Package 'rankingQ'

August 21, 2026

Type Package

Title Design-Based Methods for Ranking Questions

Version 0.2.0

Description Provides a design-based toolkit for survey ranking questions. Estimates average ranks, marginal rank probabilities, pairwise comparisons, and ranking distributions, with optional bias correction for random responding via anchor-ranking items or user-supplied random-response rates. Includes Plackett-Luce simulation, visualization, format conversion, and diagnostic checks. Methods are described in Atsusaka and Kim (2025) <doi:10.1017/pan.2024.33>.

URL https://github.com/sysilviakim/rankingQ, https://sysilviakim.com/rankingQ/

BugReports https://github.com/sysilviakim/rankingQ/issues

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports dplyr, tidyr (>= 1.3.0), tidyselect, purrr, tibble, generics, ggplot2, rlang, combinat, estimatr, stats, Rcpp

LinkingTo Rcpp

Suggests knitr, rmarkdown, cli, testthat (>= 3.0.0)

Depends R (>= 4.1.0)

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Seo-young Silvia Kim [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-8801-9210>), Yuki Atsusaka [aut, cph] (ORCID: <https://orcid.org/0000-0001-5365-1876>)

Maintainer Seo-young Silvia Kim <sy.silvia.kim@gmail.com>

Repository CRAN

Date/Publication 2026-08-21 13:50:14 UTC

Contents

`add_ipw_weights` *Add IPW Weights to the Original Data*

Description

This function is a thin convenience wrapper around `imprr_weights()` for users who want respondent-level inverse probability weights attached to the original data and do not necessarily need the full ranking-profile output.

Usage

```
add_ipw_weights(
  data,
  J = NULL,
  main_q,
  anc_correct = NULL,
  population = "non-random",
  assumption = "contaminated",
  weight = NULL,
  weight_col = "ipw_weights",
  keep_ranking = FALSE,
  ranking_col = "ranking",
  keep_rankings = FALSE,
  p_random = NULL
)
```

Arguments

data	The input dataset with ranking data.
J	The number of items in the ranking question. Defaults to NULL, in which case it will be inferred from the data.
main_q	Ranking question to be analyzed. When main_q is a single column name or unquoted symbol such as my_ranking, the function looks for my_ranking_1, my_ranking_2, my_ranking_3, and so on. You may also supply main_q directly as a character vector or unquoted c(...) expression of ranking columns such as c(party, religion, gender, race).
anc_correct	Optional indicator for passing the anchor question. If NULL, p_random is used when supplied; otherwise the function defaults to p_random = 0 and applies no correction.
population	Choice of the target population out of non-random respondents (default) or all respondents.
assumption	Choice of identifying assumption when population = "all": uniform assumes random respondents would have uniform counterfactual preferences, while contaminated assumes their counterfactual preferences match those of non-random respondents.
weight	Optional weight specification for the estimation step. This can be the name of a weight column in data, a numeric vector with one weight per row, or an unquoted column name. Defaults to NULL, which uses equal weights.
weight_col	Name of the respondent-level IPW weight column to add to the returned data. Defaults to "ipw_weights".
keep_ranking	Logical; if TRUE, keep the unified ranking-profile column in the returned augmented data. Defaults to FALSE.
ranking_col	Name of the unified ranking-profile column used in the augmented data and ranking summary. Defaults to "ranking".
keep_rankings	Logical; if TRUE, also return the permutation- level ranking summary and estimated random-response rate. Defaults to FALSE.

p_random Optional fixed proportion of random/inattentive respondents. When supplied, this overrides anc_correct and a message is shown if both are provided.

Value

If keep_rankings = FALSE, a data frame equal to the original data augmented with weight_col. If keep_rankings = TRUE, a list with three elements:

data The augmented original data with respondent-level IPW weights.

rankings The permutation-level ranking summary returned by imprr_weights().

est_p_random The estimated proportion of random responses.

Examples

```
dat_w <- add_ipw_weights(
  identity,
  main_q = c("party", "religion", "gender", "race"),
  anc_correct = "anc_correct_identity"
)
head(dat_w)

out <- add_ipw_weights(
  identity,
  main_q = c("party", "religion", "gender", "race"),
  anc_correct = "anc_correct_identity",
  keep_rankings = TRUE
)
head(out$data)
head(out$rankings)
```

---

avg_rank *Compute the Average Rank of All Items*

Description

This function calculates the average rank for the data frame that contains ranking data. It can be used for both long- and wide-type data frames.

Usage

```
avg_rank(
  x,
  rankings = NULL,
  items = NULL,
  long = FALSE,
  raw = TRUE,
  weight = NULL,
  round = NULL
)
```

Arguments

x	A data frame that contains rankings of items.
rankings	The name of the column that contains the rankings. Defaults to NULL, which means that the function will look for a data frame with two columns, "item" and "rank". For wide data, this can also be a character vector of separate ranking columns such as `c("apple", "orange", "banana")`.
items	The name of the column that contains the items' names, or, in case of a wide file, the item names in the reference choice set. Defaults to NULL.
long	The type of the data frame. Defaults to 'FALSE'. It 'TRUE', which means that the data frame is in the long format, it is presumed to be generated by `rank_longer()`. If the data frame is in the wide format, it should be set to 'FALSE'. For wide data, rankings can be supplied either as a single encoded ranking string such as `"123"` or as separate ranking columns.
raw	If `TRUE`, the function will return the raw average rank. If `FALSE`, the function will return the average rank after correcting based on the IPW estimator. Defaults to `TRUE`.
weight	Optional weight specification. This can be the name of a weight column in `x` or a numeric vector with one weight per row. Defaults to `NULL`.
round	The number of decimal places to round the output to. Defaults to `NULL`.

Value

A data frame with the average rank of each item in the reference choice set.

Examples

```
x <- data.frame(
  id = c("Bernie", "Yuki", "Silvia"),
  rank = c("123", "321", "213")
)
avg_rank(x, "rank")
avg_rank(x, "rank", items = c("Money", "Power", "Respect"))

y <- data.frame(rank = c("123", "321", "213"))
avg_rank(y, "rank")

x_sep <- data.frame(
  apple = c(2, 1, 3),
  orange = c(1, 3, 2),
  banana = c(3, 2, 1)
)
avg_rank(x_sep, rankings = c("apple", "orange", "banana"))

x_weighted <- data.frame(
  rank = c("12", "21"),
  survey_weight = c(1, 3)
)
avg_rank(x_weighted, "rank", weight = "survey_weight")
```

```
z <- rank_longer(
  y,
  cols = "rank",
  reference = c("Money", "Power", "Respect")
)
avg_rank(z, "ranking", items = "item_name", long = TRUE)

## Example output from item_to_rank
x <- data.frame(
  item = c("a", "b", "c", "a", "b", "c", "a", "b", "c"),
  rank = c(3L, 1L, 2L, 1L, 2L, 3L, 3L, 2L, 1L)
)
avg_rank(x, long = TRUE)
```

`identity` *Identity-ranking data analyzed in Atsusaka and Kim (2025)*

Description

Full dataset from survey on relative partisanship used in Atsusaka, Yuki, & Kim, Seo-young Silvia (2025). Addressing Measurement Errors in Ranking Questions for the Social Sciences. Political Analysis, 33(4), 339-360. https://doi.org/10.1017/pan.2024.33

This data contains Americans' rankings of four sources of their identity, including political party, religion, gender, and race, for 1,082 respondents. The columns consist of marginal rankings to the main identity ranking question and the corresponding anchor question, as well as whether they have answered the anchor questions "correctly." The anchor ranking question is used to estimate the proportion of random responses and to correct for measurement error bias.

Usage

```
identity
```

Format

'identity' A data frame with 1,082 rows and 16 columns:

app_identity Full ranking profile for the main identity ranking question.

party Marginal ranking for party (main identity ranking question).

religion Marginal ranking for religion (main identity ranking question).

gender Marginal ranking for gender (main identity ranking question).

race Marginal ranking for race (main identity ranking question).

anc_identity Full ranking profile for the anchor ranking question.

household Marginal ranking for household (anchor question).

neighborhood Marginal ranking for neighborhood (anchor question).

city Marginal ranking for city (anchor question).

state Marginal ranking for state (anchor question).

anc_correct_identity Whether the respondent answered the anchor questions correctly. This is a binary variable that 1 if the respondent correctly answers the anchor ranking question and 0 if otherwise.

app_identity_recorded Recorded responses for the main identity ranking question.

anc_identity_recorded Recorded responses for the anchor ranking question.

app_identity_row_rnd The order in which the items were randomly presented for the respondent in the main ranking question.

anc_identity_row_rnd The order in which the items were randomly presented for the respondent in the anchor ranking question.

s_weight Survey weight.

Source

<https://github.com/sysilviakim/ranking_error>

`identity_w` *Identity-ranking data with estimated weights based on inverse probability weighting*

Description

This data is the ‘results‘ element returned by applying ‘imprr_weights()‘ to ‘identity‘. It adds two columns to the original ‘identity‘ data: ‘weights‘, the estimated inverse probability weights, and ‘ranking‘, the pasted full ranking profile.

Usage

```
identity_w
```

Format

‘identity_w‘ A data frame with 1,082 rows and 18 columns:

weights Estimated weights based on inverse probability weighting.

s_weight Survey weight.

app_identity Full ranking profile for the main identity ranking question.

party Marginal ranking for party (main identity ranking question).

religion Marginal ranking for religion (main identity ranking question).

gender Marginal ranking for gender (main identity ranking question).

race Marginal ranking for race (main identity ranking question).

anc_identity Full ranking profile for the anchor ranking question.

household Marginal ranking for household (anchor question).

neighborhood Marginal ranking for neighborhood (anchor question).

city Marginal ranking for city (anchor question).

state Marginal ranking for state (anchor question).

anc_correct_identity Whether the respondent answered the anchor questions correctly. This is a binary variable that 1 if the respondent correctly answers the anchor ranking question and 0 if otherwise.

app_identity_recorded Recorded responses for the main identity ranking question.

anc_identity_recorded Recorded responses for the anchor ranking question.

app_identity_row_rnd The order in which the items were randomly presented for the respondent in the main ranking question.

anc_identity_row_rnd The order in which the items were randomly presented for the respondent in the anchor ranking question.

ranking Pasted full ranking profile reconstructed from the marginal ranking columns. In this dataset, it matches `app_identity`.

Source

<https://github.com/sysilviakim/ranking_error>

`imprr_direct` *Implements Plug-in Bias-Corrected Estimators for Ranking Data*

Description

This function implements the bias correction of the ranking distribution using a paired anchor question.

Usage

```
imprr_direct(
  data,
  J = NULL,
  main_q,
  anc_correct = NULL,
  population = "non-random",
  assumption = "contaminated",
  n_bootstrap = 200,
  seed = 123456,
  weight = NULL,
  verbose = FALSE,
  p_random = NULL
)
```

Arguments

data	The input dataset with ranking data.
J	The number of items in the ranking question. Defaults to NULL, in which case it will be inferred from the data.
main_q	Ranking question to be analyzed. When 'main_q' is a single column name or unquoted symbol such as 'my_ranking', the function looks for 'my_ranking_1', 'my_ranking_2', 'my_ranking_3', and so on. You may also supply 'main_q' directly as a character vector or unquoted 'c(...)' expression of ranking columns such as 'c(party, religion, gender, race)'.
anc_correct	Optional indicator for passing the anchor question. If 'NULL', 'p_random' is used when supplied; otherwise the function defaults to 'p_random = 0' and applies no correction.
population	Choice of the target population out of non-random respondents (default) or all respondents.
assumption	Choice of identifying assumption when 'population = "all"': 'uniform' assumes random respondents would have uniform counterfactual preferences, while 'contaminated' assumes their counterfactual preferences match those of non-random respondents.
n_bootstrap	Number of bootstraps. Defaults to 200.
seed	Seed for set.seed for reproducibility.
weight	The name of the weight column in 'data'. Defaults to 'NULL', which uses equal weights. This can also be supplied as a numeric vector or as an unquoted column name.
verbose	Indicator for verbose output. Defaults to FALSE.
p_random	Optional fixed proportion of random/inattentive respondents. When supplied, this overrides 'anc_correct' and a message is shown if both are provided.

Value

A list with two elements:

est_p_random A data frame with summary statistics for the estimated proportion of random respondents, including columns mean, lower, and upper (95% confidence interval).

results A tibble with bias-corrected estimates grouped by item, qoi (quantity of interest), and outcome, including columns mean, lower, and upper.

Examples

```
out <- imprr_direct(
  identity,
  main_q = c("party", "religion", "gender", "race"),
  anc_correct = "anc_correct_identity",
  n_bootstrap = 1,
  seed = 123
)
out$est_p_random
```

```
head(out$results)
```

`imprr_direct_rcpp` *Implements Plug-in Bias-Corrected Estimators for Ranking Data (Rcpp)*

Description

This function implements the bias correction of the ranking distribution using a paired anchor question. This is a fast Rcpp-based implementation that is approximately 200-300x faster than the tidyverse version.

Usage

```
imprr_direct_rcpp(
  data,
  J = NULL,
  main_q,
  anc_correct = NULL,
  population = "non-random",
  assumption = "contaminated",
  n_bootstrap = 200,
  seed = 123456,
  weight = NULL,
  verbose = FALSE,
  p_random = NULL
)
```

Arguments

`data`	The input dataset with ranking data.
`J`	The number of items in the ranking question. Defaults to NULL, in which case it will be inferred from the data.
`main_q`	Ranking question to be analyzed. When ‘main_q‘ is a single column name or unquoted symbol such as ‘my_ranking‘, the function looks for ‘my_ranking_1‘, ‘my_ranking_2‘, ‘my_ranking_3‘, and so on. You may also supply ‘main_q‘ directly as a character vector or unquoted ‘c(...)‘ expression of ranking columns such as ‘c(party, religion, gender, race)‘.
`anc_correct`	Optional indicator for passing the anchor question. If ‘NULL‘, ‘p_random‘ is used when supplied; otherwise the function defaults to ‘p_random = 0‘ and applies no correction.
`population`	Choice of the target population out of non-random respondents (default) or all respondents.

assumption	Choice of identifying assumption when ‘population = "all"‘: ‘uniform‘ assumes random respondents would have uniform counterfactual preferences, while ‘contaminated‘ assumes their counterfactual preferences match those of non-random respondents.
n_bootstrap	Number of bootstraps. Defaults to 200.
seed	Seed for set.seed for reproducibility.
weight	The name of the weight column in ‘data‘. Defaults to ‘NULL‘, which uses equal weights. This can also be supplied as a numeric vector or as an unquoted column name.
verbose	Indicator for verbose output. Defaults to FALSE.
p_random	Optional fixed proportion of random/inattentive respondents. When supplied, this overrides ‘anc_correct‘ and a message is shown if both are provided.

Value

A list with two elements:

est_p_random	Summary statistics for the estimated proportion of random respondents (mean, lower, upper)
results	A tibble with bias-corrected estimates for all items, including average ranks, pairwise probabilities, top-k probabilities, and marginal probabilities

Examples

```
out <- imprr_direct_rcpp(
  identity,
  main_q = c("party", "religion", "gender", "race"),
  anc_correct = "anc_correct_identity",
  n_bootstrap = 1,
  seed = 123
)
out$est_p_random
head(out$results)
```

---

imprr_weights *Computes Bias-Correction Weights for Ranking Data*

Description

This function implements the bias correction of the ranking distribution using a paired anchor question, using the IPW estimator.

Usage

```
imprr_weights(
  data,
  J = NULL,
  main_q,
  anc_correct = NULL,
  population = "non-random",
  assumption = "contaminated",
  weight = NULL,
  ranking = "ranking",
  p_random = NULL
)
```

Arguments

data	The input dataset with ranking data.
J	The number of items in the ranking question. Defaults to NULL, in which case it will be inferred from the data.
main_q	Ranking question to be analyzed. When ‘main_q‘ is a single column name or unquoted symbol such as ‘my_ranking‘, the function looks for ‘my_ranking_1‘, ‘my_ranking_2‘, ‘my_ranking_3‘, and so on. You may also supply ‘main_q‘ directly as a character vector or unquoted ‘c(...)‘ expression of ranking columns such as ‘c(party, religion, gender, race)‘.
anc_correct	Optional indicator for passing the anchor question. If ‘NULL‘, ‘p_random‘ is used when supplied; otherwise the function defaults to ‘p_random = 0‘ and applies no correction.
population	Choice of the target population out of non-random respondents (default) or all respondents.
assumption	Choice of identifying assumption when ‘population = "all"‘: ‘uniform‘ assumes random respondents would have uniform counterfactual preferences, while ‘contaminated‘ assumes their counterfactual preferences match those of non-random respondents.
weight	The name of the weight column in ‘data‘. Defaults to ‘NULL‘, which uses equal weights. This can also be supplied as a numeric vector or as an unquoted column name.
ranking	The name of the column that will store the full ranking profile. Defaults to "ranking". If ‘main_q‘ exists in the data, the produced column should be identical to ‘main_q‘. However, the function defaults to creating another column by combining marginal rankings, just in case.
p_random	Optional fixed proportion of random/inattentive respondents. When supplied, this overrides ‘anc_correct‘ and a message is shown if both are provided.

Details

‘imprr_weights()‘ enumerates the full permutation space of rankings, so its computational cost grows factorially in ‘J‘. In practice, it is best suited to small or moderate ranking questions. For larger ‘J‘, prefer ‘imprr_direct()‘ or ‘imprr_direct_rcpp()‘.

Value

A list with three elements:

est_p_random A numeric value representing the estimated proportion of random responses.

results A data frame with the original data augmented with a `weights` column containing inverse probability weights and a `ranking` column with unified ranking patterns.

rankings A data frame with ranking patterns, observed proportions (`prop_obs`), bias-corrected proportions (`prop_bc`), and inverse probability weights (`weights`) for each permutation.

Examples

```
out <- imprr_weights(
  identity,
  main_q = c("party", "religion", "gender", "race"),
  anc_correct = "anc_correct_identity"
)
head(out$results)
head(out$rankings)
```

---

imprr_weights_boot *Bootstrap IPW-Based Bias-Corrected Estimates for Ranking Data*

Description

This function repeatedly resamples respondents, reruns `imprr_weights()`, and summarizes downstream quantities of interest such as average ranks, pairwise probabilities, top-k probabilities, and marginal rank probabilities. It provides bootstrap uncertainty estimates for the IPW workflow in a format parallel to `imprr_direct()`.

Usage

```
imprr_weights_boot(
  data,
  J = NULL,
  main_q,
  anc_correct = NULL,
  population = "non-random",
  assumption = "contaminated",
  n_bootstrap = 200,
  seed = 123456,
  weight = NULL,
  verbose = FALSE,
  p_random = NULL
)
```

Arguments

data	The input dataset with ranking data.
J	The number of items in the ranking question. Defaults to NULL, in which case it will be inferred from the data.
main_q	Ranking question to be analyzed. When 'main_q' is a single column name or unquoted symbol such as 'my_ranking', the function looks for 'my_ranking_1', 'my_ranking_2', 'my_ranking_3', and so on. You may also supply 'main_q' directly as a character vector or unquoted 'c(...)' expression of ranking columns such as 'c(party, religion, gender, race)'.
anc_correct	Optional indicator for passing the anchor question. If 'NULL', 'p_random' is used when supplied; otherwise the function defaults to 'p_random = 0' and applies no correction.
population	Choice of the target population out of non-random respondents (default) or all respondents.
assumption	Choice of identifying assumption when 'population = "all"': 'uniform' assumes random respondents would have uniform counterfactual preferences, while 'contaminated' assumes their counterfactual preferences match those of non-random respondents.
n_bootstrap	Number of bootstrap resamples. Defaults to 200.
seed	Seed for set.seed for reproducibility.
weight	The name of the weight column in 'data'. Defaults to 'NULL', which uses equal weights. This can also be supplied as a numeric vector or as an unquoted column name.
verbose	Indicator for verbose output. Defaults to FALSE.
p_random	Optional fixed proportion of random/inattentive respondents. When supplied, this overrides 'anc_correct' and a message is shown if both are provided.

Value

A list with two elements:

est_p_random A data frame with summary statistics for the estimated proportion of random respondents, including columns mean, lower, and upper (95% confidence interval).

results A data frame with bootstrap summaries for the IPW-based bias-corrected quantities of interest, grouped by item, qoi, and outcome, with columns mean, lower, and upper.

Examples

```
out <- imprr_weights_boot(
  identity,
  main_q = c("party", "religion", "gender", "race"),
  anc_correct = "anc_correct_identity",
  n_bootstrap = 2,
  seed = 123
)
out$est_p_random
```

```
head(out$results)
```

`item_to_rank` *Return Rankings with Items as Columns*

Description

This function takes a ranking dataset with rankings as columns and returns a dataset with items as columns and rankings as cell values. This function is useful for converting rankings to a format that allows for average ranking calculations.

Usage

```
item_to_rank(
  item_rank,
  format_input = "ordering",
  reference = NULL,
  long = FALSE
)
```

Arguments

`item_rank`	A data frame with rankings as columns, with items being ranked as cell values.
`format_input`	Character string indicating the format of the data input, namely "ordering" or "ranking". The function returns the corresponding inverse representation.
`reference`	A character vector of item names to be used for renaming the columns. If not specified, will use the first 26 letters of the alphabet. Default is 'NULL'.
`long`	Whether to return the output in a long data format. Default is 'FALSE'.

Value

A data frame with items that are being ranked as columns, with rankings in cell values.

Examples

```
true_pref <- data.frame(
  first = c("b", "a", "c"),
  second = c("c", "b", "b"),
  third = c("a", "c", "a")
)
item_to_rank(true_pref)
item_to_rank(true_pref, long = TRUE)
```

ordinal_seq *Generate an Ordinal Sequence from a Number*

Description

This function generates an ordinal sequence of an arbitrary length. For example, if the length is 3, the function will return the vector `c("1st", "2nd", "3rd")`. This function is used within `avg_rank` and such functions.

Usage

```
ordinal_seq(length)
```

Arguments

`length`	The length of the ordinal sequence to generate. It should be a numeric value of length 1.

Value

A vector of ordinal strings.

Examples

```
ordinal_seq(11)
```

permn_augment *Augmenting Permutation Patterns*

Description

In some distribution of ranking data, not all possible permutation patterns may be realized due to the sample size or skewed distribution of preferences.

Usage

```
permn_augment(tab, J = NULL)
```

Arguments

`tab`	A table of observed permutation patterns.
`J`	The length of the reference choice set. Defaults to 'NULL', in which case it is inferred from the permutation labels. For 'J > 9', delimiter-separated or zero-padded labels are unambiguous. Compact labels such as '"12345678910"' are parsed when possible, and the augmented output is returned in an unambiguous canonical format.

Details

This function augments the given table with all possible observed permutation patterns with a frequency of zero for unrealized patterns. Currently, this only takes full rankings into account, as opposed to partial rankings.

Value

A table of observed permutation patterns augmented with all possible permutation patterns.

Examples

```
tab <- table(c(rep("123", 100), rep("321", 50)))
permn_augment(tab, J = 3)

tab <- table(c("123", "321", "213", "312", "132", "231"))
permn_augment(tab, J = 3)
```

---

plot.rankingQ_output *Plot rankingQ estimator outputs*

Description

Plot rankingQ estimator outputs

Usage

```
## S3 method for class 'rankingQ_output'
plot(
  x,
  y = NULL,
  type = "average_rank",
  method = NULL,
  item = NULL,
  xlab = NULL,
  ylab = "",
  ...
)

## S3 method for class 'rankingQ_output'
autoplot(
  object,
  type = "average_rank",
  method = NULL,
  item = NULL,
  conf.int = TRUE,
  xlab = NULL,
```

```
  ylab = "",
  ...
)
```

Arguments

x	A rankingQ estimator output object.
y	Ignored.
type	Estimate type to plot. Defaults to "average_rank".
method	Which estimator to plot. Defaults to the object's primary method.
item	Optional item filter.
xlab	X-axis label. If NULL, a sensible default is used.
ylab	Y-axis label. Defaults to an empty string.
...	Passed through to autoplot().
object	A rankingQ estimator output object.
conf.int	If TRUE, confidence intervals are drawn when available.

Value

A ggplot object.

---

plot_avg_ranking *Plot Average Rank Results*

Description

This function takes the output from the 'imprr_direct' function and plots the average rank results with confidence intervals. As long as the mean and the confidence intervals are provided, this function will plot other quantities of interest such as marginal, pairwise, top-k rankings.

Usage

```
plot_avg_ranking(data, qoi_filter = "average rank", xlab = NULL, ylab = "")
```

Arguments

data	The results data from the 'imprr_direct' function. If an external data frame, make sure that the column names are the same as the output from the 'imprr_direct' function.
qoi_filter	The quantity of interest (QOI) to filter for. Defaults to "average rank".
xlab	The x-axis label. Defaults to NULL. If NULL and 'qoi_filter' is provided, a simple capitalized label based on 'qoi_filter' is used. If 'qoi_filter' is NULL, the default ggplot x-axis label is left unchanged. If you'd like it to be empty, specify an empty string.
ylab	The y-axis label. Defaults to an empty string.

Value

A ggplot object.

Examples

```
avg_rank_results <- data.frame(
  item = c("Party", "Religion", "Gender", "Race"),
  qoi = rep("average rank", 4),
  mean = c(1.7, 2.1, 2.8, 3.4),
  lower = c(1.5, 1.9, 2.6, 3.2),
  upper = c(1.9, 2.3, 3.0, 3.6)
)
plot_avg_ranking(avg_rank_results)
```

---

plot_dist_ranking *Plot the Distribution of Rankings Over the Permutation Space*

Description

This function takes a table in which the frequencies of ranking patterns are recorded and plots it over the permutation space of rankings, using the ggplot2 package.

Usage

```
plot_dist_ranking(
  tab,
  x = "ranking",
  y = "prop",
  ylim = 0.315,
  fill = "firebrick4",
  xlab = "Recorded Responses",
  family = NULL,
  vjust = -0.5,
  size = 3,
  linetype = "dashed",
  h_color = "black",
  h_alpha = 0.5
)
```

Arguments

tab	A table in which the frequencies of ranking patterns are recorded.
x	Name of the column that contains permutation patterns.
y	Name of the column that contains the plotted values (for example, proportions or frequencies).
ylim	The upper limit of the y-axis.

fill	The color of the bars.
xlab	The label of the x-axis. Defaults to "Recorded Responses".
family	The font family of the text.
vjust	The vertical justification of the text.
size	The size of the text in ‘geom_text‘.
linetype	The linetype in ‘geom_hline‘.
h_color	The color in ‘geom_hline‘.
h_alpha	The transparency in ‘geom_hline‘.

Value

A ggplot2 object.

Examples

```
tab <- lapply(combinat::permn(seq(3)), paste0, collapse = "") |>
  sample(30, replace = TRUE) |>
  unlist() |>
  table() |>
  table_to_tibble()
plot_dist_ranking(tab, ylim = 0.5)
```

---

rank_longer *Convert Ranking Columns from Wide to Long Format*

Description

This function takes a data frame in wide format with columns recording rankings into a long data format.

Usage

```
rank_longer(x, cols = NULL, id = NULL, reference = NULL)
```

Arguments

x	A data frame in wide format with columns recording rankings.
cols	A character vector of column names that record rankings. If there are multiple columns, the order of the columns should be the same as the order of the reference choice set. If there is a single column, the order in which the numbers appear in respondent-level character response should be the same as the order of the reference set.
id	The column that uniquely identify the respondent.
reference	If you wish to specify the reference choice set, you can provide a character vector.

Details

If the data frame has more than one columns specified in the `cols` argument, they will be translated as the first, second, third, etc. items in the reference choice set. For example, if the first column records 2, the second column records 1, and the third column records 3, then the function will interpret that this respondent prefers the second item the most, then the first item, then the third item.

If the data frame has only one column specified in the `cols` argument, it will be parsed by character length. For example, if the column records "213", then the function will interpret that this respondent prefers the second item the most, then the first, and then the third item.

Currently, this function depends on `tidyverse` functions. Eventually, a `data.table` option will be added for large datasets.

Value

A data frame in long format with columns recording rankings. The first column is the id variable that has been pre-specified. The second and third columns record what item is being ranked. The final column records the ranking of the item.

Examples

```
x <- data.frame(
  apple = c(2, 1, 3),
  orange = c(1, 3, 2),
  banana = c(3, 2, 1)
)
rank_longer(x)

y <- data.frame(
  id = c("Bernie", "Yuki", "Silvia"),
  rank = c("123", "321", "213")
)
rank_longer(y, cols = "rank", id = "id")
rank_longer(
  y,
  cols = "rank", id = "id",
  reference = c("Money", "Power", "Respect")
)
```

`rank_wider` *Turn Long Ranking Data into a Wide Format*

Description

This function takes ranking data in long format and returns a wide-format data frame with one row per respondent. It can return either one column per ranked item or a single pasted ranking string.

Usage

```
rank_wider(
  x,
  id,
  item = "item_name",
  rank = "ranking",
  output = c("multiple", "single"),
  reference = NULL,
  ranking_name = "ranking"
)
```

Arguments

x	A data frame in long format with respondent identifiers, item names, and ranks.
id	The column that uniquely identifies the respondent.
item	The column that contains item names. Defaults to "item_name".
rank	The column that contains rank values. Defaults to "ranking".
output	The desired output format: "multiple" for one column per item or "single" for one pasted ranking string. Defaults to "multiple".
reference	Optional character vector giving the reference choice-set order. If omitted and reference_no is present in x, item order is inferred from that column.
ranking_name	The name of the output column when output = "single". Defaults to "ranking".

Value

A data frame in wide format with one row per respondent.

Examples

```
x <- data.frame(
  id = c(1, 1, 1, 2, 2, 2),
  item_name = c("A", "B", "C", "A", "B", "C"),
  ranking = c(1, 2, 3, 3, 2, 1)
)

rank_wider(x, id = "id")
rank_wider(
  x,
  id = "id",
  output = "single",
  reference = c("A", "B", "C")
)
```

`recover_recorded_responses` *Recover the Recorded Responses Given that Ranking Items were Randomized*

Description

This function, using the order of the items that was presented to the respondent as well as the true responses to the ranking question, recovers the recorded responses, or the responses that the respondent actually provided, ignoring the order in which the items were presented.

Usage

```
recover_recorded_responses(
  true_order,
  presented_order,
  df = NULL,
  reference = NULL
)
```

Arguments

`true_order`	A string representing the true ranking of the respondent with respect to the reference choice set.
`presented_order`	A string representing the order of the items that were presented to the respondent.
`df`	The input data frame. Defaults to NULL. If NULL, the function expects inputs as simple strings such as "321" or "312".
`reference`	Optional reference choice-set order. This is only needed when mixing numeric position codes with item-label inputs. It can be supplied as a character vector such as `c("A", "B", "C", "D")` or as a single compact/delimited string such as `"ABCD"` or `"A|B|C|D"`.

Details

This is to see if behavior such as diagonalization occurred. Most survey software will take the recorded response and translate it into the true ranking of the respondent (observed ranking), but not providing the recorded response.

For example, given a reference choice set of three items, A, B, and C, the respondent may have been presented with the items in the order C, B, A, and may have responded with 3-2-1 as a recorded response. The true order of the items is A, B, C, so the observed/true ranking is 1-2-3. This function takes C, B, A and 1-2-3 as inputs and returns 3-2-1.

Value

If `df = NULL`, a character string giving the recovered recorded response. Otherwise, the original data frame augmented with a column containing the recovered recorded response.

Examples

```
## This respondent's true ranking reported is A-B-C-D.
## However, the items were presented in the order B-A-D-C.
## Therefore, the respondent's recorded response is 2-1-4-3.
recover_recorded_responses(true_order = "1234", "2143") ## Output: "2143"

## This respondent's true ranking is reported as D-C-B-A.
## However, the items were presented in the order A-B-C-D.
## Therefore, the respondent's recorded response is 4-3-2-1.
recover_recorded_responses(true_order = "4321", "1234") ## Output: "4321"

## This respondent's true ranking is reported as C-A-D-B.
## However, the items were presented in the order D-C-B-A.
## Therefore, the respondent's recorded response is 3-1-4-2.
recover_recorded_responses(true_order = "2413", "4321") ## Output: "3142"

## The same example using item labels directly.
recover_recorded_responses("CADB", "DCBA") ## Output: "3142"

## You can also mix numeric rankings with labeled presentation order
## if the reference choice set is supplied explicitly.
recover_recorded_responses("2413", "D|C|B|A", reference = c("A", "B", "C", "D"))
```

---

rpluce *Draw Samples from the Plackett-Luce Model*

Description

This function draws samples from the Plackett-Luce model, using Algorithm 2.1, "Efficient Sampling from Plackett-Luce," in Xia (2019), page 20, Section 2.2.3, "Sampling from Random Utility Models." The name `rpluce` is a convention that follows random generations of numbers from statistical distributions such as `rnorm` or `rmultinom`.

Usage

```
rpluce(n, t, prob, choices = NULL, seed = NULL)
```

Arguments

`n`	The total number of samples to draw.
`t`	The number of items or alternatives to choose from.
`prob`	A vector of choice probabilities.
`choices`	A vector of choices to be ranked.
`seed`	An optional seed for the random number generator.

Details

Input: A parameter $\overrightarrow{\gamma} = (\gamma_1, \cdots, \gamma_m)$ of the Plackett-Luce model.

If all remaining Plackett-Luce weights become zero after earlier draws, the remaining items are sampled uniformly at random rather than being ordered by their input position.

Output: A ranking $R \in \mathcal{L}(\mathcal{A})$ from $\pi_{\overrightarrow{\gamma}}(\cdot)$ under Plackett-Luce.
1: Let $R = \emptyset$ and $A = \mathcal{A}$.
2: for $t = 1$ to m do
3: Choose an alternative a_{i_t} from A with probability proportional to γ_{i_t}.
4: $R \leftarrow R \succ a_{i_t}$ and $A \leftarrow A \setminus \{a_{i_t}\}$.
5: end for
6: return R.

Value

A data frame of rankings of t items for n assessors.

Examples

```
rpluce(n = 10, t = 3, prob = c(0.5, 0.3, 0.2), seed = 123)
```

`stratified_avg` *Stratified Estimate of Average Ranks*

Description

This function estimates the average ranks based on stratification.

Usage

```
stratified_avg(
  data,
  var_stratum,
  J = NULL,
  main_q,
  anc_correct = NULL,
  labels = NULL,
  seed = 1234,
  weight = NULL,
  n_bootstrap = 200,
  ipw = FALSE,
  verbose = FALSE,
  p_random = NULL
)
```

Arguments

data	A data frame containing the ranking data as well as the stratifying variable.
var_stratum	The name of the stratifying variable.
J	The number of items in the ranking question. Defaults to NULL, in which case it will be inferred from the data.
main_q	Main ranking question specification. This can be a single column name or unquoted symbol such as ‘my_ranking‘, in which case the function looks for ‘my_ranking_1‘, ‘my_ranking_2‘, and so on. You may also supply ‘main_q‘ directly as a character vector or unquoted ‘c(...)‘ expression of ranking columns.
anc_correct	Optional indicator for passing the anchor question. If ‘NULL‘, ‘p_random‘ is used when supplied; otherwise the function defaults to ‘p_random = 0‘ and applies no correction.
labels	A vector of labels for the items being ranked. Defaults to NULL.
seed	Seed for set.seed for reproducibility.
weight	Either a numeric vector of weights with length ‘nrow(data)‘, the name of a weight column in ‘data‘, or an unquoted weight column name. Defaults to ‘NULL‘.
n_bootstrap	Number of bootstraps. Defaults to 200.
ipw	Indicator for using inverse probability weighting. Defaults to FALSE, in which case direct bias estimation will be employed.
verbose	Indicator for verbose output. Defaults to FALSE.
p_random	Optional fixed proportion of random/inattentive respondents. When supplied, this overrides ‘anc_correct‘ and a message is shown if both are provided.

Value

A data frame with the bootstrap-estimated average ranks.

Examples

```
identity2 <- identity
identity2$stratum <- rep(c("group1", "group2"), length.out = nrow(identity2))
out <- suppressMessages(stratified_avg(
  identity2,
  var_stratum = "stratum",
  main_q = c("party", "religion", "gender", "race"),
  p_random = 0,
  n_bootstrap = 1,
  seed = 123
))
head(out)
```

summary.rankingQ_output *Summarize rankingQ estimator outputs*

Description

Summarize rankingQ estimator outputs

Usage

```
## S3 method for class 'rankingQ_output'
summary(object, method = NULL, type = "average_rank", item = NULL, n = 6L, ...)

## S3 method for class 'summary.rankingQ_output'
print(x, digits = 3L, ...)
```

Arguments

object	A rankingQ estimator output object.
method	Which estimator to summarize. Defaults to the object's primary method.
type	Estimate type to display. Defaults to "average_rank".
item	Optional item filter.
n	Number of rows to preview in the printed summary.
...	Unused.
x	A summary.rankingQ_output object.
digits	Number of digits to print.

Value

A summary object with a compact human-readable overview.

table_to_tibble *Turn the Frequency Table into a Tibble or Data Frame*

Description

This function converts a frequency table to a tibble or data frame. It also creates a proportion variable as well as the frequency variable. This function is useful when plotting distribution of ranking patterns; see relevant vignette.

Usage

```
table_to_tibble(tab, tibble = TRUE)
```

Arguments

tab	A frequency table.
tibble	A logical value indicating whether the output should be a tibble or data frame. Default is TRUE.

Value

A tibble or data frame, depending on the tibble argument.

Examples

```
tab <- lapply(combinat::permn(seq(3)), paste0, collapse = "") |>
  sample(30, replace = TRUE) |>
  unlist() |>
  table()
table_to_tibble(tab)
```

---

tidy.rankingQ_output *Tidy rankingQ estimator outputs*

Description

Tidy rankingQ estimator outputs

Usage

```
## S3 method for class 'rankingQ_output'
tidy(
  x,
  component = c("estimates", "p_random"),
  method = NULL,
  type = NULL,
  item = NULL,
  conf.int = TRUE,
  ...
)
```

Arguments

x	A rankingQ estimator output object.
component	Which part of the object to tidy. Defaults to "estimates".
method	Which estimator to return. Supported values are "raw", "direct", and "ipw" when available for the object.
type	Estimate type filter. Supported values are "average_rank", "pairwise", "top_k", and "marginal".

item	Optional item filter.
conf.int	If FALSE, confidence-interval columns are omitted from the returned tibble.
...	Unused.

Value

A tibble with a standardized layout for ranking estimates.

---

unbiased_correct_prop *Unbiased Estimator of the Proportion of Random and Non-random Responses*

Description

This function computes the unbiased proportion of *correct* answers after adjusting for the possibility that the respondent may have randomly guessed the correct answer. The function is based on the formula provided by Proposition 1 of Atsusaka and Kim (2025).

Usage

```
unbiased_correct_prop(mean_c, J)
```

Arguments

mean_c	This is the raw proportion of the correct answers.
J	The number of items to rank order.

Value

A number between 0-1.

Examples

```
unbiased_correct_prop(0.7, 3)
```

`uniformity_test` *Uniformity Test for Ranking Patterns*

Description

This function implements the uniformity test for ranking permutation patterns documented in Atsusaka and Kim (2025).

Usage

```
uniformity_test(data, var = NULL)
```

Arguments

`data`	The input dataset with ranking data.
`var`	The variable within `data` to be used in the test. Defaults to NULL.

Value

A chi-square test result.

Examples

```
tab <- table(c(
  rep("123", 10), rep("132", 10), rep("213", 10),
  rep("231", 10), rep("312", 10), rep("321", 10)
))
uniformity_test(tab)
```

Index