Package 'prefio'

September 28, 2025

Title Structures for Preference Data

Description Convenient structures for creating, sourcing, reading, writing and manipulating ordinal preference data. Methods for writing to/from PrefLib formats. See Nicholas Mattei and Toby Walsh ``PrefLib: A Library of Preference Data" (2013) <doi:10.1007/978-3-642-41575-3_20>.

Version 0.2.0

Depends R (>= 4.2)

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/fleverest/prefio/, https://fleverest.github.io/prefio/

BugReports https://github.com/fleverest/prefio/issues/

Imports dplyr, tidyr, tibble, vctrs, stats, rlang, purrr, utils

Suggests readr, covr, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Floyd Everest [aut, cre] (ORCID: <https://orcid.org/0000-0002-2726-6736>), Heather Turner [aut] (ORCID: <https://orcid.org/0000-0002-1256-3375>), Damjan Vukcevic [aut] (ORCID: <https://orcid.org/0000-0001-7780-9586>)

Maintainer Floyd Everest <me@floydeverest.com>

Repository CRAN

Date/Publication 2025-09-28 13:20:07 UTC

Contents

`adjacency` *Compute the Adjacency Matrix for a vector of preferences*

Description

Convert a set of preferences to an adjacency matrix summarising wins and losses between pairs of items.

Usage

```
adjacency(x, preferences_col = NULL, frequency_col = NULL, ...)
```

Arguments

`x`	A `preferences` object or a `tibble` with a `preferences`-typed column.
`preferences_col`	<`tidy-select`> When `x` is a `tibble`, the column containing the preferences.
`frequency_col`	<`tidy-select`> When `x` is a `tibble`, the column containing the frequency of the preferences. If not provided, each row is considered to be observed a single time.
`...`	Currently unused.

Details

For a `preferences` object with N items, the adjacency matrix is an N by N matrix, with element (i, j) being the number of times item i wins over item j. For example, in the preferences $\{1\} > \{3, 4\} > \{2\}$, item 1 wins over items 2, 3, and 4, while items 3 and 4 win over item 2.

If `weights` is specified, the values in the adjacency matrix are the weighted counts.

Value

An N by N matrix, where N is the number of items.

Examples

```
x <- tibble::tribble(
  ~voter_id, ~species, ~food, ~ranking,
  1, "Rabbit", "Apple", 1,
  1, "Rabbit", "Banana", 2,
  1, "Rabbit", "Carrot", 3,
  2, "Monkey", "Banana", 1,
  2, "Monkey", "Apple", 2,
  2, "Monkey", "Carrot", 3
) |>
  long_preferences(
    food_preference,
    id_cols = voter_id,
    item_col = food,
    rank_col = ranking
  ) |>
  dplyr::pull(food_preference) |>
  adjacency()
```

---

preferences *Preferences Objects*

Description

A tidy interface for working with ordinal preferences.

Usage

```
long_preferences(
  data,
  col,
  id_cols = NULL,
  rank_col = NULL,
  item_col = NULL,
  item_names = NULL,
  verbose = TRUE,
  unused_fn = NULL,
  na_action = c("drop_rows", "drop_preferences"),
  ...
)

wide_preferences(
  data,
```

```
  col = NULL,
  ranking_cols = NULL,
  verbose = TRUE,
  na_action = c("keep_as_partial", "drop_preferences"),
  ...
)

as_preferences(strings, sep = ">", equality = "=", descending = TRUE)

preferences(
  strings = character(0L),
  sep = ">",
  equality = "=",
  descending = TRUE
)

## S3 method for class 'preferences'
format(x, ...)

## S3 method for class 'preferences'
levels(x, ...)
```

Arguments

data	A `data.frame` or `tibble` to extract preferences from
col	The name of the new column, as a string or symbol.
id_cols	<tidy-select> The columns by which to group the dataset to extract a single preference selection.
rank_col	<tidy-select> For `data` in long-format: the column representing the rank for the associated item.
item_col	<tidy-select> For `data` in long-format: the column representing the items by name or by index, in which case the `item_names` parameter should also be passed.
item_names	The names of the full set of items. This is necessary when the dataset specifies items by index rather than by name, or when there are items which do not appear in any preference selection.
verbose	If `TRUE`, diagnostic messages will be sent to stdout.
unused_fn	When `format="long"`, summarise the values of unused columns (those which are not specified by `id_cols`, `item_col`, or `rank_col`). The default action is to drop all unused columns. This can be a named list (e.g. `list(column = function)`) if you want to apply different summaries for different columns or keep only specific unused columns, or it can be a single function to be applied across all unused columns.
na_action	Specifies how to handle NA values.
	`long_preferences` `"drop_rows"` Removes individual rows containing NA values before processing

	`"drop_preferences"` Removes the entire preference selection that contains any NA
`wide_preferences`	`"keep"` Interprets rows containing NAs as partial orderings
	`"drop"` Removes preferences with any NA ranks
`...`	Unused.
`ranking_cols`	<tidy-select> The columns from which to extract wide-format preferences.
`strings`	A character vector of preference strings
`sep`	Character separating the items in the string (default: ">")
`equality`	Character representing equality between items (default: "=")
`descending`	If TRUE, parse as descending order preferences.
`x`	A vector of preferences.
`format`	The format of the data: one of "ordering", "ranking", or "long" (see above). By default, `data` is assumed to be in "long" format.

Value

A `preferences` object, or a modified tibble with a column of `preferences` when `data` is a `data.frame` or `tibble`.

Examples

```
# Votes cast by two animals ranking a variety of fruits and vegetables.
# This is not real data, I made this up.
x <- tibble::tribble(
  ~voter_id, ~species, ~food, ~ranking,
  1, "Rabbit", "Apple", 1,
  1, "Rabbit", "Carrot", 2,
  1, "Rabbit", "Banana", 3,
  2, "Monkey", "Banana", 1,
  2, "Monkey", "Apple", 2,
  2, "Monkey", "Carrot", 3
)
# Process preferencial data into a single column.
x |>
  long_preferences(
    food_preference,
    id_cols = voter_id,
    item_col = food,
    rank_col = ranking
  )
# The same, but keep the species data.
x |>
  long_preferences(
    food_preference,
    id_cols = voter_id,
    item_col = food,
    rank_col = ranking,
```

```
  unused_fn = list(species = dplyr::first)
)
```

`pref_add_unranked` *Complete preferences by adding unselected items as last place occurrances.*

Description

Complete preferences by adding unselected items as last place occurrances.

Usage

```
pref_add_unranked(x)
```

Arguments

x	A vector of preferences.

Value

A new vector of preferences, with each selection starting with the corresponding selections made in x, but with all unranked items placed last.

Examples

```
# Complete partial rankings by adding unranked items last
pref_add_unranked(preferences(c("a > b", "c > a", "b")))
```

`pref_blank` *Check if a preference is blank.*

Description

Check if a preference is blank.

Usage

```
pref_blank(x)
```

Arguments

x	A vector of preferences.

Value

A logical vector indicating which preferences are blank, i.e., [].

Examples

```
pref_blank(preferences(c("a > b > c", "", "b > c")))
```

`pref_cov` *Covariance matrix for preferences, calculated using the rankings matrix.*

Description

Covariance matrix for preferences, calculated using the rankings matrix.

Usage

```
pref_cov(x, preferences_col = NULL, frequency_col = NULL, ...)
```

Arguments

`x`	A vector of preferences, or a tibble with a column of preferences.
`preferences_col`	<tidy-select> When `x` is a `tibble`, the column containing the preferences.
`frequency_col`	<tidy-select> When `x` is a `tibble`, the column containing the frequency of the preferences. If not provided, each row is considered to be observed a single time.
`...`	Extra arguments to be passed to `stats::cov.wt`.

Value

A covariance matrix containing covariances for the ranks assigned to item pairs.

Examples

```
# Simple covariance on a vector of preferences
prefs <- preferences(c("a > b > c", "b > c > a", "c > a > b"))
pref_cov(prefs)

# Weighted covariance by frequency
df <- tibble::tibble(
  prefs = preferences(c("a > b > c", "b > c > a")),
  freq = c(3, 2)
)
pref_cov(df, preferences_col = prefs, frequency_col = freq)
```

pref_get_items *Get the name of the item(s) assigned a specific rank, e.g., first.*

Description

Get the name of the item(s) assigned a specific rank, e.g., first.

Usage

```
pref_get_items(x, rank, drop = FALSE)
```

Arguments

x	A vector of preferences.
rank	A single integer, the rank which you would like to inspect.
drop	When FALSE (default), blank preferences will remain. When TRUE, blank preferences will be omitted.

Value

A list containing the name(s) of the item(s) ranked rank in each of the preferences in x.

Examples

```
# Get items ranked first
pref_get_items(preferences(c("a > b > c", "b = c > a")), rank = 1)
# Get items ranked second
pref_get_items(preferences(c("a > b > c", "b = c > a")), rank = 2)
# Get items ranked first, dropping blank preferences
pref_get_items(preferences(c("a > b > c", "", "b = c > a")), rank = 1, drop = TRUE)
```

pref_get_rank *Get the rank assigned to a specific item in a set of preferences.*

Description

Get the rank assigned to a specific item in a set of preferences.

Usage

```
pref_get_rank(x, item_name)
```

Arguments

x	A vector of preferences.
item_name	The name of the item to extract the rank for.

Value

The rank of `item_name` for each of the preferences in `x`.

Examples

```
pref_get_rank(preferences(c("a > b > c", "b > c = a", "")), "a")
```

`pref_irv` *Compute the instant-runoff voting winner for a set of preferences.*

Description

A very rudimentary implementation of the IRV counting algorithm. It does not handle ties elegantly, and should only be used for demonstration purposes. This implementation eliminates all candidates with the fewest first-choice votes in each round until one candidate has a majority or fewer than two candidates remain.

Usage

```
pref_irv(x, preferences_col = NULL, frequency_col = NULL)
```

Arguments

`x`	A vector of preferences, or a tibble with a column of preferences.
`preferences_col`	<tidy-select> When `x` is a `tibble`, the column containing the preferences.
`frequency_col`	<tidy-select> When `x` is a `tibble`, the column containing the frequency of the preferences. If not provided, each row is considered to be observed a single time.

Value

A list containing:

winner The winning candidate(s) after IRV counting

rounds A list of tibbles, each containing vote tallies for each round

eliminated Character vector of eliminated candidates in order

Examples

```
# Multi-round election with four candidates
prefs <- preferences(c(
  "alice > bob > charlie > david",
  "alice > bob > charlie > david",
  "alice > charlie > bob > david",
  "bob > alice > charlie > david",
  "bob > charlie > alice > david",
```

```
  "bob > charlie > alice > david",
  "charlie > david > alice > bob",
  "charlie > david > bob > alice",
  "david > charlie > bob > alice",
  "david > charlie > bob > alice"
))
result <- pref_irv(prefs)
result$winner # Final winner after elimination rounds
result$rounds # Vote tallies for each round

# Using aggregated data frame
df <- tibble::tibble(
  prefs = preferences(c(
    "alice > bob > charlie > david",
    "alice > charlie > bob > david",
    "bob > alice > charlie > david",
    "bob > charlie > alice > david",
    "charlie > david > alice > bob",
    "charlie > david > bob > alice",
    "david > charlie > bob > alice"
  )),
  freq = c(2, 1, 1, 2, 1, 1, 2)
)
pref_irv(df, prefs, freq)
```

`pref_keep` *Keep only specified items from preferences.*

Description

Keep only specified items from preferences.

Usage

```
pref_keep(x, items)
```

Arguments

`x`	A vector of `preferences`.
`items`	The names of the items which should be kept for preferences in `x`.

Value

A new vector of preferences, but only containing `items` from each selection.

Examples

```
# Keep only 'a' and 'c'
pref_keep(preferences(c("a > b > c", "b > c > a")), c("a", "c"))
```

pref_length *Check the length (number of rankings) of a preference.*

Description

Check the length (number of rankings) of a preference.

Usage

```
pref_length(x)
```

Arguments

x	A vector of preferences.

Value

The number of items listed on each of the preferences.

Examples

```
pref_length(preferences(c("a > b > c", "", "b > c")))
```

pref_omit *Remove specified items from preferences.*

Description

Remove specified items from preferences.

Usage

```
pref_omit(x, items)
```

Arguments

x	A vector of preferences.
items	The names of the items which should be removed from the preferences in x.

Value

A new vector of preferences, but with items removed from each selection.

Examples

```
# Remove 'b'
pref_omit(preferences(c("a > b > c", "b > c > a")), "b")
# Remove 'b' and 'd'
pref_omit(preferences(c("a > b > c > d", "b > c > a > d")), c("b", "d"))
```

pref_pop *Eliminate lowest (or highest) ranked items from preferences.*

Description

Eliminate lowest (or highest) ranked items from preferences.

Usage

```
pref_pop(x, n = 1L, lowest = TRUE, drop = FALSE)
```

Arguments

x	A vector of preferences.
n	The number of times to remove the bottom rank.
lowest	If TRUE, eliminates the lowest ranked item(s) for each selection.
drop	If TRUE, drops blank preferences from the output.

Value

A new vector of preferences which is equal to x but with the least preferred selection dropped for each selection.

Examples

```
# Remove the lowest ranked item from each preference
pref_pop(preferences(c("a > b > c", "b > c > a")))

# Remove the 2 lowest ranked items
pref_pop(preferences(c("a > b > c > d", "b > c > a > d")), n = 2)

# Remove the highest ranked item instead
pref_pop(preferences(c("a > b > c", "b > c > a")), lowest = FALSE)

# Remove blank preferences that result from popping
pref_pop(preferences(c("a > b", "c", "")), drop = TRUE)
```

pref_rev *Reverse preference rankings*

Description

Reverse preference rankings

Usage

```
pref_rev(x, ...)
```

Arguments

x	A vector of preferences.
...	Not used.

Value

A vector of preferences with rankings reversed (first becomes last, etc.)

Examples

```
pref_rev(preferences(c("a > b > c", "b > c > a")))
```

---

pref_trunc *Truncate preferences to a maximum number of ranks.*

Description

Truncate preferences to a maximum number of ranks.

Usage

```
pref_trunc(x, n = 1L, bottom = FALSE)
```

Arguments

x	A vector of preferences.
n	The maximum number of ranks to include (positive) or number of ranks to drop (negative). Must be an integer.
bottom	If FALSE (default), operates on top ranks. If TRUE, operates on bottom ranks.

Value

A vector of preferences with each selection truncated according to the parameters.

Examples

```
# Keep only the top 2 ranks
pref_trunc(preferences(c("a > b > c > d", "b > c > a")), n = 2)
# Keep only the bottom 2 ranks
pref_trunc(preferences(c("a > b > c > d", "b > c > a")), n = 2, bottom = TRUE)
# Drop the bottom 2 ranks (keep top ranks)
pref_trunc(preferences(c("a > b > c > d", "b > c > a")), n = -2)
# Drop the top 2 ranks (keep bottom ranks)
pref_trunc(preferences(c("a > b > c > d", "b > c > a")), n = -2, bottom = TRUE)
```

pref_type *pref_type*

Description

Ordinal preferences can order every item, or they can order a subset. Some ordinal preference datasets will contain ties between items at a given rank. Hence, there are four distinct types of preferential data:

`soc` Strict Orders - Complete List

`soi` Strict Orders - Incomplete List

`toc` Orders with Ties - Complete List

`toi` Orders with Ties - Incomplete List

Usage

```
pref_type(x, n_items = NULL)
```

Arguments

`x`	A `preferences` object (or vector data representing preferences)
`n_items`	The number of items, needed to assess whether a selection is complete or not. Defaults to `nlevels(x)` if `x` has class `preferences`, otherwise defaults to the length of the longest preference.

Value

One of `c("soc", "soi", "toc", "toi")`, indicating the type of preferences in `x` (with or without ties / complete or incomplete rankings).

ranking_matrix *Compute the Rankings Matrix for a vector of preferences*

Description

Convert a set of preferences to a rankings matrix, where each preference defines a single row in the output. The columns in the rankings matrix give the vector or ranks assigned to the corresponding candidate.

Usage

```
ranking_matrix(x, preferences_col = NULL, frequency_col = NULL, ...)
```

Arguments

x	A preferences object or a tibble with a preferences-typed column.
preferences_col	<tidy-select> When x is a tibble, the column containing the preferences.
frequency_col	<tidy-select> When x is a tibble, the column containing the frequency of the preferences. If not provided, each row is considered to be observed a single time.
...	Currently unused.

Details

For a preferences vector of length N with M items, the rankings matrix is an N by M matrix, with element (i, j) being the rank assigned to candidate j in the ith selection.

Value

An N by M matrix, where N is the number of preferences, and M is the number of items.

Examples

```
x <- tibble::tribble(
  ~voter_id, ~species, ~food, ~ranking,
  1, "Rabbit", "Apple", 1,
  1, "Rabbit", "Banana", 2,
  1, "Rabbit", "Carrot", 3,
  2, "Monkey", "Banana", 1,
  2, "Monkey", "Apple", 2,
  2, "Monkey", "Carrot", 3
) |>
  long_preferences(
    food_preference,
    id_cols = voter_id,
    item_col = food,
    rank_col = ranking
  ) |>
  dplyr::pull(food_preference) |>
  ranking_matrix()
```

---

read_preflib *Read Ordinal Preference Data From PrefLib*

Description

Read orderings from .soc, .soi, .toc or .toi files storing ordinal preference data format as defined by {PrefLib}: A Library for Preferences into a preferences object.

Usage

```
read_preflib(
  file,
  from_preflib = FALSE,
 preflib_url = "https://raw.githubusercontent.com/PrefLib/PrefLib-Data/main/datasets/"
)
```

Arguments

`file`	A preferential data file, conventionally with extension `.soc`, `.soi`, `.toc` or `.toi` according to data type.
`from_preflib`	A logical which, when `TRUE` will attempt to source the file from PrefLib by adding the database `HTTP` prefix.
`preflib_url`	The URL which will be preprended to `file`, if `from_preflib` is `TRUE`.

Details

Note that PrefLib refers to the items being ordered by "alternatives".

The file types supported are

.soc Strict Orders - Complete List

.soi Strict Orders - Incomplete List

.toc Orders with Ties - Complete List

.toi Orders with Ties - Incomplete List

The numerically coded orderings and their frequencies are read into a tibble, storing all original metadata in a "preflib" attribute.

A PrefLib file may be corrupt, in the sense that the ordered alternatives do not match their names. In this case, the file will still be read, but with a warning.

Value

A tibble with two columns: `preferences` and `frequency`. The `preferences` column contains all the preferential orderings in the file, and the `frequency` column the relative frequency of this selection.

Note

The Netflix and cities datasets used in the examples are from Caragiannis et al (2017) and Bennet and Lanning (2007) respectively. These data sets require a citation for re-use.

References

Mattei, N. and Walsh, T. (2013) PrefLib: A Library of Preference Data. *Proceedings of Third International Conference on Algorithmic Decision Theory (ADT 2013)*. Lecture Notes in Artificial Intelligence, Springer.

Bennett, J. and Lanning, S. (2007) The Netflix Prize. *Proceedings of The KDD Cup and Workshops*.

Examples

```
# Can take a little while depending on speed of internet connection


# strict complete orderings of four films on Netflix
netflix <- read_preflib("00004 - netflix/00004-00000138.soc", from_preflib = TRUE)
head(netflix)
levels(netflix$preferences)

# strict incomplete orderings of 6 random cities from 36 in total
cities <- read_preflib("00034 - cities/00034-00000001.soi", from_preflib = TRUE)
```

`write_preflib` *Write Ordinal Preference Data to PrefLib Formats*

Description

Write `preferences` to `.soc`, `.soi`, `.toc` or `.toi` file types, as defined by the PrefLib specification: {PrefLib}: A Library for Preferences.

Usage

```
write_preflib(
  x,
  file = "",
  preferences_col = NULL,
  frequency_col = NULL,
  title = NULL,
  publication_date = NULL,
  modification_type = NULL,
  modification_date = NULL,
  description = NULL,
  relates_to = NULL,
  related_files = NULL
)
```

Arguments

`x`	A `preferences` object or a `tibble` with a `preferences`-typed column to write to file.
`file`	Either a character string naming the a file or a writeable, open connection. The empty string `""` will write to stdout.
`preferences_col`	<tidy-select> When `x` is a `tibble`, the column containing the preferences to be written to file. If not provided and `x` is a `tibble`, then

`frequency_col` <tidy-select> When `x` is a `tibble`, the column containing the frequency of the preferences. If not provided, each row is considered to be observed a single time.

`title` The title of the data file, for instance the name of the election represented in the data file. If not provided, we check for the presence of `attr(x, "preflib")`, and if it exists we check for `TITLE`.

`publication_date` The date at which the data file was published for the first time. If not provided, we check for the presence of `attr(x, "preflib")`, and if it exists we check for `PUBLICATION DATE`.

`modification_type` The modification type of the data: one of `original`, `induced`, `imbued` or `synthetic` (see `Details`). If not provided, we check for the presence of `attr(x, "preflib")`, and if it exists we check for `MODIFICATION TYPE`.

`modification_date` The last time the data was modified. If not provided, we check for the presence of `attr(x, "preflib")`, and if it exists we check for `MODIFICATION DATE`.

`description` A description of the data file, providing additional information about it. If not provided, we check for the presence of `attr(x, "preflib")`, and if it exists we check for `DESCRIPTION`.

`relates_to` The name of the data file that the current file relates to, typically the source file in case the current file has been derived from another one. If not provided, we check for the presence of `attr(x, "preflib")`, and if it exists we check for `RELATES TO`.

`related_files` The list of all the data files related to this one, comma separated. If not provided, we check for the presence of `attr(x, "preflib")`, and if it exists we check for `RELATED FILES`.

Details

The file types supported are

.soc Strict Orders - Complete List

.soi Strict Orders - Incomplete List

.toc Orders with Ties - Complete List

.toi Orders with Ties - Incomplete List

The PrefLib format specification requires some additional metadata. Note that the additional metadata required for the PrefLib specification is not necessarily required for the `write_preflib` method; any missing fields required by the PrefLib format will simply show "NA".

TITLE (required) The title of the data file, for instance the year of the election represented in the data file.

DESCRIPTION (optional) A description of the data file, providing additional information about it.

RELATES TO (optional) The name of the data file that the current file relates to, typically the source file in case the current file has been derived from another one.

RELATED FILES (optional) The list of all the data files related to this one, comma separated.

PUBLICATION DATE (required) The date at which the data file was published for the first time.

MODIFICATION TYPE (required) The modification type of the data. One of:

- **original** Data that has only been converted into a PrefLib format.
- **induced** Data that has been induced from another context. For example, computing a pairwise relation from a set of strict total orders. No assumptions have been made to create these files, just a change in the expression language.
- **imbued** Data that has been imbued with extra information. For example, extending an incomplete partial order by placing all unranked candidates tied at the end.
- **synthetic** Data that has been generated artificially.

MODIFICATION DATE (optional) The last time the data was modified.

In addition to these fields, some required PrefLib fields will be generated automatically depending on arguments to `write_preflib()` and the attributes of the `aggregated_preferences` object being written to file:

FILE NAME The name of the output file.

DATA TYPE The data type (one of `soc`, `soi`, `toc` or `toi`).

NUMBER ALTERNATIVES The number of items.

ALTERNATIVE NAME `X` The name of each item, where `X` ranges from `0` to `length(items)`.

NUMBER VOTERS The total number of orderings.

NUMBER UNIQUE ORDERS The number of distinct orderings.

Note that PrefLib refers to the items as "alternatives". The "alternatives" in the output file will be the same as the "items" in the `aggregated_preferences` object.

Value

No return value. Output will be written to file or stdout.

Index