

Package: rmndt (via r-universe)

September 15, 2024

Title Tools for data.table objects in the REMIND context

Version 0.6.0

Description Helper functions for REMIND-related tasks with data.table objects, e.g., interpolation and (dis-)aggregation.

Depends R (>= 3.1), data.table (>= 1.11.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat, magclass, covr

Date 2024-04-18

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

RemoteUrl <https://github.com/pik-piam/rmndt>

RemoteRef HEAD

RemoteSha 7adc50e4c27032db8f55a2384e2b86cb09782faa

Contents

rmndt-package	2
aggregate_dt	2
apply_weights	3
approx_dt	4
disaggregate_dt	5
magpie2dt	6
readMIF	7
REMIND_FinalEnergy	7
REMIND_GDP	8
REMIND_RegionMap	8
varcalc_dt	9
writeMIF	9

Index	11
--------------	-----------

 rmndt-package

rmndt: Tools for data.table objects in the REMIND context

Description

Helper functions for REMIND-related tasks with data.table objects, e.g., interpolation and (dis-)aggregation.

Author(s)

Maintainer: Alois Dirnaichner <dirnaichner@pik-potsdam.de>

 aggregate_dt

Aggregate values in a data.table object using a mapping. If no weight is given, the value for the aggregated categories is the sum of the parts. Otherwise, the weight is used to calculate a weighted average accross the parts.

Description

Aggregate values in a data.table object using a mapping. If no weight is given, the value for the aggregated categories is the sum of the parts. Otherwise, the weight is used to calculate a weighted average accross the parts.

Usage

```
aggregate_dt(
  data,
  mapping,
  fewcol = "region",
  yearcol = "year",
  manycol = "iso",
  datacols = "data",
  valuecol = "value",
  weights = NULL,
  weightcol = "weight"
)
```

Arguments

data	a magpie object.
mapping	a mapping between the aggregated categories in the data and ISO3 countrycodes. *All* regions in 'data' have to be part of the mapping.
fewcol	name of the column containing aggregated categories. Default is "region".

yearcol	name of the column containing time step info. Default is "year".
manycol	name of the column containing dis-aggregated categories. Default is "iso".
datacols	index columns that label categories which have to be treated separately when aggregating with a weight.
valuecol	name of the column with the value to aggregate, default is 'value'.
weights	table with weights for a (weighted average) aggregation, the name of the column with the aggregated categories has to be 'manycol'. If columns (other than the column with the aggregated category) of the 'weights' coincide with columns of the data, the respective columns are considered when joining.
weightcol	column with the weights for aggregation, default is 'weight'.

apply_weights	<i>Internal function to apply the weights and perform some checks.</i>
---------------	--

Description

Internal function to apply the weights and perform some checks.

Usage

```
apply_weights(
  data,
  mapping,
  weights,
  fewcol,
  manycol,
  valuecol,
  datacols,
  weightcol
)
```

Arguments

data	a data.table.
mapping	a mapping between the aggregated categories and their parts. *All* aggregated categories in 'data' have to be part of the mapping.
weights	table with weights for disaggregation, the name of the column with the aggregated categories has to be 'manycol'. If columns (other than the column with the aggregated category) of the 'weights' coincide with columns of the data, the respective columns are considered when joining.
fewcol	name of the column containing aggregated categories. Default is "region".
manycol	name of the column containing dis-aggregated categories. Default is "iso".
valuecol	name of the column with the actual value to disaggregate, default is 'value'.
datacols	index columns that label categories which have to be treated separately when dis-aggregating with a weight.
weightcol	column with the weights for the dis-aggregation, default is 'weight'.

approx_dt

*Approximate missing values in a data.table.***Description**

Similar to, but not quite like, 'stats::approx'. Does only support constant extrapolation and linear interpolation. The resulting 'data.table' only contains the range provided by 'xdata' along 'xcol'. Without extrapolation, 'xcol' in the resulting 'data.table' may not cover the range given by 'xdata'.

Usage

```
approx_dt(
  dt,
  xdata,
  xcol,
  ycol,
  idxcols = NULL,
  keepna = FALSE,
  extrapolate = FALSE
)
```

Arguments

dt	a data.table.
xdata	the range to interpolate to. This is the range the result will have along the dimension 'xcol'.
xcol	name of the column for interpolation.
ycol	name of the column that contains the value to be interpolated.
idxcols	columns that identify a row (besides xcol), i.e., the remaining index dimensions.
keepna	keep NA values for rows that can not be interpolated (since they are outside of [min(xcol), max(xcol)]), default is FALSE.
extrapolate	use the closest values to fill 'ycol' outside of the interpolation domain, default is FALSE. This will also work if there is only one value along 'ycol', i.e., no interpolation is taking place.

Value

a data.table with the range given by 'xdata' along 'xcol'. Columns not given in 'idxcols' will be kept but NAs will appear on extrapolated and interpolated rows.

Examples

```
dt <- as.data.table(ChickWeight)
## delete all values but 1
dt[Chick == 1 & Time > 0, weight := NA]
## delete all values but 2
```

```

dt[Chick == 2 & Time > 2, weight := NA]

## extrapolation from 1 value
approx_dt(dt, 0:21, "Time", "weight", idxcols=c("Chick", "Diet"), extrapolate = TRUE)[Chick == 1]
## extrapolation and interpolation
approx_dt(dt, 0:21, "Time", "weight", idxcols=c("Chick", "Diet"), extrapolate = TRUE)[Chick == 2]
## column not in idxcols
approx_dt(dt, 0:21, "Time", "weight", idxcols="Chick", extrapolate = TRUE)[Chick == 2]

dt <- as.data.table(ChickWeight)
## interpolation only
approx_dt(dt, 0:21, "Time", "weight", idxcols=c("Chick", "Diet"))[Chick == 2]

```

disaggregate_dt	<i>Disaggregate data in a data.table object using a mapping. If no weights are given, the value for the aggregated categories is used on the disaggregated ones. If a weight is given, the values from the aggregated categories are distributed according to the weights.</i>
-----------------	--

Description

Disaggregate data in a data.table object using a mapping. If no weights are given, the value for the aggregated categories is used on the disaggregated ones. If a weight is given, the values from the aggregated categories are distributed according to the weights.

Usage

```

disaggregate_dt(
  data,
  mapping,
  fewcol = "region",
  manycol = "iso",
  valuecol = "value",
  datacols = "data",
  weights = NULL,
  weightcol = "weight"
)

```

Arguments

data	a data.table.
mapping	a mapping between the aggregated categories and their parts. <i>*All*</i> aggregated categories in ‘data’ have to be part of the mapping.
fewcol	name of the column containing aggregated categories. Default is "region".
manycol	name of the column containing dis-aggregated categories. Default is "iso".
valuecol	name of the column with the actual value to disaggregate, default is ‘value’.

datacols	index columns that label categories which have to be treated separately when dis-aggregating with a weight.
weights	table with weights for disaggregation, the name of the column with the aggregated categories has to be 'manycol'. If columns (other than the column with the aggregated category) of the 'weights' coincide with columns of the data, the respective columns are considered when joining.
weightcol	column with the weights for the dis-aggregation, default is 'weight'.

magpie2dt	<i>Load a magpie object as data.table object with given colnames. Replaces years by numeric values, removing the leading y.</i>
-----------	---

Description

Load a magpie object as data.table object with given colnames. Replaces years by numeric values, removing the leading y.

Usage

```
magpie2dt(
  data,
  regioncol = NULL,
  yearcol = NULL,
  datacols = NULL,
  valcol = "value"
)
```

Arguments

data	a magpie object.
regioncol	name of the column containing REMIND regions, default is "region".
yearcol	name of the column containing the year, default is "year".
datacols	the names of the data dimension(s) of the magpie object. If no value is given, the name provided in the magpie object is used.
valcol	column to host actual value, default is "value"

Examples

```
## Not run:
require(magpie)
dt <- magpie2dt(population_magpie)

## End(Not run)
```

readMIF	<i>Read a REMIND output (MIF) file.</i>
---------	---

Description

REMIND style output files are semi-colon separated CSVs with a trailing semi-colon at the end of each row. The following structure is assumed: Columns "Model", "Scenario", "Region", "Variable", "Unit" and an arbitrary number of year columns (convertable to numeric).

Usage

```
readMIF(mif)
```

Arguments

mif	A REMIND output file (.MIF)
-----	-----------------------------

Examples

```
## Not run:
dt <- readMIF("REMIND_generic_default.mif")

## End(Not run)
```

REMIND_FinalEnergy	<i>A random REMIND FE trajectory.</i>
--------------------	---------------------------------------

Description

A random REMIND FE trajectory.

Usage

```
REMIND_FinalEnergy
```

Format

A data frame with 2011 rows and 6 columns:

year	REMIND time step
region	REMIND-12 region
se	Secondary energy identifier
fe	Final energy identifier
te	Conversion technology identifier
value	The data column, EJ/yr

 REMIND_GDP

REMIND GDP Trajectories for ISO countries

Description

REMIND GDP Trajectories for ISO countries

Usage

REMIND_GDP

Format

A data frame with 9462 rows and 4 columns:

iso ISO3 country code

variable SSP GDP variant, this is gdp_SSP2 in this case

weight GDP in US\$2010

year Year

 REMIND_RegionMap

The mapping between REMIND-12 regions and ISO countries

Description

The mapping between REMIND-12 regions and ISO countries

Usage

REMIND_RegionMap

Format

A data frame with 249 rows and 3 columns:

name Name of the country

iso ISO3 country code

region REMIND region name

varcalc_dt	<i>Execute *vertical* calculations along a given column.</i>
------------	--

Description

This assumes a **long** format with a single value column, dcasts the data.table to wide format, executes the calculation(s), melts back to long format and returns the resulting data.table with the additional column(s).

Usage

```
varcalc_dt(dt, varcol, valcol, expr, ...)
```

Arguments

dt	data.table, long format
varcol	name of the column with the variable
valcol	name of the column with the value
expr	vector of expressions to be handed to j in data.table, as strings, e.g., "a := b/c"
...	other arguments are passed on to the data.table call where 'expr' is evaluated. Most likely you want to pass the 'by=' parameter for group-by calls, see examples.

Details

Note that the data.table should have at least three columns, i.e., the variable, the value and one id column.

Examples

```
mt_dt <- as.data.table(mtcars, keep.rownames = TRUE)
## to long
mt1 <- melt(mt_dt, id.vars=c("rn", "cyl"))

varcalc_dt(mt1, "variable", "value", c("`spec. hp` := wt/hp", "wsum := sum(wt)"), by="cyl")
```

writeMIF	<i>Write a REMIND output (MIF) file.</i>
----------	--

Description

Note that these files are semi-colon separated CSVs with a trailing semi-colon at the end of each entry. Required columns are "Model", "Scenario", "Region", "Variable", "Unit" and an arbitrary number of year columns (should be convertible to numeric).

Usage

```
writeMIF(dt, destination, append = FALSE, ...)
```

Arguments

<code>dt</code>	a data.table in the correct format.
<code>destination</code>	path to the resulting MIF file
<code>append</code>	append to an existing MIF file?
<code>...</code>	other parameters are passed on to data.table::fwrite

Details

NAs are written to the file as "N/A".

Examples

```
## Not run:  
writeMIF(dt, "REMIND_generic_default.mif")  
  
## End(Not run)
```

Index

* datasets

REMIND_FinalEnergy, [7](#)

REMIND_GDP, [8](#)

REMIND_RegionMap, [8](#)

aggregate_dt, [2](#)

apply_weights, [3](#)

approx_dt, [4](#)

disaggregate_dt, [5](#)

magpie2dt, [6](#)

readMIF, [7](#)

REMIND_FinalEnergy, [7](#)

REMIND_GDP, [8](#)

REMIND_RegionMap, [8](#)

rmndt (rmndt-package), [2](#)

rmndt-package, [2](#)

varcalc_dt, [9](#)

writeMIF, [9](#)