Package: harmonydata (via r-universe)

July 2, 2026

Type Package

Title R Library for 'Harmony'

Version 0.3.2

Description 'Harmony' is a tool using AI which allows you to compare items from questionnaires and identify similar content. You can try 'Harmony' at <https://harmonydata.ac.uk/app/> and you can read our blog at <https://harmonydata.ac.uk/blog/> or at <https://fastdatascience.com/how-does-harmony-work/>. Documentation at <https://harmonydata.ac.uk/harmony-r-released/>.

URL <https://harmonydata.ac.uk>

License MIT + file LICENSE

Encoding UTF-8

Imports httr, uuid, base64enc, jsonlite, utils, tools, assertthat, purrr

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Omar Hassoun [aut, cre], Thomas Wood [ctb], Alex, Nikic [ctb], Ulster University [cph]

Maintainer Omar Hassoun <omtarful@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-07-02 16:10:02 UTC

RemoteUrl https://github.com/cran/harmonydata

RemoteRef HEAD

RemoteSha 103c72aa44675f26e17e24f2a03fae048cbf4051

Contents

`create_instrument_from_list` *Create instrument from list*

Description

This function creates an instrument from a list of questions.

Usage

```
create_instrument_from_list(
  question_texts,
  question_numbers = NULL,
  instrument_name = "My instrument"
)
```

Arguments

`question_texts`	A character vector of question texts.
`question_numbers`	A character vector of question numbers. If not provided, the question number will be the index of the question text.
`instrument_name`	A character string of the instrument name.

Author(s)

Alex Nikic

Examples

```
instrument = create_instrument_from_list(
  list("How old are you?",
    "What is your gender?",
    "What is your name?")
)
```

`generate_crosswalk_table` *Generate Crosswalk Table Function*

Description

A crosswalk is a table that lists matched variables from different studies or instruments, enabling data harmonization across datasets.

Usage

```
generate_crosswalk_table(
  instruments,
  similarity,
  threshold,
  is_allow_within_instrument_matches = FALSE,
  is_enforce_one_to_one = FALSE
)
```

Arguments

`instruments`	The original list of instruments, each containing a question. The sum of the number of questions in all instruments is the total number of questions which should equal both the width and height of the similarity matrix.
`similarity`	The cosine similarity matrix that is outputed from the `match_instruments` function.
`threshold`	The minimum threshold that we consider a match. This is applied to the absolute match value. So if a question pair has similarity 0.2 and threshold = 0.5, then that question pair will be excluded. Leave as None if you don't want to apply any thresholding.
`is_allow_within_instrument_matches`	Defaults to False. If this is set to True, we include crosswalk items that originate from the same instrument, which would otherwise be excluded by default.
`is_enforce_one_to_one`	Defaults to False. If this is set to True, we force all variables in the crosswalk table to be matched with exactly one other variable.

Details

This function generates a crosswalk table using a list of instruments and a similarity matrix, produced by the `match_instruments` function.

A crosswalk is a mapping between conceptually similar items (e.g., survey questions or variables) from different instruments. It is used to identify and align comparable variables across datasets that use different formats or wordings. This is especially useful in meta-analysis, data integration, and comparative research, where consistent constructs need to be analyzed across multiple sources.

The similarity matrix passed to this function is usually obtained from `match_instruments`.

Value

A crosswalk table as a DataFrame.

Author(s)

Alex Nikic

Omar Hassoun

Examples

```
instrument_A = create_instrument_from_list(list(
  "How old are you?",
  "What is your gender?"
))

instrument_B = create_instrument_from_list(list(
  "Do you smoke?"
))

instruments = list(instrument_A, instrument_B)

match_response = match_instruments(instruments)
instrument_list = match_response$instruments
similarity_matrix = match_response$matches

crosswalk_table.df = generate_crosswalk_table(
  instrument_list, similarity_matrix, threshold = 0.7,
  is_allow_within_instrument_matches = FALSE, is_enforce_one_to_one = TRUE
)
```

---

get_example_instruments

Retrieve Example Instruments from 'Harmony Data API'

Description

This function retrieves example instruments from the 'Harmony Data API' using an HTTP POST request.

Usage

```
get_example_instruments()
```

Value

A list representing example instruments retrieved from the 'Harmony Data API'.

Author(s)

Ulster University [cph]

Examples

```
# Load required libraries (httr) and call the function
require(httr)
instruments <- get_example_instruments()

# Print the retrieved JSON content
print(instruments)
```

---

load_instruments_from_file

Load Instruments from File

Description

This function loads instruments from a file specified by the path parameter and sends the file content to an API for further processing. It also accepts a URL leading to a file.

Usage

```
load_instruments_from_file(path)
```

Arguments

path	The path to the file to load instruments from.

Value

A list of instruments returned from the API.

Author(s)

Ulster University [cph]

Examples

```
# Load instruments from a PDF file
pdf_file <- "https://www.apa.org/depression-guideline/patient-health-questionnaire.pdf"
response <- load_instruments_from_file(pdf_file)
```

`match_instruments` *Match Instruments Function*

Description

This function takes a list of instruments, converts it to a format acceptable by the database, and matches the instruments using the 'Harmony Data API'. It returns the matched instruments.

Usage

```
match_instruments(
  instruments,
  topics = list(),
  is_negate = TRUE,
  clustering_algorithm = "affinity_propagation",
  ...
)
```

Arguments

`instruments`	A list of instruments to be matched.
`topics`	A list of topics with which to tag the questions. Default is empty.
`is_negate`	A boolean indicating whether to apply negation-based preprocessing. Default is TRUE. This option addresses a common limitation in large language model (LLM) embeddings, where antonyms (e.g., "happy" and "sad") may be treated as similar due to contextual overlap. When `is_negate = TRUE`, the function prepends negation terms such as "not" or "didn't" to the input sentences and evaluates whether this increases or decreases their cosine similarity. If the similarity increases after negation, the model interprets the sentences as antonyms and returns a negative similarity score. When `is_negate = FALSE`, negation is skipped and most similarity values returned will be positive. The Harmony API defaults to `is_negate = TRUE`, as some users prefer detecting antonymy through negative similarity values, while others may prefer only positive scores.'
`clustering_algorithm`	A string value to select the clustering algorithm to use. Must be one of: "affinity_propagation", "kmeans", "deterministic", "hdbscan". Default is "affinity_propagation".
`...`	Optional named arguments: **model** The HuggingFace model to be used by the matcher. Currently recommended models: • `sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2` • `sentence-transformers/paraphrase-multilingual-mpnet-base-v2` • `harmonydata/mental_health_harmonisation_1`

Value

A list containing the matched instruments retrieved from the Harmony Data API. The returned object includes attributes such as the similarity matrix, identified clusters, associated cluster topics, and other relevant metadata.

Author(s)

Ulster University [cph]

Examples

```
instrument_A <- create_instrument_from_list(list(
  "How old are you?",
  "What is your gender?"
))

instrument_B <- create_instrument_from_list(list(
  "Do you smoke?"
))

instruments <- list(instrument_A, instrument_B)

matched_instruments <- match_instruments(
  instruments,
  topics = list("anxiety", "depression")
)

# with optional arguments
matched_instruments <- match_instruments(
  instruments,
  topics = list("anxiety", "depression"),
  is_negate = TRUE,
  clustering_algorithm = "affinity_propagation",
  model = "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2"
)
```

Index