Package 'HARplus'

May 7, 2026

Title Enhanced R Package for 'GEMPACK' .har and .sl4 Files

Version 1.2.0

Description Provides tools for processing and analyzing .har and .sl4 files, making it easier for 'GEMPACK' users and 'GTAP' researchers to handle large economic datasets. It simplifies the management of multiple experiment results, enabling faster and more efficient comparisons without complexity. Users can extract, restructure, and merge data seamlessly, ensuring compatibility across different tools. The processed data can be exported and used in 'R', 'Stata', 'Python', 'Julia', or any software that supports Text, CSV, or 'Excel' formats.

License MIT + file LICENSE

Encoding UTF-8

BugReports https://github.com/bodysbobb/HARplus/issues/

URL https://github.com/bodysbobb/HARplus/, https://www.pattawee-pp.com/HARplus/

Imports openxlsx, haven, stats, utils, tidyr, tools, tidyselect

Suggests knitr, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Pattawee Puangchit [aut, cre]

Maintainer Pattawee Puangchit <ppuangch@purdue.edu>

Repository CRAN

Date/Publication 2026-05-03 05:30:02 UTC

Contents

`compare_var_structure` *Compare Variable Structures Across SL4 and HAR Objects*

Description

Compares variable structures across multiple SL4 and HAR datasets to ensure compatibility. Identifies matching and mismatched variable structures, helping users diagnose inconsistencies.

Usage

```
compare_var_structure(variables = NULL, ..., keep_unique = FALSE)
```

Arguments

`variables`	Character vector. Variable names to compare. Use `NULL` or `"ALL"` to compare all variables.
`...`	Named SL4 or HAR objects to compare.
`keep_unique`	Logical. If `TRUE`, returns unique variable structures across datasets instead of checking for compatibility. Default is `FALSE`.

Details

- Verifies whether variables have consistent structures across multiple datasets.
- Ensures correct ordering of dimensions and checks for structural compatibility.
- If `keep_unique = TRUE`, returns a list of unique variable structures instead of performing a direct comparison.
- Useful for merging or aligning datasets before further processing.
- Helps detect differences in variable dimensions, which may arise due to model updates or dataset variations.

Value

A list containing:

- `match`: A data frame listing variables with identical structures across datasets.
- `diff`: A data frame listing variables with mismatched structures, useful for debugging and alignment.
- If `keep_unique = TRUE`, instead of `match` and `diff`, returns a data frame with distinct variable structures across datasets.

Author(s)

Pattawee Puangchit

See Also

`get_var_structure`, `get_dim_patterns`, `get_dim_elements`

Examples

```
# Import sample data:
har_data1 <- load_harx(system.file("extdata", "TAR10-WEL.har", package = "HARplus"))
har_data2 <- load_harx(system.file("extdata", "SUBT10-WEL.har", package = "HARplus"))

# Compare structure for a single variable across multiple datasets
compare_var_structure("A", har_data1, har_data2)

# Compare structure for multiple variables across multiple datasets
comparison_multiple <- compare_var_structure(c("A", "E1"), har_data1, har_data2)

# Extract unique variable structures across multiple datasets
unique_vars <- compare_var_structure("ALL", har_data1, har_data2, keep_unique = TRUE)
```

---

create_calc_config *Create Calculation Configuration*

Description

Defines calculation settings for generating shock values, including variable mapping, timeline sequence, and exclusion criteria. Used as input for both `shock_calculate` and `shock_calculate_uniform`.

Usage

```
create_calc_config(
  column_mapping = NULL,
  timeline = 1,
  exclude_self_trade = FALSE,
  exclusion_values = NULL
)
```

Arguments

`column_mapping`	Optional named vector mapping initial and target columns (e.g., `c(COMM = "COMM", REG = "REG", REG.1 = "REG.1")`).
`timeline`	Numeric or character range defining simulation periods. • Example: "1-5" expands to 1:5.
`exclude_self_trade`	Logical; if TRUE, removes intra-region records. Default is FALSE.
`exclusion_values`	Optional named list of elements to exclude from calculation.

Details

- Controls column alignment between initial and target datasets
- Supports numeric or range-based timeline definitions (e.g., "1-10")
- Excludes self-trade or specified region/sector pairs if configured
- Provides core metadata for shock calculation functions

Value

A list containing:

- `column_mapping`: variable mapping between datasets
- `timeline`: expanded numeric sequence of simulation periods
- `exclude_self_trade`: logical flag
- `exclusion_values`: exclusion list by dimension

Author(s)

Pattawee Puangchit

See Also

`create_initial_config`, `create_target_config`, `shock_calculate`, `shock_calculate_uniform`

Examples

```
# Example: Define Calculation Configuration
calc <- create_calc_config(
  column_mapping      = c(COMM = "COMM", REG = "REG", REG.1 = "REG.1"),
  timeline            = "1-5",
  exclude_self_trade  = TRUE
)
```

`create_initial_config` *Create Initial Value Configuration*

Description

Defines the configuration for loading the initial dataset, including path, format, variable header, and value column name. Used as input for `shock_calculate` and `shock_calculate_uniform`.

Usage

```
create_initial_config(path, format, header, value_col = "Value")
```

Arguments

`path`	Path to the initial data file.
`format`	File format of the initial dataset. Must be "har" or "sl4".
`header`	Header name within the HAR or SL4 file to extract.
`value_col`	Name of the column containing numeric values. Default is "Value".

Details

- Supports `HAR` and `SL4` file formats
- Specifies the header name to extract from the dataset
- Allows custom naming for the value column (`Value` by default)

Value

A list containing:

- `path`: input file path
- `format`: file format ("har" or "sl4")
- `header`: target header name
- `value_col`: column name for numeric values

Author(s)

Pattawee Puangchit

See Also

`create_target_config`, `create_calc_config`, `shock_calculate`, `shock_calculate_uniform`

Examples

```
# Example: Define Initial Configuration
initial <- create_initial_config(
  path   = "D:/Data/taxrates_2017.har",
  format = "har",
  header = "rTMS"
)
```

`create_target_config` *Create Target Value Configuration*

Description

Defines the configuration for loading the target dataset, which represents post-adjustment or comparative rate values. Supports multiple file formats.

Usage

```
create_target_config(
  path = NULL,
  type = NULL,
  header = NULL,
  value_col = "Value"
)
```

Arguments

`path`	Optional path to the target data file.
`type`	Optional file type for the target dataset ("har", "sl4", "csv", or "xlsx").
`header`	Optional header name within the HAR or SL4 file to extract.
`value_col`	Column name containing numeric target values. Default is "Value".

Details

- Supports `HAR`, `SL4`, `CSV`, and `XLSX` file formats
- Can also represent a uniform numeric target value when no file path is provided
- Used in combination with `create_initial_config` for shock computation

Value

A list containing:

- `path`: file path to target data
- `type`: file format (lowercase)
- `header`: header name in HAR/SL4 file
- `value_col`: column name for target values

Author(s)

Pattawee Puangchit

See Also

create_initial_config, create_calc_config, shock_calculate, shock_calculate_uniform

Examples

```
# Example: Define Target Configuration
target <- create_target_config(
  path   = "D:/Data/taxrates_2019.har",
  type   = "har",
  header = "rTMS"
)
```

---

export_data *Export Data to Various Formats (CSV/STATA/TEXT/RDS/XLSX)*

Description

Exports structured SL4 or HAR data to multiple file formats, including CSV, Stata, TXT, RDS, and XLSX. Supports nested lists, automatic subfolder creation, and multi-sheet Excel exports.

Usage

```
export_data(
  data,
  output_path,
  format = "csv",
  prefix = "",
  create_subfolder = FALSE,
  multi_sheet_xlsx = FALSE,
  xlsx_filename = NULL,
  report_output = FALSE
)
```

Arguments

data	A list or data frame. The SL4 or HAR data to export.
output_path	Character. The base output directory or file path.
format	Character. The export format ("csv", "stata", "txt", "rds", "xlsx"). Default is "csv".
prefix	Character. An optional prefix added to exported file names. Default is "" (empty).
create_subfolder	Logical. If TRUE, creates a subfolder for each format. Default is FALSE.

`multi_sheet_xlsx` Logical. If TRUE, exports lists as multi-sheet XLSX files.

`xlsx_filename` An optional filename for the XLSX file (used when `multi_sheet_xlsx = TRUE`).

`report_output` Logical. If TRUE, generates an export report.

Details

- Supports exporting data in `"csv"`, `"stata"`, `"txt"`, `"rds"`, and `"xlsx"` formats.
- Handles nested lists and exports each data frame individually.
- Optionally creates subfolders for each format (`create_subfolder = TRUE`).
- Customizes file names using `prefix`.
- When multi_sheet_xlsx = TRUE, all exported data is stored in a **single Excel workbook**, with each dataset as a separate sheet.
- If exporting to Stata (`"stata"` format), column names containing `.` will be replaced with `_` to ensure compatibility.
- If `multi_sheet_xlsx = TRUE`, list elements are exported as separate sheets in a single XLSX file.
- The function creates necessary directories if they do not exist.

Value

A list containing the file paths of the exported data.

Author(s)

Pattawee Puangchit

See Also

`pivot_data`, `get_data_by_var`, `get_data_by_dims`

Examples

```
# Import sample data:
sl4_data <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))

# Extract data
data_multiple <- get_data_by_var(c("qo", "pca"), sl4_data)

# Export
export_data(data_multiple, file.path(tempdir(), "output_directory"),
           format = c("csv", "xlsx", "stata", "txt", "rds"),
           create_subfolder = TRUE,
           multi_sheet_xlsx = TRUE)
```

`get_data_by_dims` *Extract Data by Dimension Patterns from SL4 or HAR Objects*

Description

Retrieves structured data from SL4 or HAR objects based on specified dimension patterns. Supports multiple experiments and merging datasets while maintaining structured dimension metadata.

Usage

```
get_data_by_dims(
  patterns = NULL,
  ...,
  experiment_names = NULL,
  subtotal_level = FALSE,
  rename_cols = NULL,
  merge_data = FALSE,
  pattern_mix = FALSE
)
```

Arguments

`patterns`	Character vector. Dimension patterns to extract. Use `"ALL"` or `NULL` to extract all available patterns.
`...`	One or more SL4 or HAR data objects loaded using `load_sl4x()` or `load_harx()`.
`experiment_names`	Character vector. Names assigned to each dataset. If `NULL`, names are inferred.
`subtotal_level`	Character or logical. Determines which decomposition levels to retain: • `"total"`: Keeps only `"TOTAL"` values. • `"decomposed"`: Keeps only decomposed values (excludes `"TOTAL"`). • `"all"`: Keeps all rows. • `TRUE`: Equivalent to `"all"`, retaining both `"TOTAL"` and decomposed values. • `FALSE`: Equivalent to `"total"`, keeping only `"TOTAL"` values.
`rename_cols`	Named vector. Column name replacements (`c("old_name" = "new_name")`).
`merge_data`	Logical. If `TRUE`, attempts to merge data across multiple experiments. Default is `FALSE`.
`pattern_mix`	Logical. If `TRUE`, allows flexible pattern matching, ignoring dimension order. Default is `FALSE`.

Details

- Extracts variables matching specified dimension patterns.
- Allows for flexible pattern matching (`pattern_mix = TRUE`).
- Supports merging data across multiple experiments (`merge_data = TRUE`).

- Provides column renaming functionality (`rename_cols`).
- Handles subtotal filtering (`subtotal_level`), controlling whether `"TOTAL"` or decomposed values are retained.

Value

A structured list of extracted data:

- If `merge_data = FALSE`, returns a named list where each element corresponds to an experiment.
- If `merge_data = TRUE`, returns a named list of all merged data

Author(s)

Pattawee Puangchit

See Also

`get_data_by_var`, `group_data_by_dims`

Examples

```
# Import sample data:
sl4_data <- load_sl4x(
  system.file("extdata", "TAR10.sl4", package = "HARplus")
)
sl4_data1 <- load_sl4x(
  system.file("extdata", "SUBT10.sl4", package = "HARplus")
)

# Extract data for a single dimension pattern
data_single_pattern <- get_data_by_dims(
  "comm*reg",
  sl4_data
)

# Extract multiple dimension patterns
data_multiple_patterns <- get_data_by_dims(
  c("comm*reg", "REG*ACTS"),
  sl4_data
)

# Extract all dimension patterns separately from multiple datasets
data_all_patterns <- get_data_by_dims(
  NULL,
  sl4_data, sl4_data1,
  merge_data = FALSE
)

# Merge data for identical patterns across multiple datasets
data_merged_patterns <- get_data_by_dims(
  NULL,
```

```
  sl4_data, sl4_data1,
  merge_data = TRUE
)

# Merge data while allowing interchangeable dimensions (e.g., A*B = B*A)
data_pattern_mixed <- get_data_by_dims(
  NULL,
  sl4_data, sl4_data1,
  merge_data = TRUE,
  pattern_mix = TRUE
)

# Retain only "TOTAL" values
data_total_only <- get_data_by_dims(
  "comm*reg",
  sl4_data,
  subtotal_level = "total"
)
data_total_only_alt <- get_data_by_dims(
  "comm*reg",
  sl4_data,
  subtotal_level = FALSE
)

# Retain only decomposed components
data_decomposed_only <- get_data_by_dims(
  "comm*reg",
  sl4_data,
  subtotal_level = "decomposed"
)

# Retain all value levels
data_all_decomp <- get_data_by_dims(
  "comm*reg",
  sl4_data,
  subtotal_level = "all"
)
data_all_decomp_alt <- get_data_by_dims(
  "comm*reg",
  sl4_data,
  subtotal_level = TRUE
)

# Rename specific columns
data_renamed <- get_data_by_dims(
  "comm*reg",
  sl4_data,
  rename_cols = c(REG = "Region", COMM = "Commodity")
)

# Merge data with custom experiment names
data_merged_experiments <- get_data_by_dims(
  "comm*reg",
```

```
  sl4_data, sl4_data1,
  experiment_names = c("EXP1", "EXP2"),
  merge_data = TRUE
)
```

`get_data_by_var` *Extract Variable Data from SL4 or HAR Objects*

Description

Extracts structured data for one or more variables from SL4 or HAR objects, transforming array-like data into a tidy format.

Usage

```
get_data_by_var(
  var_names = NULL,
  ...,
  experiment_names = NULL,
  subtotal_level = FALSE,
  rename_cols = NULL,
  merge_data = FALSE
)
```

Arguments

`var_names`	Character vector. Variable names to extract. Use `"ALL"` or `NULL` to extract all available variables.
`...`	One or more SL4 or HAR data objects loaded using `load_sl4x()` or `load_harx()`.
`experiment_names`	Character vector. Names assigned to each dataset. If `NULL`, names are inferred.
`subtotal_level`	Character or logical. Determines which decomposition levels to retain: • `"total"`: Keeps only `"TOTAL"` values. • `"decomposed"`: Keeps only decomposed values (excludes `"TOTAL"`). • `"all"`: Keeps all rows. • `TRUE`: Equivalent to `"all"`, retaining both `"TOTAL"` and decomposed values. • `FALSE`: Equivalent to `"total"`, keeping only `"TOTAL"` values.
`rename_cols`	Named vector. Column name replacements (`c("old_name" = "new_name")`).
`merge_data`	Logical. If `TRUE`, attempts to merge data across multiple experiments. Default is `FALSE`.

Details

- Retrieves specific variables, multiple variables, or all available variables from SL4 or HAR datasets.
- Supports merging data from multiple experiments (`merge_data = TRUE`).
- Allows renaming of column names (`rename_cols`).
- Handles subtotal filtering (`subtotal_level`), controlling whether `"TOTAL"` or decomposed values are retained.

Value

A list of structured data:

- If `merge_data = FALSE`, returns a named list where each element corresponds to an experiment.
- If `merge_data = TRUE`, returns a named list of all merged data

Author(s)

Pattawee Puangchit

See Also

`get_data_by_dims`, , `group_data_by_dims`, `load_sl4x`, `load_harx`

Examples

```
# Import sample data:
sl4_data <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))
sl4_data1 <- load_sl4x(system.file("extdata", "SUBT10.sl4", package = "HARplus"))

# Extract a single variable
data_qo <- get_data_by_var("qo", sl4_data)

# Extract multiple variables
data_multiple <- get_data_by_var(c("qo", "qgdp"), sl4_data)

# Extract all variables separately from multiple datasets
data_all <- get_data_by_var(NULL, sl4_data, sl4_data1, merge_data = FALSE)

# Merge variable data across multiple datasets
data_merged <- get_data_by_var(NULL, sl4_data, sl4_data1, merge_data = TRUE)

# Retain only "TOTAL" values, removing decomposed components (subtotal_level = "total" or FALSE)
data_total_only <- get_data_by_var("qo", sl4_data, subtotal_level = "total")
data_total_only_alt <- get_data_by_var("qo", sl4_data, subtotal_level = FALSE)

# Retain only decomposed components, removing "TOTAL" (subtotal_level = "decomposed")
data_decomposed_only <- get_data_by_var("qo", sl4_data, subtotal_level = "decomposed")

# Retain all value levels (subtotal_level = "all" or TRUE)
```

```
data_all_decomp <- get_data_by_var("qo", sl4_data, subtotal_level = "all")
data_all_decomp_alt <- get_data_by_var("qo", sl4_data, subtotal_level = TRUE)

# Rename specific columns
data_renamed <- get_data_by_var("qo", sl4_data, rename_cols = c(REG = "Region", COMM = "Commodity"))

# Merge data across multiple datasets with custom experiment names
data_merged_experiments <- get_data_by_var("qo", sl4_data, sl4_data1,
experiment_names = c("EXP1", "EXP2"),
merge_data = TRUE)
```

`get_dim_elements` *Get Dimension Elements from SL4 and HAR Objects*

Description

Extracts and lists unique dimension elements (e.g., `REG`, `COMM`, `ACTS`) from one or more datasets.

Usage

```
get_dim_elements(..., keep_unique = FALSE)
```

Arguments

`...`	One or more structured SL4 or HAR objects containing dimension information.
`keep_unique`	Logical. If `TRUE`, returns only unique dimension elements across inputs. Default is `FALSE`.

Value

A data frame containing unique dimension elements.

Author(s)

Pattawee Puangchit

See Also

`get_dim_patterns`, `get_var_structure`

Examples

```
# Import sample data:
sl4_data1 <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))
sl4_data2 <- load_sl4x(system.file("extdata", "SUBT10.sl4", package = "HARplus"))


# Extract dimension elements from a single dataset
```

```
get_dim_elements(sl4_data1)

# Extract dimension elements from multiple datasets
get_dim_elements(sl4_data1, sl4_data2)

# Extract unique dimension elements across datasets
get_dim_elements(sl4_data1, sl4_data2, keep_unique = TRUE)
```

`get_dim_patterns` *Get Dimension Patterns from SL4 and HAR Objects*

Description

Extracts and lists unique dimension patterns (e.g., `REG*COMM`, `REG*REG*ACTS`) from one or more datasets.

Usage

```
get_dim_patterns(..., keep_unique = FALSE)
```

Arguments

`...`	One or more structured SL4 or HAR objects containing dimension information.
`keep_unique`	Logical. If `TRUE`, returns only unique dimension patterns. Default is `FALSE`.

Details

- Extracts dimension structure details from the dataset.
- If multiple datasets are provided, combines their dimension information.
- If `keep_unique = TRUE`, returns only distinct dimension patterns.

Value

A data frame containing:

- `DimPattern`: The unique dimension patterns.

Author(s)

Pattawee Puangchit

See Also

`get_dim_elements`, `get_var_structure`

Examples

```
# Import sample data:
sl4_data <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))
sl4_data2 <- load_sl4x(system.file("extdata", "SUBT10.sl4", package = "HARplus"))

# Extract dimension patterns
get_dim_patterns(sl4_data)

# Extract only unique dimension patterns across datasets
get_dim_patterns(sl4_data, sl4_data2, keep_unique = TRUE)
```

---

get_var_structure *Get Variable Structure Summary from SL4 and HAR Objects*

Description

Generates a summary of the variables within one or more SL4 or HAR objects, listing their dimension sizes, structures, and optionally, column and observation counts.

Usage

```
get_var_structure(variables = NULL, ..., include_col_size = FALSE)
```

Arguments

`variables`	Character vector. Variable names to summarize. Use `NULL` or `"ALL"` to summarize all variables.
`...`	One or more SL4 or HAR objects created using `load_sl4x()` or `load_harx()`.
`include_col_size`	Logical. If `TRUE`, includes column and observation counts. Default is `FALSE`.

Details

- Extracts dimension structures for variables in one or more SL4 or HAR datasets.
- If `include_col_size = TRUE`, adds column and observation counts.
- Supports multiple datasets and returns results as a named list, with each dataset's summary stored separately.
- Can summarize specific variables or `"ALL"`.

Value

A named list, where each element contains a data frame with:

- `Variable`: The variable name.
- `Dimensions`: The associated dimensions.

- `DimSize`: The number of dimensions.
- `DataShape`: The shape of the data (e.g., `10x20x30`).
- `No.Col`: (Optional) The number of columns.
- `No.Obs`: (Optional) The number of observations.

Author(s)

Pattawee Puangchit

See Also

`get_dim_patterns`, `get_dim_elements`

Examples

```
# Import data sample:
sl4_data <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))
sl4_data1 <- load_sl4x(system.file("extdata", "SUBT10.sl4", package = "HARplus"))

# Get summary for all variables in a single dataset
get_var_structure(data_obj = sl4_data)

# Get summary for specific variables
get_var_structure(c("gdp", "trade"), sl4_data)

# Include column and observation counts
get_var_structure("ALL", sl4_data, include_col_size = TRUE)

# Compare structures across multiple datasets
get_var_structure("ALL", sl4_data, sl4_data1)

# Include column and observation counts across multiple datasets
get_var_structure("ALL", sl4_data, sl4_data1, include_col_size = TRUE)
```

---

group_data_by_dims *Group Data by Dimension Patterns in SL4 or HAR Objects*

Description

Groups extracted SL4 or HAR data based on specified dimension structures and priority rules. Supports automatic renaming, merging, subtotal filtering, and structured metadata handling.

Usage

```
group_data_by_dims(
  patterns = NULL,
  ...,
  priority,
  rename_cols = NULL,
  experiment_names = NULL,
  subtotal_level = FALSE,
  auto_rename = FALSE
)
```

Arguments

patterns	Character vector. Dimension patterns to extract. Use "ALL" or NULL to extract all available patterns.
...	One or more SL4 or HAR objects loaded using load_sl4x() or load_harx().
priority	Named list. Specifies priority dimension elements (c("group_name" = c("dim1", "dim2"))).
rename_cols	Named vector. Column name replacements (c("old_name" = "new_name")).
experiment_names	Character vector. Names assigned to each dataset. If NULL, names are inferred.
subtotal_level	Character or logical. Determines which decomposition levels to retain: • "total": Keeps only "TOTAL" values. • "decomposed": Keeps only decomposed values (excludes "TOTAL"). • "all": Keeps all rows. • TRUE: Equivalent to "all", retaining both "TOTAL" and decomposed values. • FALSE: Equivalent to "total", keeping only "TOTAL" values.
auto_rename	Logical. If TRUE, automatically renames dimensions for consistency. Default is FALSE.

Details

- Groups extracted variables based on dimension elements.
- Applies predefined priority rules to structure the data.
- Allows automatic renaming of dimensions (auto_rename = TRUE).
- Supports merging of grouped data across multiple experiments.
- Handles subtotal filtering (subtotal_level), controlling whether "TOTAL" or decomposed values are retained.

Value

A structured list of grouped data:

- A named list where each element corresponds to a dimension size group (e.g., "2D", "3D").
- Each group contains dimension-grouped data based on priority rules.
- If unmerged data exists, includes a report attribute detailing merge issues.

Author(s)

Pattawee Puangchit

See Also

get_data_by_dims, get_data_by_var, load_sl4x, load_harx

Examples

```
# Import sample data
sl4_data1 <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))
sl4_data2 <- load_sl4x(system.file("extdata", "SUBT10.sl4", package = "HARplus"))

# Case 1: Multiple priority levels (Sector then Region) with auto_rename
priority_list <- list(
  "Sector" = c("COMM", "ACTS"),
  "Region" = c("REG")
)
grouped_data_multiple <- group_data_by_dims(
  patterns = "ALL",
  sl4_data1,
  priority = priority_list,
  auto_rename = TRUE
)

# Case 2: Single priority (Region only) with auto_rename
priority_list <- list("Region" = c("REG"))
grouped_data_single <- group_data_by_dims(
  patterns = "ALL",
  sl4_data1, sl4_data2,
  priority = priority_list,
  auto_rename = TRUE
)

# Case 3: Multiple priorities without auto_rename
priority_list <- list(
  "Sector" = c("COMM", "ACTS"),
  "Region" = c("REG")
)
grouped_data_no_rename <- group_data_by_dims(
  patterns = "ALL",
  sl4_data1,
  priority = priority_list,
  auto_rename = FALSE
)
```

---

load_harx *Load and Process HAR Files with Header Selection*

Description

Reads a GEMPACK HAR file and extracts structured data while maintaining compatibility with standard HAR formats. Provides flexibility in naming conventions and header selection.

Usage

```
load_harx(
  file_path,
  coefAsname = FALSE,
  lowercase = FALSE,
  select_header = NULL
)
```

Arguments

`file_path`	Character. The file path to the HAR file.
`coefAsname`	Logical. If `TRUE`, replaces four-letter headers with coefficient names when available. Default is `FALSE`.
`lowercase`	Logical. If `TRUE`, converts all variable names to lowercase. Default is `FALSE`.
`select_header`	Character vector. Specific headers to read; if `NULL`, all headers are read. Example: `select_header = c("A", "E1")`.

Details

- Uses `load_harplus()` internally for efficient HAR file reading.
- Allows optional conversion of variable names to lowercase (`lowercase = TRUE`).
- Supports coefficient-based naming (`coefAsname = TRUE`).
- Enables selective header extraction via `select_header = c("A", "E1")`.
- Returns structured data with explicit dimension names and sizes.

Value

A structured list containing:

- `data`: Extracted HAR variable data stored as matrices, arrays, or vectors.
- `dimension_info`: A list with:
  - `dimension_string`: A textual representation of dimensions (e.g., "REG*COMM*YEAR").
  - `dimension_names`: The names of each dimension.
  - `dimension_sizes`: The size of each dimension.

Author(s)

Pattawee Puangchit

See Also

`load_sl4x`, `get_data_by_var`, `get_data_by_dims`

Examples

```
# Path to example files
har_path <- system.file("extdata", "TAR10-WEL.har", package = "HARplus")

# Basic loading
har_data <- load_harx(har_path)

# Load with coefficient names
har_data_coef <- load_harx(har_path, coefAsname = TRUE)

# Load with lowercase names
har_data_lower <- load_harx(har_path, lowercase = TRUE)

# Load specific headers
har_selected <- load_harx(har_path, select_header = c("A", "E1"))

# Load with multiple options
har_combined <- load_harx(har_path,
                          coefAsname = TRUE,
                          lowercase = TRUE,
                          select_header = c("A", "E1"))
```

---

load_sl4x *Load and Process SL4 Files with Enhanced Options*

Description

Reads an SL4 file and processes its structured data into an enhanced SL4 object. Extracts structured variable information, dimensions, and handles subtotal columns.

Usage

```
load_sl4x(file_path, lowercase = FALSE, select_header = NULL)
```

Arguments

file_path	Character. The full path to the SL4 file to be read.
lowercase	Logical. If TRUE, converts all variable names to lowercase. Default is FALSE.
select_header	Character vector. Specific headers to extract; if NULL, all headers are read.

Details

- Uses load_harplus() internally for optimized SL4 file reading.
- Extracts variable names, dimension structures, and metadata.
- Converts variable names to lowercase if lowercase = TRUE.
- Allows the selection of specific headers using select_header.
- Returns structured data with explicit dimension names and sizes.

Value

A structured list containing:

- `data`: Extracted SL4 variable data, stored as arrays or matrices.
- `dimension_info`: A list with:
  - `dimension_string`: A textual representation of dimensions (e.g., "REG*COMM*YEAR").
  - `dimension_names`: The names of each dimension.
  - `dimension_sizes`: The size of each dimension.

Author(s)

Pattawee Puangchit

See Also

`load_harx`, `get_data_by_var`, `get_data_by_dims`

Examples

```
# Path to example files
sl4_path <- system.file("extdata", "TAR10.sl4", package = "HARplus")

# Basic loading
sl4_data <- load_sl4x(sl4_path)

# Load with lowercase names
sl4_data_lower <- load_sl4x(sl4_path, lowercase = TRUE)

# Load specific headers
sl4_selected <- load_sl4x(sl4_path, select_header = c("qo", "qgdp"))
```

`pivot_data` *Pivot Data from SL4 or HAR Objects*

Description

Transforms long-format SL4 or HAR data into wide format by pivoting selected columns. Supports both single data frames and nested lists.

Usage

```
pivot_data(data_obj, pivot_cols, name_repair = "unique")
```

Arguments

`data_obj`	A list or data frame. The SL4 or HAR data to pivot.
`pivot_cols`	Character vector. Column names to use as pivot keys.
`name_repair`	Character. Method for handling duplicate column names (`"unique"`, `"minimal"`, `"universal"`). Default is `"unique"`.

Details

- Uses `tidyr::pivot_wider()` internally to reshape data.
- Allows multiple columns to be pivoted simultaneously.
- Recursively processes nested lists, ensuring all data frames are transformed.

Value

A transformed data object where the specified `pivot_cols` are pivoted into wide format.

Author(s)

Pattawee Puangchit

See Also

`get_data_by_var`, `get_data_by_dims`

Examples

```
# Import sample data:
sl4_data <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))

# Extract multiple variables
data_multiple <- get_data_by_var(c("qo", "qxs"), sl4_data)

# Pivot a single column
pivoted_data <- pivot_data(data_multiple, pivot_cols = "REG")

# Pivot multiple columns
pivoted_data_multi <- pivot_data(data_multiple, pivot_cols = c("REG", "COMM"))
```

`pivot_data_hierarchy` *Create Hierarchical Pivot Table from SL4 or HAR Objects*

Description

Creates hierarchical pivot tables from structured SL4 or HAR data, with optional Excel export. Supports both single data frames and nested lists, preserving dimension hierarchies.

Usage

```
pivot_data_hierarchy(
  data_obj,
  pivot_cols,
  name_repair = "unique",
  export = FALSE,
  file_path = NULL,
  xlsx_filename = NULL
)
```

Arguments

data_obj	A list or data frame. The SL4 or HAR data to pivot.
pivot_cols	Character vector. Column names to use as pivot keys in order of hierarchy.
name_repair	Character. Method for handling duplicate column names ("unique", "minimal", "universal"). Default is "unique".
export	Logical. If TRUE, exports result to Excel. Default is FALSE.
file_path	Character. Required if export = TRUE. The path for Excel export.
xlsx_filename	Character. The name for the Excel file when using multi_sheet_xlsx. If NULL, uses the name of the dataset. Default is NULL.

Details

- Transforms data into hierarchical pivot format with nested column headers.
- Supports multiple pivot columns in specified order (e.g., REG > COMM).
- Handles both single data frames and nested list structures.
- Optional direct export to Excel with formatted hierarchical headers.
- Uses efficient data processing with tidyr and openxlsx.

Value

A pivoted data object with hierarchical structure:

- If input is a data frame: Returns a hierarchical pivot table.
- If input is a list: Returns a nested list of hierarchical pivot tables.
- If export = TRUE: Invisibly returns the pivoted object after Excel export.

Author(s)

Pattawee Puangchit

See Also

pivot_data

Examples

```
# Import sample data:
sl4_data <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))

# Extract data
data_multiple <- get_data_by_var(c("qo", "pca"), sl4_data)

# Create hierarchical pivot without export
pivot_hier <- pivot_data_hierarchy(data_multiple,
                                   pivot_cols = c("REG", "COMM"))

# Create and export to Excel in one step
pivot_export <- pivot_data_hierarchy(data_multiple,
                                     pivot_cols = c("REG", "COMM"),
                                     export = TRUE,
                                   file_path = file.path(tempdir(), "pivot_output.xlsx"))
```

---

rename_dims *Rename Dimensions in SL4 or HAR Objects*

Description

Renames dimension and list names in structured SL4 or HAR objects.

Usage

```
rename_dims(data_obj, mapping_df, rename_list_names = FALSE)
```

Arguments

data_obj	A structured SL4 or HAR object.
mapping_df	A two-column data frame where the first column (old) contains the current names, and the second column (new) contains the new names.
rename_list_names	Logical. If TRUE, renames list element names. Default is FALSE.

Details

- Replaces old dimension names with new ones as specified in mapping_df.
- If rename_list_names = TRUE, renames list element names as well.
- Ensures consistency across SL4 and HAR datasets.

Value

The modified SL4 or HAR object with updated dimension names and, optionally, updated list names.

Author(s)

Pattawee Puangchit

See Also

get_data_by_var, get_data_by_dims

Examples

```
# Import sample data:
sl4_data <- load_sl4x(system.file("extdata", "TAR10.sl4", package = "HARplus"))

# Define a renaming map
mapping_df <- data.frame(
  old = c("REG", "COMM"),
  new = c("Region", "Commodity")
)

# Rename columns in the dataset
rename_dims(sl4_data, mapping_df)

# Rename both columns and list names
rename_dims(sl4_data, mapping_df, rename_list_names = TRUE)
```

---

save_har *Save Data to GEMPACK HAR Format*

Description

Writes a named list of R objects to a GEMPACK-compatible HAR file, including character sets, mapping vectors, integer matrices, numeric arrays, sparse numeric arrays, and data frames reshaped to arrays.

Usage

```
save_har(
  data_list,
  file_path,
  dimensions = NULL,
  value_cols = NULL,
  header_type = NULL,
  mappings = NULL,
  long_desc = NULL,
  coefficients = NULL,
  export_sets = TRUE,
  lowercase = TRUE,
  dim_order = NULL,
```

```
  dim_rename = NULL,
  force_sparse = NULL,
  max_chunk = 2e+06
)
```

Arguments

`data_list`	Named list of objects to write. List names are used as HAR header names after conversion to uppercase and truncation to four characters.
`file_path`	Character string giving the output HAR file path.
`dimensions`	Optional named list. For data-frame inputs, each element gives the columns used as array dimensions.
`value_cols`	Optional named list or named character vector. For data-frame inputs, each element gives the numeric value column. Defaults to `"Value"` when omitted.
`header_type`	Optional named list or named character vector giving explicit header roles. Accepted values are `"auto"`, `"set"`, `"mapping"`, `"real"`, `"sparse"`, and `"integer"`.
`mappings`	Optional named list. Each element must be `c(source_set, destination_set)` for the corresponding mapping header.
`long_desc`	Optional named list or named character vector of long header descriptions.
`coefficients`	Optional named list or named character vector of coefficient names for numeric headers.
`export_sets`	Logical. If `TRUE`, dimension sets from numeric arrays are written as character headers unless already supplied. Default is `TRUE`.
`lowercase`	Logical. If `TRUE`, character elements and dimension values are converted to lowercase during data-frame and ordering processing. Default is `TRUE`.
`dim_order`	Optional dimension-ordering specification. Can be `NULL`, a data frame, a named list, or a path to a CSV or Excel file.
`dim_rename`	Optional named list for renaming array dimensions in the HAR output.
`force_sparse`	Optional character vector of headers to write in sparse numeric format.
`max_chunk`	Integer. Maximum number of elements per dense numeric data chunk. Default is `2e6`.

Value

Invisibly returns a list containing the output path, written headers, and counts of set, data, and mapping headers.

Author(s)

Pattawee Puangchit

See Also

`load_harx`, `load_sl4x`

Examples

```
# Example 1: Save one numeric data frame
REG <- c("USA", "EU", "ROW")
COLUMN <- c("alloc_A1", "tot_E1")
WELF <- expand.grid(REG = REG, COLUMN = COLUMN, stringsAsFactors = FALSE)
WELF$Value <- seq_len(nrow(WELF))

save_har(
  data_list = list(WELF = WELF),
  file_path = file.path(tempdir(), "output_single.har"),
  dimensions = list(WELF = c("REG", "COLUMN")),
  value_cols = list(WELF = "Value"),
  long_desc = list(WELF = "Welfare Decomposition"),
  coefficients = list(WELF = "WELF"),
  export_sets = TRUE,
  lowercase = FALSE
)

# Example 2: Save multiple numeric data frames
DECOM <- expand.grid(REG = REG, ALLOCEFF = c("A1", "A2"), stringsAsFactors = FALSE)
DECOM$Value <- seq_len(nrow(DECOM))

save_har(
  data_list = list(WELF = WELF, DECOM = DECOM),
  file_path = file.path(tempdir(), "output_multi.har"),
  dimensions = list(
    WELF = c("REG", "COLUMN"),
    DECOM = c("REG", "ALLOCEFF")
  ),
  value_cols = list(
    WELF = "Value",
    DECOM = "Value"
  ),
  long_desc = list(
    WELF = "Welfare Decomposition",
    DECOM = "Allocative efficiency effect"
  ),
  coefficients = list(
    WELF = "WELF",
    DECOM = "DECOM"
  ),
  export_sets = TRUE,
  lowercase = FALSE
)

# Example 3: Save a mapping vector
SC <- c("AF", "AP", "BA")
GSEC <- c("OCR", "V_F", "GRO")
MASC <- c(AF = "OCR", AP = "V_F", BA = "GRO")

save_har(
  data_list = list(SC = SC, GSEC = GSEC, MASC = MASC),
```

```
  file_path = file.path(tempdir(), "mapping.har"),
  mappings = list(MASC = c("SC", "GSEC")),
  long_desc = list(
    SC = "Set SC",
    GSEC = "Set GSEC",
    MASC = "Mapping SC to GSEC"
  ),
  export_sets = FALSE,
  lowercase = FALSE
)

# Example 4: Save mixed headers
CODE <- matrix(as.integer(c(1, 2, 3, 4)), nrow = 2)
TAX <- expand.grid(SC = SC, REG = REG, stringsAsFactors = FALSE)
TAX$Value <- c(0, 0, 0, 1.2, 0, 0, 0, 0, 2.5)

save_har(
  data_list = list(
    WELF = WELF,
    REG = REG,
    SC = SC,
    GSEC = GSEC,
    MASC = MASC,
    TAX = TAX,
    CODE = CODE
  ),
  file_path = file.path(tempdir(), "output_mixed.har"),
  dimensions = list(
    WELF = c("REG", "COLUMN"),
    TAX = c("SC", "REG")
  ),
  value_cols = list(
    WELF = "Value",
    TAX = "Value"
  ),
  mappings = list(
    MASC = c("SC", "GSEC")
  ),
  long_desc = list(
    WELF = "Welfare Decomposition",
    REG = "Set REG",
    SC = "Set SC",
    GSEC = "Set GSEC",
    MASC = "Mapping SC to GSEC",
    TAX = "Sparse tax example",
    CODE = "Integer matrix example"
  ),
  coefficients = list(
    WELF = "WELF",
    TAX = "TAX"
  ),
  force_sparse = "TAX",
  export_sets = FALSE,
```

```
  lowercase = FALSE
)

# Example 5: Apply custom dimension ordering
dim_order <- list(
  REG = c("ROW", "USA", "EU"),
  COLUMN = c("tot_E1", "alloc_A1")
)

save_har(
  data_list = list(WELF = WELF),
  file_path = file.path(tempdir(), "output_sorted.har"),
  dimensions = list(WELF = c("REG", "COLUMN")),
  value_cols = list(WELF = "Value"),
  long_desc = list(WELF = "Welfare Decomposition"),
  coefficients = list(WELF = "WELF"),
  export_sets = TRUE,
  dim_order = dim_order,
  lowercase = FALSE
)
```

---

shock_calculate *Calculate Shocks from Initial and Target Values*

Description

Computes compounded GEMPACK-style percentage shocks between initial and target values, producing multi-period shock series for dynamic simulation models. The function automatically aligns dimensions across datasets and exports results in HAR format.

Usage

```
shock_calculate(
  initial_config,
  target_config,
  calc_config,
  output_path,
  long_desc = "Calculated shock values",
  dim_order = NULL,
  lowercase = FALSE,
  new_baseline = FALSE
)
```

Arguments

`initial_config` A list created by `create_initial_config`, defining:

- Input path, file format, and variable header for the initial dataset
- Column name of the initial value field ("Value_ini")

`target_config` A list created by `create_target_config`, defining:

- Path, format, and variable header for the target dataset
- Target value column ("Value_tar") or numeric target for uniform shock

`calc_config` A list created by `create_calc_config`, specifying:

- `column_mapping`: mapping between initial and target columns
- `timeline`: sequence of years or periods (e.g., "1-5")
- `exclude_self_trade`: logical, whether to drop self-pairs
- `exclusion_values`: list of region/sector values to omit

`output_path` Path to the output HAR file where calculated shocks will be written.

`long_desc` Optional text for header description. Default is "Calculated shock values".

`dim_order` Optional dimension ordering specification. Can be:

- NULL (default): alphabetical A-Z ordering
- a named list defining order for each dimension (e.g., REG, COMM)
- a data frame or path to Excel/CSV file containing order definitions

`lowercase` Logical; if TRUE, converts dimension elements to lowercase. Default is FALSE.

`new_baseline` Logical; if TRUE, generates an additional HAR file with the new baseline rates (target values where available, initial values otherwise). The output file will have "_baseline" appended to the filename. Default is FALSE.

Details

- Computes percentage shocks using compounded "power of tax" formula
- Supports multiple periods defined via `timeline` configuration
- Compatible with HAR, SL4, CSV, or XLSX input formats
- Excludes self-trade or specified region-sector pairs when configured
- Exports results as multi-header HAR file (one header per timeline period)
- Optionally generates new baseline rate file showing post-adjustment values

Value

Invisibly returns a list containing summary metadata:

- `n_observations`: total records processed
- `n_included`: records included in shock computation
- `n_excluded`: records excluded by configuration
- `output_path`: normalized path to output HAR file
- `baseline_path`: normalized path to baseline file (if new_baseline = TRUE)

Author(s)

Pattawee Puangchit

See Also

shock_calculate_uniform, create_initial_config, create_target_config, create_calc_config, save_har

Examples

```
# Example 1: Target-Based Shock Calculation with New Baseline
har_path <- system.file("extdata", "baserate.har", package = "HARplus")

# Sorting Column
mapping <- list(
  REG = c("USA", "EU", "ROW")
)

# Initial File
initial <- create_initial_config(
  path   = har_path,
  format = "har",
  header = "rTMS"
)

# Target File
target <- create_target_config(
  path   = har_path,
  type   = "har",
  header = "rTMS"
)

# Calculation Setup with Column Mapping
calc <- create_calc_config(
  column_mapping      = c(TRAD_COMM = "TRAD_COMM", REG = "REG", REG.1 = "REG.1"),
  timeline            = "1-5",
  exclude_self_trade  = TRUE
)

# Compute Shock Based on Initial and Target Values
# Also generate new baseline file
shock_calculate(
  initial_config = initial,
  target_config  = target,
  calc_config    = calc,
  output_path    = file.path(tempdir(), "output_target.har"),
  dim_order      = mapping,
  new_baseline   = TRUE
)
```

---

shock_calculate_uniform

Calculate Shocks with Uniform Adjustment

Description

Generates GEMPACK-style percentage shocks using a uniform proportional adjustment applied to all values in the initial dataset. The function supports additive or multiplicative adjustments, single or multi-period configurations, and direct HAR export.

Usage

```
shock_calculate_uniform(
  initial_config,
  adjustment_value,
  calculation_method = "*",
  calc_config,
  output_path,
  long_desc = "Uniform shock adjustment",
  dim_order = NULL,
  lowercase = FALSE,
  new_baseline = FALSE
)
```

Arguments

`initial_config` A list created by `create_initial_config`, defining:

- Input file path, format, and variable header
- Column name for initial rate values ("Value_ini")

`adjustment_value`

Numeric scalar specifying the uniform adjustment to apply.

- Interpreted according to `calculation_method`
- For example, 0.5 with method "*" halves the base rate

`calculation_method`

Operator defining the adjustment method:

- "*" multiply (default)
- "/" divide
- "+" add
- "-" subtract

`calc_config` A list created by `create_calc_config`, specifying:

- `timeline`: sequence of simulation periods (e.g., "1-10")
- `exclude_self_trade`: logical, whether to omit intra-region pairs
- `exclusion_values`: optional list defining excluded elements

`output_path` Path to the output HAR file where calculated shocks will be written.

`long_desc` Optional text for header description. Default is "Uniform shock adjustment".

`dim_order` Optional dimension ordering specification. Can be:

- NULL (default): alphabetical A-Z ordering
- a named list defining preferred order per dimension
- a data frame or path to Excel/CSV with explicit order definitions

lowercase	Logical; if TRUE, converts all dimension elements to lowercase. Default is FALSE.
new_baseline	Logical; if TRUE, generates an additional HAR file with the new baseline rates (target values where available, initial values otherwise). The output file will have "_baseline" appended to the filename. Default is FALSE.

Details

- Applies uniform adjustment to all base rates across defined dimensions
- Supports additive ("+", "-") and proportional ("*", "/") adjustments
- Computes compounded shocks following the "power of tax" formulation
- Handles multiple time periods as defined by `timeline` in `calc_config`
- Excludes self-trade or specified region/sector pairs if configured
- Outputs results as multi-header HAR file (one per timeline period)
- Optionally generates new baseline rate file showing post-adjustment values

Value

Invisibly returns a list containing summary metadata:

- `n_observations`: total records processed
- `n_included`: records included in shock computation
- `n_excluded`: records excluded by configuration
- `output_path`: normalized path to the generated HAR file
- `baseline_path`: normalized path to baseline file (if new_baseline = TRUE)

Author(s)

Pattawee Puangchit

See Also

shock_calculate, create_initial_config, create_calc_config, save_har

Examples

```
# Example 1: Uniform Shock (50% Reduction) with New Baseline
har_path <- system.file("extdata", "baserate.har", package = "HARplus")

# Sorting Column
mapping <- list(
  REG = c("USA", "EU", "ROW")
)

# Initial File
initial <- create_initial_config(
  path   = har_path,
  format = "har",
```

```
  header = "rTMS"
)

# Calculation Setup
calc <- create_calc_config(
  timeline           = "1-10",
  exclude_self_trade = TRUE
)

# Compute Uniform 50% Reduction (Value_tar = Value_ini * 0.5)
# Also generate new baseline file showing the adjusted rates
shock_calculate_uniform(
  initial_config     = initial,
  adjustment_value   = 0.5,
  calculation_method = "*",
  calc_config        = calc,
  output_path        = file.path(tempdir(), "output_uniform.har"),
  dim_order          = mapping,
  new_baseline       = TRUE
)
```

Index