sum","mean","weighted_mean".
`level`	Aggregation level: Either a level type (as a name) or a vector of 3-Character region names with the same length as the data has cells. Allowed level types are global "glo", regional "reg", per Country "country" and per REMIND regions "remind_reg". "country" and "remind_reg" are only supported for 0.5 grid datacells with 59199 cells and are always returned as arrays. If you use a vector of regions the aggregation will take place according to your regions.
`weight`	Currently only used for weighted_mean (see `weighted.mean`, yet also applicable for individualized functions. Has to be of the same size as data.
`na.rm`	If TRUE, NAs are ignored both in data and weight.
`crop_aggr`	determines whether output should be crop-specific (FALSE) or aggregated over all crops (TRUE). The method used for aggregation is set by aggr_type (Currently works only for levels "reg" and "glo")
`...`	additional arguments for the aggregation method for the standard functions (not for self-created ones)

Value

In the case of level="glo" or "reg", the function returns a MAgPIE object. In the case of level="country" an array is returned. In the case of an LPJ object being aggregated, a list of MAgPIE objects is returned, each entry being one of the 4th dimension slices.

Author(s)

Benjamin Bodirsky, Jan Philipp Dietrich, Florian Humpenoeder

See Also

`colSums`

Examples

```
data(population_magpie)
superAggregate(population_magpie,"sum",level="glo")
superAggregate(population_magpie,"mean",level="glo")
superAggregate(population_magpie,"weighted_mean",level="glo",weight=population_magpie)
aggregation_function<-function(func_data,func_weight) {
  colMeans(func_data)
}
superAggregate(population_magpie,aggregation_function,level="glo",weight=population_magpie)
```

Index