Package ‘gtaptools’

April 25, 2023

Type Package

Title A set of tools to improve the productivity of CGE modelers.

Version 0.1.0

Description The gtaptools is a package under development that aims to offer a set of functions designed for supporting simulation exercises with CGE (Computable General Equilibrium) models in R language. The primary goal of this package is to facilitate and improve file management, increase the analytical potential of the database, and provide graphical visualizations of simulation results.

Encoding UTF-8

LazyData true

Depends R (>= 3.1)

Imports data.table (>= 1.14.8),
HARr

Remotes github::USDA-ERS/MTED-HARr

RoxygenNote 7.2.3

License GPL (>= 3)

R topics documented:

`agg_har` *Aggregates headers of data in .har structure.*

Description

It aggregates variables from a .har file on disk or an object with the structure exported by the read_har function. It is possible to adopt customized weights and functions to calculate aggregations. The specification of GTAP models (...) through the *model* parameter is supported so that the respective weight variables are automatically detected according to the model being analyzed.

Usage

```
agg_har(
  input_data,
  model = NULL,
  correspondences,
  vars_weighted_mean,
  output_har_file = NULL
)
```

Arguments

`input_data`	It can indicate a path to a .har file or an existing object in the R environment that has the output structure of the read_har function.
`model`	Indicates the CGE model being worked on (Supports only GTAP, ...). For supported models, this information is sufficient to define the variables that must be aggregated through weighted mean.
`correspondences`	A list indicating the original sets and new aggregated sets that will be exported. It can indicate a path to a .csv file or an existing object in the R environment that has the correspondences between the *input_data* sets and the new aggregated sets. The first line will be considered as header and identifier, and must necessarily contain the same name as the set that must be aggregated from *input_data*.
`vars_weighted_mean`	Vector of characters relating the variables that must be grouped with a weighted average of their respective weight variables, in the format c("var" = "weight"). Please note the example section. The sets of the variable and its weight will be made compatible through the aggregation by sum of the weight variable, if necessary.
`output_har_file`	Output .har file name.

`har_shape` *Merge and generate new headers.*

Description

Allows the combination of different databases in data.frame or array format. Generate new variables flexibly from custom functions. Calculations can be performed between headers/variables of different dimensions/sets.

Allows the combination of different databases in data.frame or array format. Generate new variables flexibly from custom functions. Calculations can be performed between headers/variables of different dimensions/sets.

Usage

```
har_shape(
  input_data,
  new_calculated_vars = NULL,
  del_headers = NULL,
  export_sets = NULL,
  output_har_file = NULL
)

har_shape(
  input_data,
  new_calculated_vars = NULL,
  del_headers = NULL,
  export_sets = NULL,
  output_har_file = NULL
)
```

Arguments

`input_data`	It must consist of one or more input databases, which must be separated from each other by sublists (see example). In the case of multiple databases, all will be combined for the final output.Arrays and data.frames must be inside sublists (list(....)) as indicated in the examples section. Aggregations on input data can only be performed on single array and data.frame inputs.
`new_calculated_vars`	New variables resulting from custom calculations between the headers contained in input_data. Each variable's parameters must be informed (it can be *x, y, z ...*), the function *fun* that represents the calculation to be done, the *new_header_name*, and the *sets* for the output structure. The different headers must have at least one similar set so that it is possible to establish correspondence between them. Please note the example section.
`del_headers`	Vector of characters with the names of headers that must be excluded from the output.
`export_sets`	If an output .har file is indicated, it will be created and exported to that .har file. If FALSE, they will not be exported.
`output_har_file`	Output .har file name.

Note

Bear in mind that, for performance reasons, before carrying-out custom calculations between different Headers with different sets, the function aggregates all headers involved in the computation to the output set. That is, in a MAKE(COM, IND)/1CAP(COM) division operation, the first operation is the aggregation for MAKE(COM), and only then is the division calculation processed. Therefore, for example, a weighted average aggregation cannot be done directly. It is recommended to use the agg_har function in this case.

Bear in mind that, for performance reasons, before carrying-out custom calculations between different Headers with different sets, the function aggregates all headers involved in the computation to the output set. That is, in a MAKE(COM, IND)/1CAP(COM) division operation, the first operation is the aggregation for MAKE(COM), and only then is the division calculation processed. Therefore, for example, a weighted average aggregation cannot be done directly. It is recommended to use the agg_har function in this case.

Examples

```
# example code
# -Reads list_df, a list of input data (data.frame(1), list of arrays(2), array(3)) >
# -Aggregates the input (1) for "MAR1" in 3 sets by simple addition (default) >
# -Aggregates the input(3) for "XPLC" into 1 set per average >
# -Deletes headers "MAR1" and "3pur" >
# -Saves the output in a .har file ("gtaptools_shape_example1.har") >
# -Returns the list of "binded_df" arrays.

list_df <- list(
 list(
   input_data = gtaptools::example_df, # 1 - data.frame
   sets = c("COM", "SRC", "MAR"), # sum on IND
   col_values = "Freq",
 new_header_name = "MAR1"
),
gtaptools::example_arrays_har, # 2 - list of arrays
 list(
   input_data = gtaptools::example_arrays_har$xplh, # 3 - array
   sets = c("COM"), # sum in HOU,
   fun = function(x) mean(x),
   new_header_name = "XPLC"
)
)

binded_df <-
  gtaptools::har_shape(
    input_data = list_df,
    del_headers = c("MAR1", "3pur"),
    output_har_file = "gtaptools_shape_example1.har"
  )
```

```
# example code

# -Reads list_df, a list of input data (data.frame(1), list of arrays(2), array(3)) >
# -Aggregates the input (1) for "MAR1" in 3 sets by simple addition (default) >
# -Aggregates the input(3) for "XPLC" into 1 set per average >
# -Deletes headers "MAR1" and "3pur" >
# -Saves the output in a .har file ("gtaptools_shape_example1.har") >
# -Returns the list of "binded_df" arrays.

list_df <- list(
 list(
   input_data = gtaptools::example_df, # 1 - data.frame
   sets = c("COM", "SRC", "MAR"), # sum on IND
   col_values = "Freq",
 new_header_name = "MAR1"
),
gtaptools::example_arrays_har, # 2 - list of arrays
 list(
   input_data = gtaptools::example_arrays_har$xplh, # 3 - array
   sets = c("COM"), # sum in HOU,
   fun = function(x) mean(x),
   new_header_name = "XPLC"
)
)

binded_df <-
  gtaptools::har_shape(
    input_data = list_df,
    del_headers = c("MAR1", "3pur"),
    output_har_file = "gtaptools_shape_example1.har"
  )
```

---

squeeze_sim — *Squeeze the simulation files into a .zip file.*

Description

Scans the .cmf file and selects just essential files for the simulation and compresses them in a .zip file. It also creates a .bat file that makes it easy to run the simulation later. The files that are included are those specified in the .cmf file and that have the extension .tab, .cmf, .sti, .bat, .har, .prm, .shk, .cls, and in the case output = T, .sl4, .upd, .slc. (This function does not support dynamic simulations operationalized in the RunDynam software.)

Usage

```
squeeze_sim(cmf_file, zip_file, add_files = NULL, output = F, bat = T)
```

Arguments

`cmf_file`	Path to .cmf file which manages the simulation.
`zip_file`	Name of the .zip file that will be created.
`add_files`	Vector with the names or extensions of the files that will also be included in the .zip file in addition to the files mentioned in the description.
`output`	Includes simulation output files (default = F).
`bat`	Create a batch file to compile (if necessary) and run the simulation (default = T). For this functionality it is necessary to have Gempack installed.

`summarise_header` *Aggregates headers of data in .har structure.*

Description

Summarizes a single database to an array compatible format for writing to .har files.

Usage

```
summarise_header(
  input_data,
  sets,
  col_values = NULL,
  input_header = NULL,
  fun = function(x) sum(x, na.rm = T),
  new_header_name,
  export_sets = T,
  output_har_file = NULL,
  output_csv_file = NULL,
  ...
)
```

Arguments

`input_data`	An array that has the output structure of the read_har function or a data.frame.
`sets`	Dimensions/columns names that contain the categorical variables for summarizing the output sets. It must be a maximum of 3 characters each.
`col_values`	Name of the numerical columns with the values. (only necessary when input_data is a data.frame)
`input_header`	Name of the reader to be read. (Only necessary when input_data is an .har file)
`fun`	Function used for aggregation in case of non-unique values in sets (default = sum).
`new_header_name`	Name of the new header created.
`export_sets`	If TRUE, the vectors with the set elements are incorporated with the output. If an output .har file is indicated, it will be created and exported to that .har file. If FALSE, they will not be exported.
`output_har_file`	Output .har file name.

`output_csv_file`	Output .csv file name.
`...`	Any additional arguments to be used to write the .csv file through data.table::fwrite, such as separator character (sep = ","), the character for decimal points (dec = "."), etc.

Value

An array or vector of characters (sets) structured in a compatible way to compose a .har file.

Examples

```
# -Take a data.frame as input
#   (Could be a path to a .har file, in which case,
#   instead of col_values, input_header should be specified) >
# -Aggregates the numeric variable "Freq" from 4 sets
#   to 3 sets (COM, SRC, MAR) by simple sum (default) >
# -Write the new header with the name "EXAM"
#   in a .har file ("gtaptools_summarise_example.har") >
# -Save the sets that compose the new header
#   in a separate .har file ("gtaptools_summarise_example_sets.har") >
# -Return an array "example_df_CSM".

example_df_CSM <- gtaptools::summarise_header(
 input_data = gtaptools::example_df,
 sets = c("COM", "SRC", "MAR"),
 col_values = "Freq",
 new_header_name = "EXAM",
 export_sets = "gtaptools_summarise_example_sets.har",
 output_har_file = "gtaptools_summarise_example.har"
)


# To explore the object as a data.frame:
as.data.frame.table(example_df_CSM$EXAM)

#-Get an array as input >
#-Aggregate from the 3 sets of the previous output
#  to 2 sets (SRC, MAR) by simple mean >
#-Save the new header with the name "EXAM" in a .har file
#  ("gtaptools_summarise_example.har") and creates headers for
  each the set that compose it (export_sets = T, its default)>
#-Export the data as unpivot table to a .csv file ("gtaptools_summarise_example.csv") with
#  separator "," (default) and decimal point "." (default).
#-Return an array "example_df_SM".

example_df_SM <- gtaptools::summarise_header(
input_data = example_df_CSM$EXAM,
sets = c("SRC", "MAR"),
#col_values = "Freq",
fun = function(x) mean(x, na.rm = T),
new_header_name = "EXAM",
export_sets = T,
output_har_file = "gtaptools_summarise_example.har",
output_csv_file = "gtaptools_summarise_example.csv",
sep = ",",
dec = "."
)

# To explore the object as a data.frame:
as.data.frame.table(example_df_SM$EXAM)
```

Index