Package: mrdrivers (via r-universe)

January 17, 2025

Type Package

Title Create GDP and Population Scenarios

Version 4.0.7

Description Create GDP and population scenarios This package constructs the GDP and population scenarios used as drivers in both the REMIND and MAgPIE models.

License LGPL (>= 3)

URL https://pik-piam.github.io/mrdrivers, https://github.com/pik-piam/mrdrivers

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

Depends madrat (>= 2.5.1), magclass (>= 6.0.3), GDPuc (>= 1.3.0)

Imports countrycode, dplyr, glue, magrittr, purrr, readr, readxl, rlang, tibble, tidyr, tidyselect

Suggests covr, crayon, knitr, rmarkdown, testthat (>= 3.0.0), WDI, withr (>= 2.4.2), yaml, zoo

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Date 2024-11-18

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev libx11-dev

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

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

RemoteRef HEAD

RemoteSha 208285c98a6ef1cbb4f4634307960fda567782ad

Contents

`calcGDP` *Get GDP and GDP per capita scenarios*

Description

Like all scenarios in mrdrivers, the GDP and GDP per capita scenarios are the result of a harmonization exercise between past data and future projections. Together with the corresponding population scenarios (see `calcPopulation()`) they comprise a consistent set of scenarios.

By default the following scenarios are returned:

- the SSPs, i.e. SSP1-5
- the SDPs, i.e. SDP, SDP_EI, SDP_RC, and SDP_MC

See the vignette: `vignette("scenarios")` for scenario options, definitions and references.

Usage

```
calcGDP(
  scenario = c("SSPs", "SDPs", "SSP2EU"),
  unit = toolGetUnitDollar(inPPP = TRUE),
  average2020 = TRUE,
  ...
)

calcGDPpc(
  scenario = c("SSPs", "SDPs", "SSP2EU"),
  unit = toolGetUnitDollar(inPPP = TRUE),
```

```
  average2020 = TRUE,
  ...
)
```

Arguments

scenario	A string (or vector of strings) designating the scenario(s) to be returned. Use toolGetScenarioDefinition() to learn what scenarios are available.
unit	A string specifying the unit of GDP. Can be either: • "constant 2017 Int$PPP" (default): Scenarios are constructed in constant 2017 Int$PPP. • "constant 2017 US$MER": Scenarios are constructed in constant 2017 US$MER and then converted with GDPuc::toolConvertGDP(). In all cases, GDP is returned in millions.
average2020	If TRUE (default), then the 2020 value is replaced by the 2018-2022 average. To be consistent, the yearly resolution is decreased to 5 year intervals.
...	Arguments passed on to calcDriver(), of which "extension2150" and "naming" are most often of interest.

Value

magpie object with the requested output data either on country or on regional level depending on the choice of argument "aggregate" or a list of information if supplementary is set to TRUE.

See Also

- toolGetScenarioDefinition() for scenario options and definitions.
- madrat::calcOutput() for how to return supplementary information and other control options.
- calcDriver(), calcScenarioConstructor() and calcHarmonizedData() for how to create new scenarios (for developers).

Examples

```
## Not run:
# Return default scenarios
calcOutput("GDP")
calcOutput("GDPpc")

# Return only the SSP2 GDP scenario
calcOutput("GDP", scenario = "SSP2")

## End(Not run)

## Not run:
calcOutput("GDPpc")

## End(Not run)
```

`calcPopulation` *Get population and labour scenarios*

Description

Like all scenarios in mrdrivers, the Population, Labour and Urban population share scenarios are the result of a harmonization exercise between past data and future projections.

By default the following scenarios are returned:

- the SSPs, i.e. SSP1-5
- the SDPs, i.e. SDP, SDP_EI, SDP_RC, and SDP_MC

See the vignette: `vignette("scenarios")` for scenario options, definitions and references.

Usage

```
calcPopulation(scenario = c("SSPs", "SDPs", "SSP2EU"), ...)

calcLabour(scenario = c("SSPs", "SDPs", "SSP2EU"), ...)

calcUrban(scenario = c("SSPs", "SDPs", "SSP2EU"), asShare = TRUE, ...)
```

Arguments

`scenario`	A string (or vector of strings) designating the scenario(s) to be returned. Use `toolGetScenarioDefinition()` to learn what scenarios are available.
`...`	Arguments passed on to `calcDriver()`, of which "extension2150" and "naming" are most often of interest. Other `calcDriver()` arguments are used for scenario fine-tuning and by package developers.
`asShare`	If TRUE (default) urban population shares are returned. If FALSE, then urban population in millions is returned.

Value

magpie object with the requested output data either on country or on regional level depending on the choice of argument "aggregate" or a list of information if supplementary is set to TRUE.

See Also

- `toolGetScenarioDefinition()` for scenario options and definitions.
- `madrat::calcOutput()` for how to return supplementary information and other control options.
- `calcDriver()`, `calcScenarioConstructor()` and `calcHarmonizedData()` for how to create new scenarios (for developers).

Examples

```
## Not run:
# Return the default scenarios
calcOutput("Population")

# Return only the SSP2 scenario
calcOutput("Population", scenario = "SSP2")

# Return the ISIMIP SSP scenarios
calcOutput("Population", scenario = "ISIMIP", extension2150 = "none", aggregate = FALSE)

## End(Not run)
## Not run:
calcOutput("Labour")

## End(Not run)
## Not run:
calcOutput("Urban")

## End(Not run)
```

`calcRatioPPP2MER` *MER over PPP ratio*

Description

Get a conversion factor to convert GDP in constant 2017 Int$PPP into constant 2017 US$MER. Use the "when" argument to switch the year of the conversion factor. Source = WDI. Countries with missing data are filled in with 1. Regional aggregation is weighed by GDP from WDI-MI in the year set by "when".

Usage

```
calcRatioPPP2MER(when = 2017)
```

Arguments

`when`	An integer (defaults to 2017) specifying the year of the PPP2MER factor.

Value

magpie object with the requested output data either on country or on regional level depending on the choice of argument "aggregate" or a list of information if supplementary is set to TRUE.

See Also

`madrat::calcOutput()`

Examples

```
## Not run:
calcOutput("RatioPPP2MER")

## End(Not run)
```

`downloadIMF` *Read IMF*

Description

Read-in data from the IMF's World Economic Outlook. Currently reading GDP per capita and current account balance data.

Usage

```
downloadIMF()

readIMF()

convertIMF(x, subtype = "all")
```

Arguments

`x`	MAgPIE object returned by readIMF
`subtype`	Use to filter the IMF data

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

`madrat::readSource()` and `madrat::downloadSource()`

Examples

```
## Not run:
readSource("IMF")

## End(Not run)
```

`downloadJames` *Read James et al. (2012) dataset*

Description

Read-in GDP per-capita data from the publication James, Spencer L., Paul Gubbins, Christopher JL Murray, and Emmanuela Gakidou. 2012. "Developing a Comprehensive Time Series of GDP per Capita for 210 Countries from 1950 to 2015." Population Health Metrics 10 (1): 12. doi:10.1186/1478-7954-10-12.

Usage

```
downloadJames()

readJames(subtype)

convertJames(x, subtype)
```

Arguments

`subtype`	String indicating the data series
`x`	MAgPIE object returned by readJames

Details

The data is in Annex 3

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

`madrat::readSource()` and `madrat::downloadSource()`

Examples

```
## Not run:
readSource("James", subtype = "gdp")

## End(Not run)
```

`downloadMissingIslands` *Read in the "Missing Islands" dataset*

Description

Read in gdp or population data for minor islands (not included in big inventories) from a custom made data set that gets data from a variety of sources (e.g. CIA World Factbook, Insee, BEA, PRISM, and Woldometers).

Usage

```
downloadMissingIslands()

readMissingIslands(subtype)

convertMissingIslands(x, subtype)
```

Arguments

subtype	pop for population, or gdp for gdp
x	MAgPIE object returned by readMissingIslands

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

`madrat::readSource()` and `madrat::downloadSource()`

Examples

```
## Not run:
readSource("MissingIslands", subtype = "pop")

## End(Not run)
```

downloadPEAP *Read Population Estimates And Projections from the World Bank*

Description

Read-in xlsx file from the World Bank's Population Estimates And Projections (PEAP) The PEAP data can't seemed to be accessed by the WDI::WDI package nor the World Bank's API directly. Manual download required from https://databank.worldbank.org/source/population-estimates-and-projections#

Usage

```
downloadPEAP()

readPEAP()

convertPEAP(x)
```

Arguments

x	MAgPIE object returned by readPEAP

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

`madrat::readSource()` and `madrat::downloadSource()`

downloadUN_PopDiv *Read UN Population Division Data*

Description

Read UN population data.

Usage

```
downloadUN_PopDiv()

readUN_PopDiv(subtype, subset = "estimates")

convertUN_PopDiv(x)
```

Arguments

subtype	Either "pop" or "lab".
subset	Either "estimates" or "medium".
x	MAgPIE object returned from readUN_PopDiv

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

madrat::readSource() and madrat::downloadSource()

---

downloadWDI *Read WDI data*

Description

Download, read and convert WDI (World development indicators) data.

Usage

```
downloadWDI()

readWDI(subtype)

convertWDI(x, subtype)
```

Arguments

subtype	A string. Type of WDI data that should be read. Use the World Bank indicator abbreviation. Available subtypes are: • "pop" or "SP.POP.TOTL": Population, total • "lab" or "SP.POP.1564.TO": Working age population (15-64 years old) • "urb" or "SP.URB.TOTL.IN.ZS": Urban Population (% of total) • "gdp" or "NY.GDP.MKTP.PP.KD": GDP, PPP (constant 2017 international Dollar) • "NV.AGR.TOTL.KD": Ag GDP, MER, (2010 US$) • "PA.NUS.PPPC.RF": Price level ratio of PPP conversion factor (GDP) to market exchange rate • "AG.SRF.TOTL.K2": Surface area (in square kms)
x	MAgPIE object returned by readWDI

Details

The workflow to update the WDI data is the following: call the download function manually, and rename the new WDI.rds file including the download date. Then change the file_name that is read by readWDI. This ensures, that the past data isn't changed between users.

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

`madrat::readSource()` and `madrat::downloadSource()`

Examples

```
## Not run:
readSource("WDI", subtype = "pop")

## End(Not run)
```

---

readADB *Read ADB data*

Description

Read-in an ADB data as magclass object

Usage

```
readADB()

convertADB(x, subtype = "all")

downloadADB()
```

Arguments

x	MAgPIE object returned from readADB
subtype	A string, either "all", "gdppc", "pop"

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

madrat::readSource()

Examples

```
## Not run:
readSource("ADB", subtype = "gdppc")

## End(Not run)
```

---

readSSP *Read SSP*

Description

Read-in an SSP data as magclass object. Filter for subtype and subset in the convert Function to use common read cache (speeds up the compuations).

Usage

```
readSSP()

convertSSP(
  x,
  subtype = "all",
  subset = c("SSP1", "SSP2", "SSP3", "SSP4", "SSP5")
)

downloadSSP()
```

Arguments

x	MAgPIE object returned from readSSP
subtype	A string, either "all", "gdp", "pop", "lab", "urb"
subset	A vector of strings designating the scenarios. Defaults to c("SSP1", "SSP2", "SSP3", "SSP4", "SSP5"). "Historical Reference" is also available as a scenario.

Value

The read-in data, usually a magpie object. If supplementary is TRUE a list including the data and metadata is returned instead. The temporal and data dimensionality should match the source data. The spatial dimension should either match the source data or, if the convert argument is set to TRUE, should be on ISO code country level.

See Also

madrat::readSource()

Examples

```
## Not run:
readSource("SSP", subtype = "gdp")

## End(Not run)
```

`toolGeneralConvert` *Tool used to consolidate the most common "convert" operations*

Description

The most important and common "convert" operations are:

- removing undefined countries,
- substituting NAs, see the "substituteNAsWith" argument,
- using default set names "iso3c", "year", and "variable",
- fill in countries,
- sort in chronological order.

Usage

```
toolGeneralConvert(
  x,
  useDefaultSetNames = TRUE,
  countryFillWith = 0,
  substituteNAsWith = 0,
  warn = TRUE,
  note = TRUE,
  ...
)
```

Arguments

`x`	A magpie object.
`useDefaultSetNames`	TRUE or FALSE.
`countryFillWith`	1.
`substituteNAsWith`	1.
`warn`	TRUE or FALSE.
`note`	TRUE or FALSE.
`...`	Arguments passed on to `madrat::toolCountryFill()`

Value

A magpie object.

Examples

```
## Not run:
toolGeneralConvert(x)

## End(Not run)
```

`toolGetScenarioDefinition`

Get information on available scenarios

Description

toolGetScenarioDefinition can be used to figure out which scenarios are made available by mrdrivers, and how they are constructed, i.e. what past data, future data and harmonization methods are used.

Usage

```
toolGetScenarioDefinition(driver = NULL, scen = NULL, aslist = FALSE)
```

Arguments

`driver`	NULL or a character vector designating the driver for which information is to be returned. If NULL, information for all drivers is returned. Available drivers are: • GDP • Population • GDPpc • Labour • Urban
`scen`	NULL or a character vector designating the scenario for which information is to be returned. If NULL, information for all scenarios is returned.
`aslist`	TRUE or FALSE (default). If TRUE then the pastData, futureData and harmonization strings are returned as a list.

Value

A tibble with the driver and scenario information.

Examples

```
toolGetScenarioDefinition()
toolGetScenarioDefinition(driver = "GDP")
toolGetScenarioDefinition(scen = "SSP2")
toolGetScenarioDefinition(driver = "Population", scen = "SSPs", aslist = TRUE)
```

---

toolGetUnitDollar *Get monetary unit*

Description

toolGetUnitDollar returns unit used for monetary values.

Usage

```
toolGetUnitDollar(returnOnlyBase = FALSE, inPPP = FALSE)
```

Arguments

returnOnlyBase	TRUE or FALSE (default). If true only the base year is returned (as string).
inPPP	TRUE or FALSE (default). If TRUE the the string ends in 'Int$PPP', instead of 'Int$MER'.

Value

A string with the monetary unit: constant 2017 US$MER.

Examples

```
toolGetUnitDollar()
toolGetUnitDollar(inPPP = TRUE)
toolGetUnitDollar(returnOnlyBase = TRUE)
```

---

toolHarmonizeFuture *Harmonization tool Future*

Description

Like all harmonization tools in mrdrivers, toolHarmonizeFuture takes two magpie objects, 'past' and 'future', and returns a single magpie object, i.e. the harmonized time-series. In this case, the harmonized time-series is always equal to 'future', in the years of 'future'. After that the harmonized time-series depends on the 'method' argument chosen.

Usage

```
toolHarmonizeFuture(past, future, method = "level")
```

Arguments

past	A magpie object.
future	A magpie object with only one scenario/datatype, i.e. the length of the third dimension should be 1.
method	A string defining the harmonization method: • "level": the harmonized time-series is exactly equal to 'past' in the years before the first year of 'future'. • "growth": the harmonized time-series follows the same growth rates as 'past', in the years before the first year of 'future'.

Value

A magpie object with the same dimensions as 'past'.

Dimensions of 'past' and 'future'

If the 'past' object has multiple scenarios/datatypes, i.e. the length of the third dimension is larger than 1, then the a harmonized time-series is created for every scenario/datatype in past. The same 'future' object is used in every case - hence the requirement that 'future' only have one scenario/datatype.

Examples

```
## Not run:
toolHarmonizePast(past, future)

## End(Not run)
```

---

toolHarmonizePast *Harmonization tool Past*

Description

Like all harmonization tools in mrdrivers, toolHarmonizePast takes two magpie objects, 'past' and 'future', and returns a single magpie object, i.e. the harmonized time-series. In this case, the harmonized time-series is always equal to 'past', in the years of 'past'. After that the harmonized time-series depends on the 'method' argument chosen.

Usage

```
toolHarmonizePast(past, future, method = "level", yEnd = 2100)
```

Arguments

past	A magpie object with only one scenario/datatype, i.e. the length of the third dimension should be 1.
future	A magpie object.
method	A string defining the harmonization method: • "level": the harmonized time-series is exactly equal to 'future' in the years after the last year of 'past'. • "growth": the harmonized time-series follows the same growth rates as 'future', in the years after the last year of 'past'. • "transition": the harmonized time-series transitions to 'future' by the year 'yEnd'. After yEnd, the harmonized time-series is equal to 'future'. In the transition phase, a share of the absolute difference between past' and future' in the last year of past' is added to future. This share starts at 1 in the last year of 'past' and decreases linearly to 0 by yEnd.
yEnd	Additional input for "transition" method. Year by which the transition period is completed.

Value

A magpie object with the same dimensions as 'future'.

Dimensions of 'past' and 'future'

If the 'future' object has multiple scenarios/datatypes, i.e. the length of the third dimension is larger than 1, then the a harmonized time-series is created for every scenario/datatype in future. The same 'past' object is used in every case - hence the requirement that 'past' only have one scenario/datatype.

Examples

```
## Not run:
toolHarmonizePast(past, future)

## End(Not run)
```

Index