

Package ‘gtapssp’

January 23, 2025

Title GTAPSSP: Tools for Processing SSPs in the GTAP Framework

Version 0.0.0.9000

Description Provides tools for preprocessing, aggregating, interpolating, and expanding Shared Socioeconomic Pathways (SSPs) data for integration with the Global Trade Analysis Project (GTAP) modelling framework.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

LazyData true

Depends R (>= 3.1)

Imports stats,
data.table (>= 1.14.8),
dplyr,
tidyr,
HARr,
tools

Remotes github::USDA-ERS/MTED-HARr

Contents

aggData	2
calc_beers	3
csvtorda	4
growth_rate	4
iiasa_gtap	5
interpolate_beers	7
interpolate_spline	8
read_csv_from_zip	9
txtToData	10
updateCorresp	11
updateData	12
Index	14

aggData

*Aggregate IIASA Data Using Regional Mappings***Description**

This function aggregates IIASA data based on a specified regional correspondence table. Optionally, it can use additional regional mappings from a Gempack-style text file to refine the correspondence table. The aggregated data is grouped by specified columns and years.

Usage

```
aggData(
  iiasa_raw = gtapssp::iiasa_raw,
  corresp_reg = gtapssp::corresp_reg,
  aggTxtFile = NULL,
  group_cols = c("model", "scenario", "reg_gtap_code", "variable", "unit")
)
```

Arguments

<code>iiasa_raw</code>	A list containing IIASA raw data, typically the output of the <code>updateData</code> function. Default is <code>gtapssp::iiasa_raw</code> .
<code>corresp_reg</code>	A data frame containing the correspondence table between regions and their codes. Default is <code>gtapssp::corresp_reg</code> .
<code>aggTxtFile</code>	Character. The path to a Gempack-style text file containing additional regional mappings. If <code>NULL</code> , this step is skipped. Default is <code>NULL</code> .
<code>group_cols</code>	Character vector. The column names to use for grouping the aggregated data. Default is <code>c("model", "scenario", "reg_gtap_code", "variable", "unit")</code> .

Value

A data frame containing the aggregated IIASA data grouped by the specified columns and years. The aggregation sums up the value column for each group.

See Also

[updateData](#), [group_by](#), [drop_na](#)

Examples

```
## Not run:
# Example with an additional mapping file
agg_data <- aggData(
  iiasa_raw = gtapssp::iiasa_raw,
  corresp_reg = gtapssp::corresp_reg,
  aggTxtFile = "path/to/aggFile.agg"
)

# Example without an additional mapping file
agg_data <- aggData(
  iiasa_raw = gtapssp::iiasa_raw,
  corresp_reg = gtapssp::corresp_reg
```

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

calc_beers

Perform Beers Interpolation or Subdivision

Description

This function implements the Beers interpolation or subdivision methods, either in their ordinary or modified forms. It generates interpolated or subdivided points from a given data set. The Beers interpolation method, first introduced in "The Record of the American Institute of Actuaries" (Vol. 34, Part I, 1945), is a six-term formula designed to minimize the fifth differences in interpolated results. More details can be found in "The Methods and Materials of Demography, Volume 2" (page 877).

Usage

```
calc_beers(values, method = "ordinary")
```

Arguments

values	A numeric vector of data points for interpolation or subdivision.
method	A character string specifying the method to be used. (default = "ordinary") Must be one of "ordinary", "modified", "subdivision_ordinary", or "subdivision_modified". - "ordinary": In interpolation, include original data points unchanged; in subdivision, each set of 5 subdivided values sums to the original data point. - "modified": In interpolation, includes some smoothing with only the first and last points unchanged; in subdivision, similar smoothing occurs.

Value

A numeric vector with interpolated or subdivided values.

Examples

```
calc_beers(c(1, 2, 3, 4, 5), "ordinary")
calc_beers(c(1, 2, 3, 4, 5), "modified")
calc_beers(c(1, 2, 3, 4, 5), "subdivision_ordinary")
calc_beers(c(1, 2, 3, 4, 5), "subdivision_modified")
```

 csvtorda

Save CSV Files as RDA Objects

Description

This function reads all .csv files in a specified folder, converts each into a data frame, and saves them as .rda files in the same folder. The name of each saved object matches the name of its source .csv file (without the extension).

Usage

```
csvtorda(data_folder = "data/")
```

Arguments

data_folder A character string specifying the path to the folder containing .csv files. Defaults to "data/".

Value

NULL. The function performs file operations and saves .rda files but does not return any value.

Examples

```
## Not run:
  csvtorda("data/") # Save all .csv files in the "data/" folder as .rda

## End(Not run)
```

 growth_rate

Calculate Growth Rate by Year for Specified Groups

Description

This function calculates the growth rate of a specified value column grouped by specified columns in the data. The growth rate is computed as the percentage change between consecutive years within each group. Missing growth rates (NA) are replaced with a user-defined value.

Usage

```
growth_rate(
  data,
  group_cols = c("model", "scenario", "reg_gtap_code", "variable", "unit"),
  year_col = "year",
  value_col = "value",
  growth_rate_col = "growth_rate",
  na_replace = 0
)
```

Arguments

<code>data</code>	A data frame containing the input data for which growth rates are to be calculated.
<code>group_cols</code>	A character vector specifying the columns to group by before calculating the growth rate. Default is <code>c("model", "scenario", "reg_gtap_code", "variable", "unit")</code> .
<code>year_col</code>	A string specifying the column representing the year. Default is <code>"year"</code> .
<code>value_col</code>	A string specifying the column containing the values for which growth rates are calculated. Default is <code>"value"</code> .
<code>growth_rate_col</code>	A string specifying the name of the new column where the calculated growth rates will be stored. Default is <code>"growth_rate"</code> .
<code>na_replace</code>	A numeric value used to replace missing growth rates (NA). Default is 0.

Value

A data frame containing the original data with an additional column, representing the calculated growth rates as percentage changes.

See Also

[mutate](#), [group_by](#), [arrange](#)

Examples

```
## Not run:
# Example usage
growth_rates <- growth_rate(
  data = my_data,
  group_cols = c("model", "scenario", "reg_gtap_code", "variable", "unit"),
  year_col = "year",
  value_col = "value",
  growth_rate_col = "annual_growth_rate"
)

## End(Not run)
```

Description

This function executes the routine for processing and aggregating IIASA data for GTAP SSP integration. The routine includes data aggregation, interpolation (spline and beers methods), column expansion, filtering, and growth rate calculation. Optionally, the final dataset can be saved as a CSV file.

Usage

```
iiasa_gtap(
  outFile = NULL,
  group_cols = c("model", "scenario", "reg_gtap_code", "variable", "unit")
)
```

Arguments

<code>outFile</code>	Character. Optional path to save the final dataset as a CSV file. If NULL, the file is not saved. Default is NULL.
<code>group_cols</code>	Character vector. Columns to group by during aggregation. Default is <code>c("model", "scenario", "reg_gtap_code", "variable", "unit")</code> .

Details

Steps in the Routine:

- **Step 1: Data Aggregation:** Aggregates the IIASA raw data using the `aggData()` function, which allows for regional mappings based on `aggTxtFile`. The aggregation groups the data by columns specified in `group_cols` and combines regional data accordingly.
- **Step 2: Spline Interpolation:** Applies the `interpolate_spline()` function to interpolate missing values in the value column for the specified models ("IIASA GDP 2023", "OECD ENV-Growth 2023"). The cubic spline method creates a smooth curve that passes through the available data points, ensuring continuity in derivatives and producing realistic interpolations.
- **Step 3: Beers Interpolation:** Applies the `interpolate_beers()` function to interpolate missing population data (`model = "IIASA-WiC POP 2023"`) using Beers interpolation. This method is specifically designed for demographic data, providing age-structured estimates that align with population distribution and totals. It preserves consistency between age groups while interpolating.
- **Step 4: Combine Outputs:** Combines the results of the spline and beers interpolations. This step stacks the interpolated data vertically to produce a unified dataset.
- **Step 5: Expand Variables:** Splits the variable column into multiple columns (e.g., variable, gender, cohort, and education_level). This expansion helps isolate specific attributes encoded in the variable column.
- **Step 6: Filter Data:** Filters rows based on several conditions:
 - Removes total rows (`education_level` is NA) unless the scenario is "Historical Reference" or the cohort is among Age 0-4, Age 5-9, or Age 10-14.
- **Step 7: Calculate Growth Rates:** Uses `growth_rate()` to calculate annual growth rates for the value column. Growth rates are computed as percentage changes between consecutive years within each group specified in `group_cols`.
- **Step 8: Save to File (Optional):** If `outFile` is provided, saves the final processed dataset to a .HAR or .CSV file

Value

A processed data frame containing the summarized and processed IIASA database.

See Also

[aggData](#), [interpolate_spline](#), [interpolate_beers](#), [growth_rate](#)

Examples

```
## Not run:
# Run the routine and save output to a CSV
final_data <- iiasa_gtap(
  aggTxtFile = "path/to/aggFile.agg",
  outFile = "path/to/output.csv"
)

# Run the routine without saving
final_data <- iiasa_gtap(
  aggTxtFile = "path/to/aggFile.agg"
)

## End(Not run)
```

interpolate_beers	<i>Fill Gaps Using Beers Method Interpolation</i>
-------------------	---

Description

This function applies the Beers method interpolation on a grouped data frame. It requires each group to have at least 6 non-NA values and assumes data to be in 5-year intervals. Groups with fewer than 6 non-NA values are excluded from interpolation.

Usage

```
interpolate_beers(input_df, groups, year, values, method = "ordinary")
```

Arguments

input_df	A data frame containing the data to be interpolated.
groups	A vector of column names to group by.
year	The name of the column containing year information.
values	The name of the column containing values for interpolation.
method	A character string specifying the method to be used. Must be one of "ordinary", "modified", "subdivision_ordinary", or "subdivision_modified". - "ordinary" (default): In interpolation, include original data points unchanged; in subdivision, each set of 5 subdivided values sums to the original data point. - "modified": In interpolation, includes some smoothing with only the first and last points unchanged; in subdivision, similar smoothing occurs.

Value

A data frame with interpolated values using the Beers method.

Examples

```
data <- data.frame(
  Scenario = rep(c("Scenario1", "Scenario2"), each = 5),
  Region = rep(c("Region1", "Region2"), each = 5),
  year = rep(c(2000, 2005, 2010, 2015, 2020), 2),
  value = rnorm(10)
)

filled_data <- interpolate_beers(
  input_df = data,
  groups = c("Scenario", "Region"),
  year = "year",
  values = "value"
)
```

interpolate_spline	<i>Fill Gaps with Spline Interpolation</i>
--------------------	--

Description

This function takes a data frame and performs cubic spline interpolation to fill in missing year gaps for specified groups. Groups with fewer than 2 non-NA values are excluded from interpolation.

Usage

```
interpolate_spline(input_df, groups, year, values, method = "fmm")
```

Arguments

input_df	A data frame containing the data to be processed.
groups	A vector of column names to group by (to loop applying spline).
year	The name of the column containing years.
values	The name of the numeric column containing values for interpolation.
method	The method of interpolation. Possible values: Must be one of "fmm", "natural", "periodic", "monoH.FC" or "hyman". <ul style="list-style-type: none"> "fmm" (default): Forsythe, Malcolm, and Moler method. An exact cubic spline is fitted through the four points at each end of the data. This is used for determining end conditions. Not suitable for extrapolation. "natural": Natural spline method. Uses natural splines for interpolation. Linear extrapolation outside the range of x using the slope of the interpolating curve at the nearest data point. "periodic": Periodic spline method. Suitable for periodic data. "monoH.FC": Monotone Hermite spline according to Fritsch and Carlson. Ensures the spline is monotone (increasing or decreasing) if the data are monotone. "hyman": Monotone cubic spline using Hyman filtering of an "fmm" fit for strictly monotonic inputs.

Value

A data frame with gaps in years filled using spline interpolation.

Examples

```
data <- data.frame(
  Scenario = rep(c("Scenario1", "Scenario2"), each = 5),
  Region = rep(c("Region1", "Region2"), each = 5),
  year = rep(c(2000, 2005, 2010, 2015, 2020), 2),
  value = rnorm(10)
)

filled_data <- interpolate_spline(
  input_df = data,
  groups = c("Scenario", "Region"),
  year = "year",
  values = "value"
)
```

read_csv_from_zip

*Combine CSV Files from ZIP Archives***Description**

This function searches for ZIP files in a specified directory that match a given pattern. It then extracts CSV files from these ZIP archives that match another specified pattern. The function can either combine these CSV files vertically into a single data frame or return them as separate data frames in a list. Each data frame in the list is named after its respective CSV file.

Usage

```
read_csv_from_zip(zip_dir, zip_pattern, csv_pattern, combine_vertically = TRUE)
```

Arguments

zip_dir	Directory containing the ZIP files.
zip_pattern	Pattern to match the ZIP file names.
csv_pattern	Pattern to match the CSV file names inside the ZIP archives.
combine_vertically	Logical, if TRUE, the function combines all CSV files vertically into a single data frame. If FALSE, returns a list of data frames, each named after its corresponding CSV file.

Value

Either a single combined data frame or a list of data frames, depending on the value of `combine_vertically`.

Examples

```
# Assuming ZIP files are in 'path/to/zip/files'
combined_data <- combine_csv_from_zip(
  zip_dir = "path/to/zip/files",
  zip_pattern = "zip_file_pattern",
  csv_pattern = "csv_file_pattern",
  combine_vertically = TRUE
```

```

)

# To get a list of data frames instead of a combined one
data_list <- combine_csv_from_zip(
  zip_dir = "path/to/zip/files",
  zip_pattern = "zip_file_pattern",
  csv_pattern = "csv_file_pattern",
  combine_vertically = FALSE
)

```

txtToData

Convert Gempack Text Files into a List of DataFrames

Description

This function reads a Gempack-style text file and converts its sections into a list of DataFrames. Each section in the file is represented as a DataFrame, with rows and columns extracted based on the file's structure.

Usage

```
txtToData(file_path)
```

Arguments

`file_path` Character. The path to the Gempack text file to be processed.

Value

A named list of DataFrames. Each DataFrame corresponds to a section in the text file. The names of the list elements represent the sections, and the DataFrames contain the rows and columns of data extracted from those sections.

See Also

[readLines](#), [data.frame](#)

Examples

```

## Not run:
# Example usage
file_path <- "path_to_gempack_file.txt" # Replace with the actual file path
section_dfs <- txtToData(file_path)

# Accessing a specific section
head(section_dfs[["Section 3"]]) # Preview the DataFrame for a specific section

## End(Not run)

```

updateCorresp

*Update Country Correspondence Data with IIASA Regions***Description**

This function processes and updates a correspondence table between country names and their ISO3 codes using data from the IIASA raw dataset and an existing correspondence table. It cleans the region names, standardizes country names, and ensures alignment with the ISO3 standard. The updated correspondence data can either be saved to a file or returned directly.

Usage

```
updateCorresp(
  iiasa_raw = gtapssp::iiasa_raw,
  corresp_reg = gtapssp::corresp_reg,
  outputFile = NULL
)
```

Arguments

iiasa_raw	A list containing IIASA raw data, typically the output of the updateData function. Default is gtapssp::iiasa_raw.
corresp_reg	A data frame containing the existing correspondence table, typically gtapssp::corresp_reg. This table is merged with the processed IIASA region data.
outputFile	Character. Optional. The file path where the updated correspondence data will be saved. If NULL, the data will not be saved but returned instead. Default is NULL.

Details

The function performs the following steps:

- Extracts unique region names from the iiasa_raw dataset.
- Cleans and standardizes country names, removing invalid entries such as "World" and entries with parentheses.
- Handles specific name corrections, such as replacing "Micronesia" with "Federated States of Micronesia".
- Maps country names to their ISO3 codes using the countrycode package.
- Merges the cleaned IIASA region data with the existing correspondence table (corresp_reg).
- Optionally saves the final updated correspondence data to the specified file.

Value

If outputFile is NULL, returns a data frame with the updated correspondence data. Otherwise, saves the data to the specified file and invisibly returns NULL.

See Also

[countrycode](#), [filter](#), [mutate](#), [right_join](#)

Examples

```
## Not run:
# Update correspondence data and save it to a file
updateCorresp(
  iiasa_raw = gtapssp::iiasa_raw,
  corresp_reg = gtapssp::corresp_reg,
  outputFile = "data/corresp_iiasa_updated.rda"
)

# Update correspondence data and return it directly
updated_data <- updateCorresp(
  iiasa_raw = gtapssp::iiasa_raw,
  corresp_reg = gtapssp::corresp_reg
)

## End(Not run)
```

updateData

Update Data from IIASA Using pyam

Description

This function reads and processes data from the IIASA database using the Python pyam package. It validates the Python environment, configures the appropriate Python executable or Conda environment, and ensures the required Python module is available. The resulting data is saved as a .rda file.

Usage

```
updateData(
  pythonExePath = NULL,
  condaenv = TRUE,
  outputFile = "data/iiasa_raw.rda"
)
```

Arguments

pythonExePath	Character. Path to the Python executable. If NULL, the function will attempt to use a Conda environment specified by condaenv. Default is NULL.
condaenv	Logical. If TRUE, the function will use a Conda environment named "base" if pythonExePath is not provided. Default is TRUE.
outputFile	Character. Path where the processed data will be saved as a .rda file. Default is "data/iiasa_raw.rda".

Details

Note: Downloading the database from IIASA may take several minutes depending on the size of the database and the speed of the connection.

This function requires the pyam Python package to be installed in the specified Python or Conda environment. Ensure that the Python environment is correctly set up before calling the function. Downloading the IIASA database may take several minutes.

Value

A list containing the processed IIASA data:

index Indices of the IamDataFrame.

data Time series data.

meta Metadata.

region Regions.

variable Variables.

unit Units.

year Years.

The data is also saved to the specified output file.

See Also

[use_python](#), [use_condaenv](#), [py_config](#), [import](#)

Examples

```
## Not run:  
# Using a specific Python executable  
updateData(pythonExePath = "C:/path/to/python.exe", outputFile = "data/iiasa_raw.rda")  
  
# Using the default Conda environment  
updateData(condaenv = TRUE, outputFile = "data/iiasa_raw.rda")  
  
## End(Not run)
```

Index

aggData, [2](#), [6](#)
aggData(), [6](#)
arrange, [5](#)

calc_beers, [3](#)
countrycode, [11](#)
csvtorda, [4](#)

data.frame, [10](#)
drop_na, [2](#)

filter, [11](#)

group_by, [2](#), [5](#)
growth_rate, [4](#), [6](#)
growth_rate(), [6](#)

iiasa_gtap, [5](#)
import, [13](#)
interpolate_beers, [6](#), [7](#)
interpolate_beers(), [6](#)
interpolate_spline, [6](#), [8](#)
interpolate_spline(), [6](#)

mutate, [5](#), [11](#)

py_config, [13](#)

read_csv_from_zip, [9](#)
readLines, [10](#)
right_join, [11](#)

txtToData, [10](#)

updateCorresp, [11](#)
updateData, [2](#), [12](#)
use_condaenv, [13](#)
use_python, [13](#)