Package: electsys21 (via r-universe)

November 20, 2026

Type Package

Title Voting Methods for Ranked, Rated and Approval Ballots

Version 0.1.0

Description Implements a range of voting methods and electoral systems for determining election winners, including the D21 method with and without minus votes (Janecek, <https://www.ih21.org/en/d21-janecek-method>), first-past-the-post, two-round runoff, instant runoff, the Borda count, approval voting, majority judgement and the Condorcet method. The functions accept several ballot formats - ranking, cardinal utilities, approvals and scores - with automatic detection of the input type, configurable tie-breaking and tidy summaries of the results.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/ivan-ih21/electsys21

BugReports https://github.com/ivan-ih21/electsys21/issues

Imports stats

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Repository https://ivan-ih21.r-universe.dev

Date/Publication 2026-11-20 15:11:53 UTC

RemoteUrl https://github.com/ivan-ih21/electsys21

RemoteRef HEAD

RemoteSha 2e18b36adf4e4c22f318ef5b925f149fb47b3ecf

Contents

approval 2
borda 3

condorcet 5
d21 6
d21_minus 8
fptp 10
gen_approvals 11
gen_ranks 12
gen_utilities 13
irv 15
majority_judgement 16
tworound 18

Index **20**

`approval` *Approval Voting*

Description

Runs an election under Approval Voting, where each voter may approve any number of candidates and the candidate with the most approvals wins. When the input is not already a binary approval matrix, `rule` converts the rankings or utility scores into approvals. Approvals are then summed across voters and ties for the top approval count are resolved according to `ties`.

Usage

```
approval(
  x,
  type = c("auto", "rank", "utility", "approval"),
  rule = c("mean", "median", "topk", "topp", "quantile", "above0", "nonzero"),
  threshold = NULL,
  ties = c("random", "lexicographic", "all"),
  return_approvals = FALSE
)
```

Arguments

`x`	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank `1` = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named `Candidate_1`, `Candidate_2`, and so on.
`type`	Character string giving the ballot type. One of `"auto"` (the default, detected from `x`), `"rank"`, `"utility"` or `"approval"`.
`rule`	The approval rule used to convert non-approval ballots (ranks or utilities) into approvals; ignored when the input is already an approval matrix. May be a preset string – `"mean"` (the default; score above the voter's mean), `"median"` (above the voter's median), `"topk"` (top `k` by score, `k` from `threshold`), `"topp"` (top

proportion by score, p from threshold), "quantile" (above the threshold-th quantile), "above0" (positive score, or all ranked candidates for rank input), or "nonzero" (non-zero score, or all ranked candidates for rank input) – or a numeric scalar/vector of length ncol(x) (eligible if score > rule for utility, or rank <= rule for rank input), or a custom function with signature function(row, inferred_type) returning a logical vector of length ncol(x).

threshold — Single numeric value (or NULL, the default) parameterizing the "topk" (number of candidates to approve), "topp" (proportion in [0, 1]) and "quantile" (quantile in [0, 1]) presets. Ignored for all other rules.

ties — Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of "random" (the default; pick one tied candidate at random), "lexicographic" (pick the alphabetically first) or "all" (return all tied candidates).

return_approvals — Logical; if TRUE, the result additionally includes the derived binary approval matrix as $approvals. Defaults to FALSE.

Value

An object of class "approval_result", a list with elements summary (a data frame of candidates, approval counts, percentages and ranks), winners (the winning candidate name(s)), n_voters, n_valid, n_candidates and method (a list recording the inferred type, rule, ties and threshold). When return_approvals = TRUE, the derived binary approval matrix is also returned as $approvals. Print the object or call summary() on it for a formatted results table.

See Also

fptp(), borda(), tworound()

Examples

```
# Direct approval ballots
ballots <- gen_approvals(n_voters = 50, n_candidates = 4, seed = 1)
approval(ballots)

# Utility ballots converted to approvals via a rule
scores <- gen_utilities(n_voters = 50, n_candidates = 4, seed = 1)
approval(scores, type = "utility", rule = "mean")
```

---

borda *Borda Count*

Description

Runs an election using the Borda Count. Each voter's ballot is turned into a ranking and each candidate earns points according to their position: with `n` candidates, a candidate ranked first receives `n - 1` points, the second `n - 2`, down to `0` for the last. Points are summed across all voters and the candidate with the highest total wins. Utility and approval ballots are converted to rankings internally before scoring.

Usage

```
borda(
  x,
  type = c("auto", "rank", "utility", "approval"),
  ties = c("random", "lexicographic", "all"),
  return_ranks = FALSE,
  return_scores = FALSE
)
```

Arguments

`x`	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank `1` = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named `Candidate_1`, `Candidate_2`, and so on.
`type`	Character string giving the ballot type. One of `"auto"` (the default, detected from `x`), `"rank"`, `"utility"` or `"approval"`.
`ties`	Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of `"random"` (the default; pick one tied candidate at random), `"lexicographic"` (pick the alphabetically first) or `"all"` (return all tied candidates).
`return_ranks`	Logical. If `TRUE`, the ranking matrix derived internally is attached to the result as `$ranks`. Defaults to `FALSE`.
`return_scores`	Logical. If `TRUE`, the per-voter Borda points matrix (`n_voters` by `n_candidates`) used for scoring is attached to the result as `$scores`. Defaults to `FALSE`.

Value

An object of class `"borda_result"`, a list with elements: `summary` (a data frame of candidates ordered by total score, with columns `candidate`, `score`, `percentage` and `rank`), `winners` (a character vector of the winning candidate name(s), or `character(0)` if no voter provided a valid preference), `n_voters`, `n_valid`, `n_candidates`, and `method` (a list recording the inferred `type` and the `ties` rule). Optionally includes `$ranks` (when `return_ranks = TRUE`) and `$scores` (when `return_scores = TRUE`). Print the object or call `summary()` on it for a formatted results table.

See Also

`fptp()`, `condorcet()`, `approval()`

Examples

```
ballots <- gen_ranks(n_voters = 50, n_candidates = 4, seed = 1)
borda(ballots)

# Resolve ties alphabetically and keep the derived ranking matrix
borda(ballots, ties = "lexicographic", return_ranks = TRUE)
```

---

condorcet *Condorcet Method*

Description

Determines the winner by comparing every candidate head-to-head against every other candidate. In each pairwise contest the candidate preferred by a strict majority of voters wins; the **Condorcet winner** is the candidate who beats all others. Such a candidate need not exist, in which case `winners` is empty. A Copeland-style score (wins minus losses) is always computed so the full field is ordered even when no Condorcet winner exists.

Usage

```
condorcet(
  x,
  type = c("auto", "rank", "utility", "approval"),
  ties = c("random", "lexicographic", "all"),
  return_pairwise = FALSE
)
```

Arguments

`x`	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank `1` = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named `Candidate_1`, `Candidate_2`, and so on.
`type`	Character string giving the ballot type. One of `"auto"` (the default, detected from `x`), `"rank"`, `"utility"` or `"approval"`.
`ties`	Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of `"random"` (the default; pick one tied candidate at random), `"lexicographic"` (pick the alphabetically first) or `"all"` (return all tied candidates).
`return_pairwise`	Logical. If `TRUE`, two extra components are attached to the result: `$pairwise`, the candidate-by-candidate matrix where `[i, j]` is 1 iff candidate `i` beats candidate `j` head-to-head, and `$pairwise_margins`, the matrix whose `[i, j]` entry counts the voters who prefer candidate `i` over candidate `j`. Defaults to `FALSE`.

Value

An object of class `"condorcet_result"`, a list with elements `summary` (a data frame of candidates ordered by Copeland score, with columns `candidate`, `wins`, `losses`, `ties`, `rank` and `status`), `winners` (name(s) of the Condorcet winner, or `character(0)` if none), `losers` (name(s) of the Condorcet loser, or `character(0)` if none), `n_voters`, `n_valid`, `n_candidates`, and `method` (a list recording the inferred `type` and the `ties` rule). When `return_pairwise = TRUE` the elements `pairwise` and `pairwise_margins` are also included. Print the object or call `summary()` on it for a formatted results table.

See Also

`borda()`, `irv()`, `approval()`

Examples

```
ballots <- gen_ranks(n_voters = 30, n_candidates = 4, seed = 1)
condorcet(ballots)

condorcet(ballots, return_pairwise = TRUE)
```

---

d21 *D21 Voting*

Description

Runs the D21 method. Each voter is granted a fixed number of **plus votes** (1, 2, or 3, depending on the number of candidates) and uses them to support their most preferred candidates. When the input is not already an approval matrix, each voter's eligible candidates are first selected from rankings or utility scores using a configurable `rule`, then capped to the allowed number of plus votes, keeping the highest-scored candidates. The candidate with the most plus votes across all voters wins.

Usage

```
d21(
  x,
  type = c("auto", "rank", "utility", "approval"),
  rule = c("mean", "median", "topk", "topp", "quantile", "above0", "nonzero"),
  threshold = NULL,
  ties = c("random", "lexicographic", "all"),
  overflow = c("random", "lexicographic"),
  return_approvals = FALSE
)
```

Arguments

x	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank 1 = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named Candidate_1, Candidate_2, and so on.
type	Character string giving the ballot type. One of "auto" (the default, detected from x), "rank", "utility" or "approval".
rule	The approval rule used to convert non-approval ballots (ranks or utilities) into eligibility for plus votes; ignored when the input is already an approval matrix. May be a preset string – "mean" (the default; score above the voter's mean), "median" (above the voter's median), "topk" (top k by score, k from threshold), "topp" (top proportion by score, p from threshold), "quantile" (above the threshold-th quantile), "above0" (positive score, or all ranked candidates for rank input), or "nonzero" (non-zero score, or all ranked candidates for rank input) – or a numeric scalar/vector of length ncol(x) (eligible if score > rule for utility, or rank <= rule for rank input), or a custom function with signature function(row, inferred_type) returning a logical vector of length ncol(x).
threshold	Single numeric value (or NULL, the default) parameterizing the "topk" (number of candidates to approve), "topp" (proportion in [0, 1]) and "quantile" (quantile in [0, 1]) presets. Ignored for all other rules.
ties	Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of "random" (the default; pick one tied candidate at random), "lexicographic" (pick the alphabetically first) or "all" (return all tied candidates).
overflow	Character string giving the tie-breaking rule applied per voter when more candidates are eligible than the plus-vote budget allows (and at the "topk"/"topp" cutoffs). One of "random" (the default; sample among tied candidates) or "lexicographic" (choose alphabetically by name).
return_approvals	Logical; if TRUE, the result additionally includes the derived binary approval matrix as $approvals. Defaults to FALSE.

Value

An object of class "d21_result": a list with elements summary (a data frame of candidates, plus-vote counts, percentages and ranks), winners (name(s) of the winning candidate(s)), n_voters, n_valid, n_candidates, and method (a list recording type, rule, ties, threshold, plus_votes and overflow). If return_approvals = TRUE, the element approvals (the derived approval matrix) is also included. Print the object or call summary() on it for a formatted results table.

See Also

d21_minus(), approval(), fptp()

Examples

```
u <- gen_utilities(n_voters = 40, n_candidates = 5, seed = 1)
d21(u, type = "utility")

r <- gen_ranks(n_voters = 40, n_candidates = 5, seed = 1)
d21(r, type = "rank", rule = "topk", threshold = 2)
```

`d21_minus` *D21 Voting with minus votes*

Description

Runs the single-winner D21 method with minus votes. Each voter receives a fixed number of plus votes (1, 2, or 3, depending on the number of candidates) to support their most preferred candidates, and voters who use at least two plus votes may additionally cast one minus vote against the candidate they most oppose. Eligibility for plus votes is derived from ranks or utility scores via a configurable `rule` and then capped to the allowed number; the minus vote targets the lowest-scored candidate the voter did not plus. The candidate with the highest net total wins.

Usage

```
d21_minus(
  x,
  type = c("auto", "rank", "utility", "approval"),
  rule = c("mean", "median", "topk", "topp", "quantile", "above0", "nonzero"),
  threshold = NULL,
  minus_prob = 1,
  ties = c("random", "lexicographic", "all"),
  overflow = c("random", "lexicographic"),
  return_approvals = FALSE
)
```

Arguments

`x`	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank `1` = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named `Candidate_1`, `Candidate_2`, and so on.
`type`	Character string giving the ballot type. One of `"auto"` (the default, detected from `x`), `"rank"`, `"utility"` or `"approval"`.
`rule`	The approval rule used to convert non-approval ballots (ranks or utilities) into eligibility for plus votes; ignored when the input is already an approval matrix. May be a preset string – `"mean"` (the default; score above the voter's mean), `"median"` (above the voter's median), `"topk"` (top `k` by score, `k` from

threshold), "topp" (top proportion by score, p from threshold), "quantile" (above the threshold-th quantile), "above0" (positive score, or all ranked candidates for rank input), or "nonzero" (non-zero score, or all ranked candidates for rank input) – or a numeric scalar/vector of length ncol(x) (eligible if score > rule for utility, or rank <= rule for rank input), or a custom function with signature function(row, inferred_type) returning a logical vector of length ncol(x).

threshold — Single numeric value (or NULL, the default) parameterizing the "topk" (number of candidates to approve), "topp" (proportion in [0, 1]) and "quantile" (quantile in [0, 1]) presets. Ignored for all other rules.

minus_prob — Numeric in [0, 1] (default 1) giving the probability that a voter eligible for a minus vote actually casts it. Setting minus_prob = 0 suppresses all minus votes.

ties — Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of "random" (the default; pick one tied candidate at random), "lexicographic" (pick the alphabetically first) or "all" (return all tied candidates).

overflow — Character string giving the tie-breaking rule used at the plus-vote cap, within the "topk"/"topp" rules, and when selecting the worst candidate for the minus vote. One of "random" (the default; sample among tied candidates) or "lexicographic" (choose alphabetically by name).

return_approvals — Logical. If TRUE, the result additionally includes the per-voter $plus_matrix, $minus_matrix and $total_matrix. Defaults to FALSE.

Value

An object of class "d21_minus_result": a list with elements summary (a data frame of per-candidate plus, minus and total counts with percentages and ranks), winners, n_voters, n_valid, n_candidates and method (a list recording type, rule, ties, threshold, plus_votes, minus_votes, minus_prob and overflow). When return_approvals = TRUE, the elements plus_matrix, minus_matrix and total_matrix are also included. Print the object or call summary() on it for a formatted results table.

See Also

d21(), approval(), borda()

Examples

```
# Utility ballots with 5 candidates (2 plus votes, 1 minus vote)
u <- gen_utilities(n_voters = 50, n_candidates = 5, seed = 1)
d21_minus(u)

# Ranking ballots, eligible = explicitly ranked candidates
r <- gen_ranks(n_voters = 50, n_candidates = 6, seed = 1)
d21_minus(r, rule = "above0", ties = "lexicographic")

# Suppress the minus phase (equivalent to plain D21)
d21_minus(u, minus_prob = 0)
```

fptp *First-Past-The-Post voting*

Description

Runs a First-Past-The-Post election. Each voter contributes a single first-preference vote, and the candidate receiving the most first-preference votes wins. The function accepts ranking, utility (score) or approval ballots, reduces each ballot to its single top choice, tallies the first-preference votes, and resolves any tie for the lead according to `ties`.

Usage

```
fptp(
  x,
  type = c("auto", "rank", "utility", "approval"),
  ties = c("random", "lexicographic", "all"),
  return_ranks = FALSE
)
```

Arguments

`x`	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank `1` = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named `Candidate_1`, `Candidate_2`, and so on.
`type`	Character string giving the ballot type. One of `"auto"` (the default, detected from `x`), `"rank"`, `"utility"` or `"approval"`.
`ties`	Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of `"random"` (the default; pick one tied candidate at random), `"lexicographic"` (pick the alphabetically first) or `"all"` (return all tied candidates).
`return_ranks`	Logical. If `TRUE`, the ranking matrix derived internally is attached to the result as `$ranks`. Defaults to `FALSE`.

Value

An object of class `"fptp_result"`, a list with elements:

- `summary`: a data frame of candidates ordered by votes, with columns `candidate`, `vote`, `percentage` (share of valid votes) and `rank`.
- `winners`: a character vector of the winning candidate name(s).
- `n_voters`: the total number of voters (rows of `x`).
- `n_candidates`: the total number of candidates (columns of `x`).

- `method`: a list recording the inferred `type` and the `ties` rule used.
- `ranks`: optionally, the derived ranking matrix, present only when `return_ranks = TRUE`.

Print the object or call `summary()` on it for a formatted results table.

See Also

`borda()`, `irv()`, `condorcet()`

Examples

```
ballots <- gen_ranks(n_voters = 20, n_candidates = 4, seed = 1)
fptp(ballots)

fptp(ballots, ties = "lexicographic", return_ranks = TRUE)
```

---

gen_approvals *Generate random approval ballots*

Description

Simulates an approval-ballot matrix for testing and demonstration. Each candidate is independently approved by each voter with probability `p_approve`, and voters may abstain entirely.

Usage

```
gen_approvals(
  n_voters,
  n_candidates,
  p_approve = 0.5,
  p_abstain = 0,
  candidate_names = NULL,
  voter_names = NULL,
  seed = NULL
)
```

Arguments

`n_voters`	Single positive integer. Number of voters (matrix rows).
`n_candidates`	Single positive integer. Number of candidates (matrix columns).
`p_approve`	Single number in `[0, 1]`. Probability that a voter approves any given candidate. Defaults to `0.5`.
`p_abstain`	Single number in `[0, 1]`. Probability that a voter abstains entirely, leaving that whole row `NA`. Defaults to `0`.
`candidate_names`	`NULL` or a character vector of length `n_candidates` giving the column names. Defaults to `NULL` (`Candidate_1`, `Candidate_2`, ...).

voter_names	NULL or a character vector of length n_voters giving the row names. Defaults to NULL (Voter_1, Voter_2, ...).
seed	NULL or a single numeric value passed to set.seed() for reproducible output. Defaults to NULL.

Value

A logical matrix with n_voters rows and n_candidates columns: TRUE marks an approved candidate, FALSE a non-approved one, and an abstaining voter's row is all NA. Suitable as the x argument of the voting functions.

See Also

gen_ranks(), gen_utilities()

Examples

```
gen_approvals(n_voters = 5, n_candidates = 3, seed = 1)

# Stricter voters approve fewer candidates
gen_approvals(n_voters = 5, n_candidates = 3, p_approve = 0.25, seed = 1)
```

---

gen_ranks *Generate random ranking ballots*

Description

Simulates a ranking-ballot matrix for testing and demonstration. Latent utilities are drawn uniformly at random for every voter-candidate pair and converted into strict ranks (rank 1 = most preferred). Optionally truncates each ballot to its top n_ranked candidates and lets voters abstain.

Usage

```
gen_ranks(
  n_voters,
  n_candidates,
  n_ranked = NULL,
  p_abstain = 0,
  candidate_names = NULL,
  voter_names = NULL,
  seed = NULL
)
```

Arguments

n_voters	Single positive integer. Number of voters (matrix rows).
n_candidates	Single positive integer. Number of candidates (matrix columns).
n_ranked	NULL or a single positive integer. If set, each voter ranks only their top n_ranked candidates and the remaining entries are set to NA. Defaults to NULL (every candidate is ranked).
p_abstain	Single number in [0, 1]. Probability that a voter abstains entirely, leaving that whole row NA. Defaults to 0.
candidate_names	NULL or a character vector of length n_candidates giving the column names. Defaults to NULL (Candidate_1, Candidate_2, ...).
voter_names	NULL or a character vector of length n_voters giving the row names. Defaults to NULL (Voter_1, Voter_2, ...).
seed	NULL or a single numeric value passed to set.seed() for reproducible output. Defaults to NULL.

Value

A numeric matrix with n_voters rows and n_candidates columns holding ranks (1 = most preferred); unranked or abstained entries are NA. Suitable as the x argument of the voting functions.

See Also

gen_utilities(), gen_approvals()

Examples

```
gen_ranks(n_voters = 5, n_candidates = 3, seed = 1)

# Truncated ballots: each voter ranks only their top 2 candidates
gen_ranks(n_voters = 5, n_candidates = 4, n_ranked = 2, seed = 1)
```

---

gen_utilities *Generate random utility ballots*

Description

Simulates a cardinal utility-ballot matrix for testing and demonstration. Utilities are drawn uniformly at random over a configurable scale (higher = more preferred), and voters may abstain on some or all candidates.

Usage

```
gen_utilities(
  n_voters,
  n_candidates,
  scale = NULL,
  p_abstain = 0,
  candidate_names = NULL,
  voter_names = NULL,
  seed = NULL
)
```

Arguments

n_voters	Single positive integer. Number of voters (matrix rows).
n_candidates	Single positive integer. Number of candidates (matrix columns).
scale	NULL or a numeric vector c(lo, hi) with lo < hi giving the range of generated utilities. Defaults to NULL (the unit interval c(0, 1)).
p_abstain	Single number in [0, 1]. Probability that a voter abstains on any given candidate, leaving that entry NA. Defaults to 0.
candidate_names	NULL or a character vector of length n_candidates giving the column names. Defaults to NULL (Candidate_1, Candidate_2, ...).
voter_names	NULL or a character vector of length n_voters giving the row names. Defaults to NULL (Voter_1, Voter_2, ...).
seed	NULL or a single numeric value passed to set.seed() for reproducible output. Defaults to NULL.

Value

A numeric matrix with n_voters rows and n_candidates columns of utilities (higher = more preferred); abstained entries are NA. Suitable as the x argument of the voting functions.

See Also

gen_ranks(), gen_approvals()

Examples

```
gen_utilities(n_voters = 5, n_candidates = 3, seed = 1)

# Utilities on a 0-10 scale with occasional abstentions
gen_utilities(n_voters = 5, n_candidates = 3,
              scale = c(0, 10), p_abstain = 0.1, seed = 1)
```

irv *Instant Runoff Voting*

Description

Runs an Instant Runoff Voting. Each voter ranks the candidates, and in each round the candidate with the fewest first-preference votes among the active candidates is eliminated and their ballots transfer to the next-ranked active candidate. The process repeats until a candidate's share of the active (non-exhausted) votes **strictly exceeds** the `majority` threshold or only one candidate remains.

Usage

```
irv(
  x,
  type = c("auto", "rank", "utility", "approval"),
  ties = c("random", "lexicographic", "all"),
  elim_ties = c("random", "lexicographic", "all"),
  majority = 0.5,
  return_history = FALSE
)
```

Arguments

`x`	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank `1` = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named `Candidate_1`, `Candidate_2`, and so on.
`type`	Character string giving the ballot type. One of `"auto"` (the default, detected from `x`), `"rank"`, `"utility"` or `"approval"`.
`ties`	Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of `"random"` (the default; pick one tied candidate at random), `"lexicographic"` (pick the alphabetically first) or `"all"` (return all tied candidates).
`elim_ties`	Character string giving the tie-breaking rule applied at the elimination step when two or more active candidates share the lowest vote count in a round. One of `"random"` (the default; eliminate one tied candidate at random), `"lexicographic"` (eliminate the alphabetically first) or `"all"` (batch elimination: simultaneously remove every candidate tied at the minimum).
`majority`	Single numeric value in `[0, 1]` giving the fraction of active votes a candidate must **strictly exceed** to win a round. Defaults to `0.5`, so a candidate needs more than 50% of the active votes.
`return_history`	Logical. If `TRUE`, the result gains a `$history` element: a character matrix (`n_voters` x `n_rounds`) recording which active candidate received each voter's vote in each round, or `NA` where the ballot was exhausted. Defaults to `FALSE`.

Value

An object of class `"irv_result"`, a list with the elements:

- `summary`: a data frame with columns `candidate`, `vote`, `percentage`, `rank` and `eliminated_in_round`, where `vote` and `percentage` reflect each candidate's score at their last appearance.
- `winners`: character vector of the winning candidate name(s).
- `n_voters`: total number of voters.
- `n_valid`: number of voters with at least one valid preference.
- `n_candidates`: total number of candidates.
- `rounds`: a list with one entry per round (active set, vote counts, percentages, totals, exhausted ballots and candidate(s) eliminated).
- `counts`: candidate-by-round matrix of integer vote counts.
- `percentages`: candidate-by-round matrix of active-vote percentages.
- `method`: a list recording `type`, `ties`, `elim_ties` and `majority`.
- `history`: present only when `return_history = TRUE` (see that argument).

Print the object or call `summary()` on it for a formatted results table.

See Also

`fptp()`, `tworound()`, `borda()`

Examples

```
ballots <- gen_ranks(n_voters = 40, n_candidates = 4, seed = 1)
result <- irv(ballots)
result

# Deterministic tie-breaking and round-by-round history
irv(ballots, ties = "lexicographic", elim_ties = "lexicographic",
    return_history = TRUE)
```

---

majority_judgement *Majority Judgment*

Description

Runs the Majority Judgment voting. Each voter assigns every candidate an absolute grade on a common ordinal scale (1 = worst, K = best), and for each candidate the **median grade** across voters is computed (using the lower of the two middle grades when the count is even). The candidate with the highest median wins. Ties on the median are resolved either by the majority-gauge proportions or by the iterative "majority sequence" rule, which repeatedly removes the median grade until the candidates' sequences diverge.

Usage

```
majority_judgement(
  x,
  type = c("auto", "rank", "utility", "approval", "score"),
  num_categories = NULL,
  median_tie_break = c("gauge", "iterative"),
  ties = c("random", "lexicographic", "all"),
  return_grades = FALSE
)
```

Arguments

x	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank 1 = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named Candidate_1, Candidate_2, and so on.
type	Character string giving the ballot type. One of "auto" (the default, detected from x), "rank", "utility", "approval" or "score". The "score" type treats each entry directly as an integer grade.
num_categories	Integer >= 2 giving the number of grade levels K, or NULL (the default) to choose a sensible value from the inferred type: 2 for approval, the number of candidates for rank, min(5, max(2, n_candidates)) for utility, and the maximum observed grade for score. Approval is always coerced to 2, rank is capped at the number of candidates (both with a warning), and a score K below the maximum observed grade is an error.
median_tie_break	Character string giving the rule used to separate candidates that share the same median grade. One of "gauge" (the default; compares the majority-gauge proportions of voters above and below the median) or "iterative" (compares the majority sequences obtained by repeatedly removing the median grade).
ties	Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of "random" (the default; pick one tied candidate at random), "lexicographic" (pick the alphabetically first) or "all" (return all tied candidates).
return_grades	Logical. If TRUE, the internal integer grade matrix (voters by candidates, on the 1..K scale) is attached to the result as $grades. Defaults to FALSE.

Value

An object of class "majority_judgement_result": a list with elements summary (a data frame with columns candidate, median, p_above, p_below and rank), winners, n_voters, n_valid, n_candidates, grade_distribution and method (itself a list of type, num_categories, median_tie_break and ties), plus grades when return_grades = TRUE. Print the object or call summary() on it for a formatted results table.

See Also

borda(), approval(), condorcet()

Examples

```
utilities <- gen_utilities(n_voters = 30, n_candidates = 4, seed = 1)
majority_judgement(utilities)

majority_judgement(utilities, num_categories = 5, median_tie_break = "iterative")
```

---

tworound *Two-Round System (runoff) election*

Description

Runs a Two-Round System election. In the first round each voter's top preference is counted; if a candidate wins more than 50% of the valid votes they are elected outright. Otherwise the top two candidates advance to a second round, where each voter's preference between the two finalists is counted and the finalist with the most votes wins. The majority threshold is strictly greater than 50%.

Usage

```
tworound(
  x,
  type = c("auto", "rank", "utility", "approval"),
  ties = c("random", "lexicographic", "all"),
  return_ranks = FALSE
)
```

Arguments

x	A matrix of ballots with one row per voter and one column per candidate. Accepts a ranking matrix (rank 1 = most preferred), a utility (score) matrix (higher = more preferred), or a binary/logical approval matrix. Column names are used as candidate names; if absent, candidates are named Candidate_1, Candidate_2, and so on.
type	Character string giving the ballot type. One of "auto" (the default, detected from x), "rank", "utility" or "approval".
ties	Character string giving the tie-breaking rule applied when several candidates are tied for a winning position. One of "random" (the default; pick one tied candidate at random), "lexicographic" (pick the alphabetically first) or "all" (return all tied candidates).
return_ranks	Logical. If TRUE, the ranking matrix derived internally is attached to the result as $ranks. Defaults to FALSE.

Value

An object of class "tworound_result", a list with elements: round1 (a list with the first-round summary data frame and n_valid vote count), round2 (a list with the second-round summary and n_valid, or NULL when the election is decided in round 1), finalists (the candidate name(s) advancing to round 2, or character(0)), winners (the winning candidate name(s)), n_voters, n_candidates, and method (a list recording type, ties, and tworound, a logical flag for whether a second round was held). If return_ranks = TRUE, the derived ranking matrix is also attached as $ranks. Print the object or call summary() on it for a formatted results table.

See Also

fptp(), irv(), borda()

Examples

```
ranks <- gen_ranks(n_voters = 30, n_candidates = 4, seed = 1)
tworound(ranks)

utilities <- gen_utilities(n_voters = 30, n_candidates = 4, seed = 1)
tworound(utilities, type = "utility", ties = "lexicographic")
```

Index

approval, 2
approval(), *4*, *6, 7*, *9*, *18*

borda, 3
borda(), *3*, *6*, *9*, *11*, *16*, *18, 19*

condorcet, 5
condorcet(), *4*, *11*, *18*

d21, 6
d21(), *9*
d21_minus, 8
d21_minus(), *7*

fptp, 10
fptp(), *3, 4*, *7*, *16*, *19*

gen_approvals, 11
gen_approvals(), *13, 14*
gen_ranks, 12
gen_ranks(), *12*, *14*
gen_utilities, 13
gen_utilities(), *12, 13*

irv, 15
irv(), *6*, *11*, *19*

majority_judgement, 16

set.seed(), *12–14*
summary(), *3, 4*, *6, 7*, *9*, *11*, *16, 17*, *19*

tworound, 18
tworound(), *3*, *16*