Package ‘SafeVote’

January 20, 2025

Type Package

Title Election Vote Counting with Safety Features

Version 1.0.1

Date 2024-10-04

Description Fork of 'vote_2.3-2', Raftery et al. (2021) <DOI:10.32614/RJ-2021-086>, with additional support for stochastic experimentation.

Depends R (>= 3.5.0)

Imports formattable, knitr, fields, grDevices, graphics, utils, ggplot2, data.table, stringr, forcats, dplyr

Encoding UTF-8

License GPL (>= 2)

Language EN-GB

NeedsCompilation no

RoxygenNote 7.3.2

LazyData true

URL `https://cran.r-project.org/package=SafeVote`, `https://cthombor.github.io/SafeVote/`

Suggests testthat (>= 3.0.0), vote, STV

Config/testthat/edition 3

Author Clark Thomborson [cre, aut] (<`https://orcid.org/0000-0002-4147-7898`>)

Maintainer Clark Thomborson <`c.thomborson@auckland.ac.nz`>

Repository CRAN

Date/Publication 2024-10-05 22:50:03 UTC

Contents

`.print.summary.SafeVote` *.print method for summary object*

Description

.print method for summary object

Usage

```
.print.summary.SafeVote(x, ...)
```

Arguments

`x, ...`	undocumented

Value

undocumented

`.summary.SafeVote` *summarises vote-totals for subsequent printing*

Description

summarises vote-totals for subsequent printing

Usage

```
.summary.SafeVote(object, larger.wins = TRUE, reorder = TRUE)
```

Arguments

`object`	vector of total votes per candidate
`larger.wins`	TRUE if candidates are "voted in" rather than voted-out
`reorder`	TRUE if output data.frame columns should be in rank-order

Value

a data.frame with three columns and nc+1 rows, where nc is the number of candidates. The first column contains candidate names and a final entry named "Sum". The second column contains vote totals. The third column is a vector of chars which indicate whether the candidate has been elected. The data.frame has four named attributes carrying election parameters.

TODO: refactor into a modern dialect of R, perhaps by defining a constructor for an election_info S3 object with a summary method and a print method

`a3_hil` *Tideman a3_hil*

Description

This data is one of 87 sets of ballots from the Tideman data collection, as curated by The Center for Range Voting.

This set of ballots was collected in 1987 by Nicolaus Tideman, with support from NSF grant SES86-18328. "The data are records of ballots from elections of British organizations (mostly trade unions using PR-STV or IRV voting) in which the voters ranked the candidates. The data were gathered under a stipulation that the organizations involved would remain anonymous."

The ballots were encoded in David Hill's format, and have been converted to the preference-vector format of this package. The archival file A4.HIL at rangevoting.org contains eight blank ballot papers (1, 616, 619, 620, 685, 686, 687, 688) which we have retained. This set may be counted by 'stv(a3_hil,nseats=attr(a3_hil,"nseats"))'.

Usage

```
data(a3_hil)
```

Format

A data frame with attribute "nseats" = 7, consisting of 989 observations and 15 candidates.

`a4_hil` *Tideman a4_hil*

Description

This data is one of 87 sets of ballots from the Tideman data collection, as curated by The Center for Range Voting. The ballots were archived in David Hill's format, and have been converted to the preference-vector format of this package.

This set of ballots was collected in 1987 by Nicolaus Tideman, with support from NSF grant SES86-18328. "The data are records of ballots from elections of British organizations (mostly trade unions using PR-STV or IRV voting) in which the voters ranked the candidates. The data were gathered under a stipulation that the organizations involved would remain anonymous."

Usage

```
data(a4_hil)
```

Format

A data frame with attribute "nseats" = 2, consisting of 43 observations and 14 candidates.

`a53_hil` *Tideman a53_hil*

Description

This data is one of 87 sets of ballots from the Tideman data collection, as curated by The Center for Range Voting.

This set of ballots was collected in 1988 by Nicolaus Tideman, with support from NSF grant SES86-18328. "The data are records of ballots from elections of British organizations (mostly trade unions using PR-STV or IRV voting) in which the voters ranked the candidates. The data were gathered under a stipulation that the organizations involved would remain anonymous."

The ballots were encoded in David Hill's format, and have been converted to the preference-vector format of this package. Candidates have been renamed to letters of the alphabet, for ease of comparison with Table 3 of Tideman (2000). Note: the DOI for this article is 10.1023/A:1005082925477, with an embedded colon which isn't handled by the usual DOI-to-URL conversions.

As noted in this table, it is a very close race between candidates D, F, and B in the final rounds of a Meek count of 'a53_hil'.

Tideman's implementation of Meek's method excludes B (on 59.02 votes), then elects D in the final round (on 88.33 votes) with a margin of 0.95 votes ahead of F (on 87.38 votes).

In v1.0, 'stv(a53.hil,quota.hare=TRUE)' excludes F (on 56.418 votes), then elects D in the final round (on 79.705 votes) with a winning margin of 0.747 votes ahead of B (on 78.958 votes). The result of the election is the same but the vote counts and winning margins differ significantly; so we conclude that 'stv(quota.hare=TRUE)' in SafeVote v1.0 is *not* a reliable proxy for Tideman's implementation of Meek's algorithm.

Future researchers may wish to adjust the quota calculation of 'vote.stv()' so that it is no longer biased upward by a "fuzz" of 0.001, to see if this change significantly reduces the discrepancies with Tideman's implementation of Meek.

It would be unreasonable to expect an exact replication of results from two different implementations of an STV method. We leave it to future researchers to develop a formal specification, so that it would be possible to verify the correctness of an implementation. We also leave it to future researchers to develop a set of test cases with appropriate levels of tolerance for the vagaries of floating-point roundoff in optimised (or even unoptimised!) compilations of the same code on different computing systems. We suggest that 'a53_hil' be included in any such test set.

We note in passing that B.A. Wichmann, in "Checking two STV programs", Voting Matters 11, 2000, discussed the cross-validation exercise he conducted between the ERBS implementation of its voting rules and the Church of England's implementation of its voting rules. In both cases, he discovered ambiguities in the specification as well as defects in the implementation.

Usage

```
data(a53_hil)
```

Format

A data frame with attribute "nseats" = 4, consisting of 460 observations and 10 candidates.

`approval` *Count votes using the approval method*

Description

See https://arxiv.org/abs/2102.05801

Usage

```
approval(votes, nseats = 1, fsep = "\t", quiet = FALSE, ...)
```

Arguments

`votes`, `nseats`, `fsep`, `quiet`, `...`	undocumented

Value

undocumented

`as.SafeRankExpt` *as.SafeRankExpt()*

Description

as.SafeRankExpt()

Usage

```
as.SafeRankExpt(df)
```

Arguments

`df`	data.frame object

Value

a SafeRankExpt object, or stop() if df fails some sanity checks

`assemble.args.for.check.score` *undocumented internal method*

Description

undocumented internal method

Usage

```
assemble.args.for.check.score(x, max.score = NULL, ...)
```

Arguments

`x`, `max.score`, `...`	undocumented

Value

undocumented

`assemble.args.for.check.stv` *undocumented internal method*

Description

undocumented internal method

Usage

```
assemble.args.for.check.stv(x, equal.ranking = FALSE, ...)
```

Arguments

`x`, `equal.ranking`, `...`	undocumented

Value

undocumented

`backwards.tiebreak` *Undocumented internal method*

Description

Undocumented internal method

Usage

```
backwards.tiebreak(prefs, icans, elim = TRUE)
```

Arguments

`prefs`	undocumented
`icans`	undocumented
`elim`	undocumented

`check.nseats` *parameter-checking method for nseats (internal)*

Description

parameter-checking method for nseats (internal)

Usage

```
check.nseats(
  nseats = NULL,
  ncandidates,
  default = 1,
  mcan = NULL,
  complete.ranking = FALSE
)
```

Arguments

`nseats`	initially-specified number of seats to be filled in an election
`ncandidates`	the number of candidates standing for election
`default`	the return value of this function when nseats=NULL
`mcan`	a deprecated name for nseats
`complete.ranking`	when TRUE, the return value is in 1..ncandidates When FALSE, the return value is in 1..ncandidates-1 (for backwards compatibility)

Value

a valid non-NULL value for the number of seats to be filled

`check.ranking` *check the validity of a partial ranking*

Description

check the validity of a partial ranking

Usage

```
check.ranking(r)
```

Arguments

`r`	a numeric vector

Value

a partial ranking of the elements of ’r’, using ’ties.method="min"’

`check.votes` *undocumented internal method*

Description

undocumented internal method

Usage

```
check.votes(x, ..., quiet = FALSE)
```

Arguments

`x`, `quiet`, `...`	undocumented

Value

undocumented

`check.votes.approval` *undocumented internal method*

Description

undocumented internal method

Usage

```
check.votes.approval(record, ...)
```

Arguments

`record, ...`	undocumented

Value

undocumented

`check.votes.condorcet` *undocumented internal method*

Description

undocumented internal method

Usage

```
check.votes.condorcet(record, ...)
```

Arguments

`record, ...`	undocumented

Value

undocumented

`check.votes.plurality` *undocumented internal method*

Description

undocumented internal method

Usage

```
check.votes.plurality(record, ...)
```

Arguments

`record`, `...`	undocumented

Value

undocumented

`check.votes.score` *undocumented internal method*

Description

undocumented internal method

Usage

```
check.votes.score(record, max.score, ...)
```

Arguments

`record`, `max.score`, `...`	undocumented

Value

undocumented

`check.votes.stv` *undocumented internal method*

Description

undocumented internal method

Usage

```
check.votes.stv(record, equal.ranking = FALSE, ...)
```

Arguments

`record`, `equal.ranking`, `...`	undocumented

Value

undocumented

`check.votes.tworound.runoff` *undocumented internal method*

Description

undocumented internal method

Usage

```
check.votes.tworound.runoff(record, ...)
```

Arguments

`record`, `...`	undocumented

Value

undocumented

`combineRankings` *the least upper bound on a pair of rankings*

Description

the least upper bound on a pair of rankings

Usage

```
combineRankings(r1, r2)
```

Arguments

`r1, r2`	numeric vectors

Value

the most complete (but possibly partial) ranking which is consistent with both r1 and r2. Uses 'ties.method="min"'

Examples

```
combineRankings( c(3,1,2), c(2,1,3) )
```

`completeRankingTable` *internal method to analyse the partial results of an stv() ballot count, to discover a complete ranking of all candidates. The ranking may depend on the value of nseats, because this affects how votes are transferred.*

Description

internal method to analyse the partial results of an stv() ballot count, to discover a complete ranking of all candidates. The ranking may depend on the value of nseats, because this affects how votes are transferred.

Usage

```
completeRankingTable(object, quiet, verbose)
```

Arguments

`object`	partial results
`quiet`	TRUE to suppress console output
`verbose`	TRUE to produce diagnostic output

Value

data.frame with columns TotalRank, Margin, Candidate, Elected, SafeRank

---

`condorcet` *Count votes using the Condorcet voting method.*

Description

The Condorcet method elects the candidate who wins a majority of the ranked vote in every head to head election against each of the other candidates. A Condorcet winner is a candidate who beats all other candidates in pairwise comparisons. Analogously, a Condorcet loser is a candidate who loses against all other candidates. Neither Condorcet winner nor loser might exist.

Usage

```
condorcet(
  votes,
  runoff = FALSE,
  nseats = 1,
  safety = 1,
  fsep = "\t",
  quiet = FALSE,
  ...
)
```

Arguments

`votes`	A matrix or data.frame containing the votes. Rows correspond to the votes, columns correspond to the candidates. If 'votes' is a character string, it is interpreted as a file name from which the votes are to be read. See below.
`runoff`	Logical. If TRUE and no Condorcet winner exists, the election goes into a run-off, see below.
`nseats`	the number of seats to be filled in this election
`safety`	Parameter for a clustering heuristic on a total ranking of the candidates. Conjecture: the default of '1.0' ensures a separation of one s.d. between clusters, when 'votes' are i.u.d. permutations on the candidates.
`fsep`	If 'votes' is a file name, this argument gives the column separator in the file.
`quiet`	If TRUE no output is printed.
`...`	Undocumented intent (preserved from legacy code)

Details

If the runoff argument is set to 'TRUE' and no Condorcet winner exists, two or more candidates with the most pairwise wins are selected and the method is applied to such subset. If more than two candidates are in such run-off, the selection is performed repeatedly, until either a winner is selected or no more selection is possible.

The input data votes is structured the same way as for the stv method: Row 'i' contains the preferences of voter 'i' numbered '1; 2; : : : ; r; 0; 0; 0; 0', in some order, while equal preferences are allowed. The columns correspond to the candidates. The dimnames of the columns are the names of the candidates; if these are not supplied then the candidates are lettered 'A, B, C, ...'. If the dataset contains missing values (NA), they are replaced by zeros.

If a ballot has equally-ranked candidates, its rankings are tested for validity: for each preference i which does not have any duplicate, there are exactly $i - 1$ preferences j with $0 < j < i$. If any ballot 'x' fails this validity test, it is automatically corrected (aka "converted") into a valid ballot using 'x <- rank(x, ties.method = "min")', and a warning is issued.

This method also computes a Borda ranking of all candidates, using tournament-style scoring. This ranking is "fuzzed" into a 'safeRank', with approximately 1 s.d. of fuzz when 'safety=1.0' and voter preferences are i.u.d. A warning is thrown if a 'safeRank' violates the (extended) Condorcet principle: that Candidate i is more highly ranked than Candidate j only if a majority of voters agree with this.

Value

Object of class 'SafeVote.condorcet'

Examples

```
{
data(food_election)
condorcet(food_election)
}
```

`correct.ranking` *Amend ballots with equal or incomplete preferences*

Description

The 'correct.ranking' function returns a modified set of ballots. Its argument 'partial' determines if ballots are partially set to '0' ('TRUE'), or if it is a complete re-ranking, as allowed when 'equal.ranking = TRUE'. It can be used by calling it explicitly. It is called by 'stv' if 'equal.ranking = TRUE' or 'invalid.partial = TRUE'. It is also called from within the 'condorcet' function with the default value ('FALSE') for 'partial', i.e. interpreting any '0' as a last= preference.

Usage

```
correct.ranking(votes, partial = FALSE, quiet = FALSE)
```

Arguments

`votes`	original contents of ballot box
`partial`	if 'FALSE' (default), each ballot is interpreted, if possible, as a complete (but not necessarily total) ranking of the candidates. If 'TRUE', a ballot will contain a '0' on unranked candidates.
`quiet`	suppress diagnostics

Value

corrected ballots

`dublin_west` *Dublin West*

Description

Dataset containing ranked votes for the Dublin West constituency in 2002, Ireland.

Usage

```
data(dublin_west)
```

Format

A data frame with 29988 observations and 9 candidates. Each record corresponds to one ballot with candidates being ranked between 1 and 9 with zeros allowed.

See Also

Wikipedia

`election.info` *prints the basic results of an election*

Description

prints the basic results of an election

Usage

```
election.info(x)
```

Arguments

`x`	basic election results, as named attributes of an R structure or object

Value

data.frame : an invisible copy of the printed results

TODO: refactor into a modern dialect of R, e.g. defining a constructor for an election_info S3 object with a print method

---

`extractMargins` *extract margins from the results of a ballot count*

Description

extract margins from the results of a ballot count

Usage

```
extractMargins(marginNames, crRanks, cr)
```

Arguments

`marginNames`	list of colnames of the margins in our SafeRank result
`crRanks`	ranks of candidates, not necessarily total
`cr`	structure returned by a ballot-counting method Margins are adjusted for tied candidates, such that candidates within a tie group have margins indicative of their relative strengths. Extremely small margins are indicative of floating-point roundoff errors.

Value

named list of margins

`extractRank` *Extract a ranking vector by name from the results of a ballot count*

Description

Extract a ranking vector by name from the results of a ballot count

Usage

```
extractRank(rankMethod, cr)
```

Arguments

`rankMethod`	"safeRank", "elected", or "rank"
`cr`	structure returned by a ballot-counting method

Value

a numeric ranking vector, in order of colnames(cr$data)

`food_election` *Food Election*

Description

Sample data for testing SafeVote

Usage

```
data(food_election)
```

Format

A data frame with 20 observations and 5 candidates (Oranges, Pears, Chocolate, Strawberries, Sweets). Each record corresponds to one ballot with ranking for each of the candidates.

`forwards.tiebreak` *Undocumented internal method*

Description

Undocumented internal method

Usage

```
forwards.tiebreak(prefs, icans, elim = TRUE)
```

Arguments

`prefs`	undocumented
`icans`	undocumented
`elim`	undocumented

image.SafeVote.condorcet

The image function visualizes the joint distribution of two preferences (if 'all.pref=FALSE') given 'xpref' and 'ypref', as well as the marginal distribution of all preferences (if 'all.pref=TRUE'). The joint distribution can be shown as proportions (if 'proportion=TRUE') or raw vote counts (if 'proportion=FALSE').

Description

The image function visualizes the joint distribution of two preferences (if 'all.pref=FALSE') given 'xpref' and 'ypref', as well as the marginal distribution of all preferences (if 'all.pref=TRUE'). The joint distribution can be shown as proportions (if 'proportion=TRUE') or raw vote counts (if 'proportion=FALSE').

Usage

```
## S3 method for class 'SafeVote.condorcet'
image(x, ...)
```

Arguments

`x`	object of type SafeVote.condorcet
`...`	See arguments for image.SafeVote.stv, especially 'xpref', 'ypref', 'all.pref' and 'proportion'.

Value

image object, with side-effect in RStudio Plots pane

image.SafeVote.stv

visualisation of joint and marginal distributions in STV preferences

Description

visualisation of joint and marginal distributions in STV preferences

Usage

```
## S3 method for class 'SafeVote.stv'
image(x, xpref = 2, ypref = 1, all.pref = FALSE, proportion = TRUE, ...)
```

Arguments

`x`	STV results to be visualised
`xpref`, `ypref`	candidates shown in a joint distribution plot
`all.pref`	plot the joint distribution of two preferences (if 'all.pref=FALSE') or the marginal distribution of all preferences (if 'all.pref=TRUE').
`proportion`	The joint distribution can be shown either as proportions (if 'proportion=TRUE') or raw vote counts (if 'proportion=FALSE').
`...`	args passed to fields::image.plot()

Value

image object, with side-effect in RStudio Plots pane

`ims_approval` *IMS Approval*

Description

Modified version of ims_election, for use in approval voting.

Usage

```
data(ims_approval)
```

Format

A data frame with 620 observations and 10 candidates (names were made up). Each record corresponds to one ballot, with 0 indicating disapproval of a candidate and 1 indicating approval.

`ims_election` *IMS Election*

Description

Datasets containing anonymized votes for a past Council election of the Institute of Mathematical Statistics (IMS). The dataset ims_election is the original dataset used with single transferable vote, where candidate names have been changed.

Usage

```
data(ims_election)
```

Format

A data frame with 620 observations and 10 candidates (names were made up). Each record corresponds to one ballot. The IMS Council voting is done using the STV method, and thus the ims_election dataset contains ballots with candidates being ranked between 1 and 10 with zeros allowed.

`ims_plurality` *IMS Plurality*

Description

Modified version of ims_election, for use in plurality voting.

Usage

```
data(ims_plurality)
```

Format

A data frame with 620 observations and 10 candidates (names were made up). Each record corresponds to one ballot, with 1 against the voter's most-preferred candidate and 0 against all other candidates.

`ims_score` *IMS Score*

Description

Modified version of ims_election, for use in score voting.

Usage

```
data(ims_score)
```

Format

A data frame with 620 observations and 10 candidates (names were made up). Each record corresponds to one ballot, with higher values indicating the more-preferred candidates.

`ims_stv` *IMS STV*

Description

Copy of ims_election, included for backwards compatibility.

Usage

```
data(ims_election)
```

Format

A data frame with 620 observations and 10 candidates (names were made up). Each record corresponds to one ballot. The IMS Council voting is done using the STV method, and thus the ims_election dataset contains ballots with candidates being ranked between 1 and 10 with zeros allowed.

`invalid.votes` *Extracts the invalid.votes member (if any) from the result of a count*

Description

This method was added Jan 2022 – it was named in a warning message but had apparently either never been implemented, or had been "lost" through versioning.

Usage

```
invalid.votes(x)
```

Arguments

`x`	value returned by stv, condorcet, approval, plurality, or score

Value

matrix with one column per candidate and one row per invalid ballot `is.SafeRankExpt` *is.SafeRankExpt()*

Description

is.SafeRankExpt()

Usage

```
is.SafeRankExpt(x)
```

Arguments

`x`	object of unknown class

Value

TRUE if x is a valid SafeRankExpt object

`is.valid.vote` *undocumented internal method*

Description

undocumented internal method

Usage

```
is.valid.vote(x, method, ...)
```

Arguments

`x`, `method`, `...`	undocumented

Value

undocumented

loserMargin — *Find a loser and their margin of victory*

Description

Find a loser and their margin of victory

Usage

```
loserMargin(votes)
```

Arguments

`votes`	cleaned ballots

Value

length-2 vector: the index of a losing candidate, and their margin of loss (0 if a tie, NA if no winners)

new_SafeRankExpt — *Constructor for the results of a SafeRank experiment*

Description

Constructor for the results of a SafeRank experiment

Usage

```
new_SafeRankExpt(
  rankNames = list(),
  marginNames = list(),
  countMethod = character(0),
  rankMethod = character(0),
  datasetName = character(0),
  experimentalMethod = character(0),
  countArgs = list(),
  nseats = integer(0),
  otherFactors = list(),
  unitFactors = list()
)
```

Arguments

`rankNames`	colnames for per-candidate ranks
`marginNames`	colnames for per-candidate margins
`countMethod`	secondary factor: counting method e.g. "stv"
`rankMethod`	secondary factor: ranking method e.g. "elected"
`datasetName`	secondary factor: name of the dataset of ballots
`experimentalMethod`	secondary factor: name of the method which simulated these elections e.g. "testFraction"
`countArgs`	secondary factor: args passed to countMethod
`nseats`	secondary factor: number of seats to be filled
`otherFactors`	other secondary factors, e.g. parameters to experimentalMethod
`unitFactors`	per-unit factors derived from PRNG of the experimental harness, e.g describing the ballots randomly deleted during testDeletions

Value

object of class SafeRankExpt

`ordered.preferences` *Undocumented internal method*

Description

Undocumented internal method

Usage

```
ordered.preferences(vmat)
```

Arguments

`vmat`	undocumented

`ordered.tiebreak` *Undocumented internal method*

Description

Undocumented internal method

Usage

```
ordered.tiebreak(vmat, seed = NULL)
```

Arguments

`vmat`	undocumented
`seed`	undocumented

`plot.SafeRankExpt` *plot() method for the result of an experiment with varying numbers of ballots*

Description

The "adjusted rank" of a candidate is their ranking r plus their scaled "winning margin". The scaled margin is $e^{-cx/\sqrt{n}}$, where x is the adjusted margin (i.e. the number of votes by which this candidate is ahead of the next-weaker candidate, adjusted for the number of ballots n and the number of seats s), and $c > 0$ is the margin-scaling parameter 'cMargin'.

Usage

```
## S3 method for class 'SafeRankExpt'
plot(
  x,
  facetWrap = FALSE,
  nResults = NA,
  anBallots = 0,
  cMargin = 1,
  xlab = "Ballots",
  ylab = "Adjusted Rank",
  title = NULL,
  subtitle = "(default)",
  line = TRUE,
  boxPlot = FALSE,
  boxPlotCutInterval = 10,
  pointSize = 1,
  ...
)
```

Arguments

`x`	object containing experimental results
`facetWrap`	TRUE provides per-candidate scatterplots
`nResults`	number of candidates whose results are plotted (omitting the least-favoured candidates first)
`anBallots, cMargin`	parameters in the rank-adjustment formula
`xlab, ylab`	axis labels
`title`	overall title for the plot. Default: NULL
`subtitle`	subtitle for the plot. Default: value of nSeats and any non-zero rank-adjustment parameters
`line`	TRUE will connect points with lines, and will disable jitter
`boxPlot`	TRUE for a boxplot, rather than the default xy-scatter
`boxPlotCutInterval`	parameter of boxplot, default 10
`pointSize`	diameter of points
`...`	params for generic plot()

Details

The default value of 'cMargin=1.0' draws visual attention to candidates with a very small winning margin, as their adjusted rank is very near to $r + 1$. Candidates with anything more than a small winning margin have only a small rank adjustment, due to the exponential scaling.

A scaling linear in s/n is applied to margins when 'anBallots>0'. Such a linear scaling may be a helpful way to visualise the winning margins in STV elections because the margin of victory for an elected candidate is typically not much larger than the quota of $n/(s + 1)$ (Droop) or n/s (Hare). The linear scaling factor is as/n, where a is the value of 'anBallots', s is the number of seats, and n is the number of ballots. For plotting on the (inverted) adjusted rank scale, the linearly-scaled margin is added to the candidate's rank. Note that the linearly-scaled margins are zero when $a = 0$, and thus have no effect on the adjusted rank. You might want to increase the value of 'anBallots', starting from 1.0, until the winning candidate's adjusted rank is 1.0 when all ballots are counted, then confirm that the adjusted ranks of other candidates are still congruent with their ranking (i.e. that the rank-adjustment is less than 1 in all cases except perhaps on an initial transient with small numbers of ballots).

When both 'anBallots' and 'cMargins' are non-zero, the ranks are adjusted with both exponentially-scaled margins and linearly-scaled margins. The resulting plot would be difficult to interpret in a valid way.

Todo: Accept a list of SafeVoteExpt objects.

Todo: Multiple counts with the same number of ballots could be summarised with a box-and-whisker graphic, rather than a set of jittered points.

Todo: Consider developing a linear scaling that is appropriate for plotting stochastic experimental data derived from Condorcet elections.

Value

graphics object, with side-effect in RStudio Plots pane

`plot.SafeVote.stv` *plot() method for the result of an stv() ballot-count*

Description

The 'plot' function shows the evolution of the total score for each candidate as well as the quota.

Usage

```
## S3 method for class 'SafeVote.stv'
plot(x, xlab = "Count", ylab = "Preferences", point.size = 2, ...)
```

Arguments

`x`	stv results
`xlab`, `ylab`	axis labels
`point.size`	diameter of elected/eliminated points
`...`	params for generic plot()

Value

graphics object, with side-effect in RStudio's Plots pane

`plurality` *Count votes using the plurality method*

Description

See https://arxiv.org/abs/2102.05801

Usage

```
plurality(votes, nseats = 1, fsep = "\t", quiet = FALSE, ...)
```

Arguments

`votes`, `nseats`, `fsep`, `quiet`, `...`	undocumented

Value

undocumented

prepare.votes *Coerce input ’data’ into a matrix*

Description

Coerce input ’data’ into a matrix

Usage

```
prepare.votes(data, fsep = "\n")
```

Arguments

`data`	possibly a .csv file, possibly an R object
`fsep`	separation character for .csv e.g. tab or comma

Value

a matrix with one row per ballot, one column per candidate, with named rows and columns

print.summary.SafeRankExpt *Print method for summary.SafeRankExpt*

Description

Print method for summary.SafeRankExpt

Usage

```
## S3 method for class 'summary.SafeRankExpt'
print(x, ...)
```

Arguments

`x`	experimental results
`...`	args for generic print()

Value

invisible(x), with side-effects to console

print.summary.SafeVote.approval
print method for summary object

Description

print method for summary object

Usage

```
## S3 method for class 'summary.SafeVote.approval'
print(x, ...)
```

Arguments

`x, ...`	undocumented

Value

undocumented

print.summary.SafeVote.condorcet
print method for summary.SafeVote.condorcet

Description

print method for summary.SafeVote.condorcet

Usage

```
## S3 method for class 'summary.SafeVote.condorcet'
print(x, ...)
```

Arguments

`x`	object of type summary.SafeVote.condorcet
`...`	parameters passed to generic print

Value

textual description of 'x'

print.summary.SafeVote.plurality

print method for summary of plurality object

Description

print method for summary of plurality object

Usage

```
## S3 method for class 'summary.SafeVote.plurality'
print(x, ...)
```

Arguments

`x, ...`	undocumented

Value

undocumented

print.summary.SafeVote.score

print method for summary.score object

Description

print method for summary.score object

Usage

```
## S3 method for class 'summary.SafeVote.score'
print(x, ...)
```

Arguments

`x, ...`	undocumented

Value

undocumented

`print.summary.SafeVote.stv` *print() method for a summary() of a SafeVote result*

Description

print() method for a summary() of a SafeVote result

Usage

```
## S3 method for class 'summary.SafeVote.stv'
print(x, ...)
```

Arguments

`x`	election results
`...`	args to be passed to kable()

Value

no return value, called for side-effect of printing to console

`rbind_SafeRankExpt` *add a row to a SafeRankExpt object*

Description

add a row to a SafeRankExpt object

Usage

```
rbind_SafeRankExpt(object, row)
```

Arguments

`object`	prior results of experimentation
`row`	new observations

Value

SafeRankExpt object with an additional row

`readHil` *read a set of ballots in .HIL format*

Description

rangevoting.org/TidemanData.html: The data are in a format developed by David Hill. The first line contains the number of candidates and the number to be elected. (Many but not all elections were multi-winner.) In subsequent lines that represent ballot papers, the first number is always 1. (The format was designed for a counting program that treats the first number as the number of instances of the ordering of the candidates on the line.) Next on these lines is a sequence of numbers representing a voter's reported ranking: The number of the candidate ranked first, the number of the candidate ranked second, and so on. The end of the reported ranking is signaled by a zero. A zero at the beginning of the ranking is a signal that the list of ballot papers has ended. Next come the names of the candidates, each in parentheses, as required by the counting program, and finally the name of the election.

Usage

```
readHil(filnm, quiet = FALSE)
```

Arguments

`filnm`	name of a file in .HIL format
`quiet`	suppress diagnostic output

Value

a matrix with one row per ballot, one column per candidate, with named rows and columns, and with attributes "nseats" and "ename"

`remove.candidate` *Remove a candidate, amending ballot papers as required*

Description

Remove a candidate, amending ballot papers as required

Usage

```
remove.candidate(votes, can, quiet = TRUE)
```

Arguments

`votes`	ballot box
`can`	candidate to be removed
`quiet`	suppress diagnostics

Value

amended ballot box

`score` *Count votes using the score (or range) method.*

Description

See https://arxiv.org/abs/2102.05801

Usage

```
score(
  votes,
  nseats = 1,
  max.score = NULL,
  larger.wins = TRUE,
  fsep = "\t",
  quiet = FALSE,
  ...
)
```

Arguments

`votes`, `nseats`, `max.score`, `larger.wins`, `fsep`, `quiet`, `...`	undocumented

Value

undocumented

`solveTiebreak` *Undocumented internal method, renamed from 'solve.tiebreak' to avoid confusion with generic solve()*

Description

Undocumented internal method, renamed from 'solve.tiebreak' to avoid confusion with generic solve()

Usage

```
solveTiebreak(method, prefs, icans, ordered.ranking = NULL, elim = TRUE)
```

Arguments

`method`	undocumented
`prefs`	undocumented
`icans`	undocumented
`ordered.ranking`	undocumented
`elim`	undocumented

Value

undocumented

`stv` *Count preferential ballots using an STV method*

Description

The 'votes' parameter is as described in `condorcet()` with the following additional semantics.

Usage

```
stv(
  votes,
  nseats = NULL,
  eps = 0.001,
  equal.ranking = FALSE,
  fsep = "\t",
  ties = c("f", "b"),
  quota.hare = FALSE,
  constant.quota = FALSE,
  win.by.elim = TRUE,
  group.nseats = NULL,
  group.members = NULL,
  complete.ranking = FALSE,
  invalid.partial = FALSE,
  verbose = FALSE,
  seed = NULL,
  quiet = FALSE,
  digits = 3,
  backwards.compatible = FALSE,
  safety = 1,
  ...
)
```

Arguments

`votes`	an array with one column per candidate and one row per ballot, as described in `condorcet()`
`nseats`	the number of seats to be filled in this election
`eps`	fuzz-factor when comparing fractional votes. The default of 0.001 is preserved from the legacy code, injecting substantial validity hazards into the codebase. We have not attempted to mitigate any of these hazards in 'SafeVote v1.0.0'. We prefer instead to retain backwards-compatibility with the legacy code in 'vote_2.3-2' in the knowledge that, even if these hazards were adequately addressed, the resulting code is unlikely to be reliable at replicating the results of any other implementation of any of the many variants of "STV" counting methods. Please see the description of the 'a53_hil' dataset in this package for some preliminary findings on the magnitude of the vote-count-variances which may be injected by differing implementations of broadly-similar "STV" counting methods.
`equal.ranking`	if 'TRUE', equal preferences are allowed.
`fsep`	column-separator for output
`ties`	vector of tie-breaking methods: "f" for forward, "b" for backward
`quota.hare`	'TRUE' if Hare quota, 'FALSE' if Droop quota (default)
`constant.quota`	'TRUE' if quota is held constant. Over-rides 'quota.hare'. Default is 'FALSE'
`win.by.elim`	'TRUE' (default) if the quota is waived when there are no more candidates than vacant seats. Note: there is no lower limit when the quota is waived, so a candidate may be elected on zero votes.
`group.nseats`	number of seats reserved to members of a group
`group.members`	vector of members of the group with reserved seats
`complete.ranking`	is 'TRUE' by default. This parameter is retained solely for backwards compatibility with `vote::stv()`. It has no effect on elections in which 'nseats' is explicitly specified in the call to `stv()`.
`invalid.partial`	'TRUE' if ballots which do not specify a complete ranking of candidates are informal (aka "invalid") *i.e.* ignored (with a warning). Default is 'FALSE'.
`verbose`	'TRUE' for diagnostic output
`seed`	integer seed for tie-breaking. Warning: if non-'NULL', the PRNG for R is reseeded prior to *every* random tie-break among the possibly-elected candidates. We have preserved this functionality in this branch to allow regression against the legacy codebase of `vote::stv()`. In `stv()` the default value for seed is 'NULL' rather than the legacy value of 1234, to mitigate the validity hazard of PRNG reseedings during a stochastic experiment.
`quiet`	'TRUE' to suppress console output
`digits`	number of significant digits in the output table
`backwards.compatible`	'TRUE' to regress against vote2_3.2 by disabling $margins, $fuzz, $rankingTable, $safeRank

`safety`	number of standard deviations on vote-counts, when producing a safeRank by clustering near-ties in a complete ranking
`...`	undocumented intent (preserved from legacy code)

Details

By default the preferences are not allowed to contain duplicates per ballot. However, if the argument 'equal.ranking' is set to 'TRUE', ballots are allowed to have the same ranking for multiple candidates. The desired format is such that for each preference $i$ that does not have any duplicate, there must be exactly $i – 1$ preferences $j$ with $0 < j < i$. For example, valid ordered preferences are '1; 1; 3; 4; . . . ', or '1; 2; 3; 3; 3; 6; . . . ', but NOT '1; 1; 2; 3; . . . ', or NOT '1; 2; 3; 3; 3; 5; 6; . . . '. If the data contain such invalid votes, they are automatically corrected and a warning is issued by calling the 'correct.ranking' function.

If equal ranking is not allowed ('equal.ranking = FALSE'), the argument 'invalid.partial' can be used to make ballots containing duplicates or gaps partially valid. If it is 'TRUE', a ballot is considered valid up to a preference that is in normal case not allowed. For example, ballots '1; 2; 3; 4; 4; 6' or '1; 2; 3; 5; 6; 7' would be both converted into '1; 2; 3; 0; 0; 0', because the ballots contain valid ranking only up to the third preference.

By default, ties in the STV algorithm are resolved using the forwards tie-breaking method, see Newland and Briton (Section 5.2.5). Argument 'ties' can be set to "b" in order to use the backwards tie-breaking method, see O'Neill (2004). In addition, both methods are complemented by the following "ordered" method: Prior to the STV election candidates are ordered by the number of first preferences. Equal ranks are resolved by moving to the number of second preferences, then third and so on. Remaining ties are broken by random draws. Such complete ordering is used to break any tie that cannot be resolved by the forwards or backwards method. If there is at least one tie during the processing, the output contains a row indicating in which count a tie-break happened (see the 'ties' element in the Value section for an explanation of the symbols).

The ordered tiebreaking described above can be analysed from outside of the 'stv' function by using the 'ordered.tiebreak' function for viewing the a-priori ordering (the highest number is the best and lowest is the worst). Such ranking is produced by comparing candidates along the columns of the matrix returned by 'ordered.preferences'.

Value

object of class 'vote.stv'. Note: the winning margins in this object are valid for the elected candidates and their (total) ranking, but must be adjusted within tiegroups to be valid for the candidates' (possibly partial) safeRank.

Examples

```
data(food_election)
stv(food_election, safety = 0.0)
stv(food_election, nseats = 2)
```

`summary.SafeRankExpt` *summary method for SafeRankExpt*

Description

summary method for SafeRankExpt

Usage

```
## S3 method for class 'SafeRankExpt'
summary(object, ...)
```

Arguments

`object`	experimental results to be summarised
`...`	args for generic summary()

Value

summary.SafeRankExpt object

`summary.SafeVote.approval` *summary method for approval results*

Description

summary method for approval results

Usage

```
## S3 method for class 'SafeVote.approval'
summary(object, ...)
```

Arguments

`object, ...`	undocumented

Value

undocumented

summary.SafeVote.condorcet

Summary method for condorcet() results

Description

Summary method for condorcet() results

Usage

```
## S3 method for class 'SafeVote.condorcet'
summary(object, ...)
```

Arguments

`object`	of type SafeVote.condorcet
`...`	undocumented, currently unused

Value

data.frame object

summary.SafeVote.plurality

summary method for plurality object

Description

summary method for plurality object

Usage

```
## S3 method for class 'SafeVote.plurality'
summary(object, ...)
```

Arguments

`object, ...`	undocumented

Value

descriptive dataframe

`summary.SafeVote.score` *summary method for score object*

Description

summary method for score object

Usage

```
## S3 method for class 'SafeVote.score'
summary(object, ...)
```

Arguments

`object, ...`	undocumented

Value

undocumented

`summary.SafeVote.stv` *summary() method for a SafeVote result*

Description

summary() method for a SafeVote result

Usage

```
## S3 method for class 'SafeVote.stv'
summary(object, ..., digits = 3)
```

Arguments

`object`	undocumented, legacy code
`...`	undocumented
`digits`	undocumented

Value

data.frame summarising 'object', for use by 'print' method `sumOfVotes` *internal method, computes column-sums*

Description

Renamed from 'sum.votes' to avoid confusion with the generic sum()

Usage

```
sumOfVotes(votes)
```

Arguments

`votes`	ballots are rows, candidates are columns

Value

vector of votes for each candidate

`testAdditions` *Test the sensitivity of a result to tactical voting.*

Description

Ballots are added until a specified number of simulated elections ('arep') have been held. If a 'favoured' candidate is specified, then the ballot-box is stuffed with ballots awarding first-preference to this candidate. Alternatively, a 'tacticalBallot' may be specified. If both 'favoured' and 'tacticalBallot' are 'NULL', then a random candidate is selected as the favoured one.

Usage

```
testAdditions(
  votes,
  ainc = 1,
  arep = NULL,
  favoured = NULL,
  tacticalBallot = NULL,
  rankMethod = "safeRank",
  countMethod = "stv",
  countArgs = list(),
  exptName = NULL,
  equiet = FALSE,
  everbose = FALSE
)
```

Arguments

`votes`	A set of ballots, as in vote_2.3.2
`ainc`	Number of ballots to be added in each step
`arep`	Maximum number of ballot-stuffed elections to run
`favoured`	Name of the candidate being "plumped". If 'NULL', a random candidate is selected from among the candidates not initially top-ranked. All other candidates are fully-ranked at random, with an identical ballot paper being stuffed multiple times. An integer value for 'favoured' is interpreted as an index into the candidate names.
`tacticalBallot`	A ballot paper i.e. a vector of length 'ncol(ballots)'. If this argument is non-'NULL', it takes precedence over 'favoured' when the ballot box is being stuffed.
`rankMethod`	"safeRank" (default), "elected", or "rank". "rank" is a total ranking of the candidates, with ties broken at random. "elected" assigns rank=1 to elected candidates, rank=2 for eliminated candidates.
`countMethod`	countMethod "stv" (default) or "condorcet"
`countArgs`	List of args to be passed to countMethod (in addition to votes)
`exptName`	stem-name of experimental units *e.g.* "E". If 'NULL', then a 3-character string of capital letters is chosen at random.
`equiet`	'TRUE' to suppress all experimental output
`everbose`	'TRUE' to produce diagnostic output from the experiment

Value

A matrix of experimental results, of dimension n by $2m + 1$, where n is the number of elections and m is the number of candidates. The first column is named "nBallots". Other columns indicate the ranking of the eponymous candidate, and their margin over the next-lower-ranked candidate.

Examples

```
data(food_election)
testAdditions(food_election, arep = 2, favoured = "Strawberries",
  countArgs = list(safety = 0))
```

testDeletions *Assess the safety of a preliminary result for an election*

Description

Ballots are deleted at random from the ballot-box, with election results computed once per 'dinc' ballot-deletions. The experiment terminates after a specified number of ballots have been deleted, or a specified number of ballot-counts have occurred. Note: these ballot-counts are correlated. Use `testFraction()` to experiment with independently-drawn samples from the ballot-box.

Usage

```
testDeletions(
  votes,
  countMethod = "stv",
  countArgs = list(),
  dstart = NULL,
  dinc = NULL,
  dlimit = NULL,
  drep = NULL,
  rankMethod = "safeRank",
  exptName = NULL,
  equiet = FALSE,
  everbose = FALSE
)
```

Arguments

`votes`	A set of ballots, as in vote_2.3.2
`countMethod`	"stv" (default) or "condorcet"
`countArgs`	List of args to be passed to 'countMethod' (in addition to 'votes')
`dstart`	Number of ballots in the first ballot-count (selected at random from 'votes', without replacement)
`dinc`	Number of ballots to be deleted in subsequent steps
`dlimit`	Maximum number of ballots to delete (in addition to 'dstart')
`drep`	Maximum number of elections (required if 'dinc=0')
`rankMethod`	"safeRank" (default), "elected", or "rank". "rank" is a total ranking of the candidates, with ties broken at random. "elected" assigns rank=1 to elected candidates, rank=2 for eliminated candidates.
`exptName`	stem-name of experimental units *e.g.* "E". If 'NULL', then a 3-character string of capital letters is chosen at random.
`equiet`	TRUE to suppress all experimental output
`everbose`	TRUE to produce diagnostic output from the experiment

Value

'SafeRankExpt' object, describing this experiment and its results

Examples

```
data(food_election)
testDeletions(food_election)
testDeletions(food_election, countMethod="stv",
  countArgs=list(complete.ranking=TRUE))
```

testFraction — *Bootstrapping experiment, with fractional counts of a ballot box.*

Description

Starting from some number ('astart') of randomly-selected ballots, an increasingly-large collection of randomly-selected ballots are counted. The ballots are chosen independently without replacement for each experimental unit; if you want to count decreasingly-sized portions of a single sample of ballots, use `testDeletions()`.

Usage

```
testFraction(
  votes = NULL,
  astart = NULL,
  ainc = NULL,
  arep = NULL,
  trep = NULL,
  rankMethod = "safeRank",
  countMethod = "stv",
  countArgs = list(),
  exptName = NULL,
  equiet = FALSE,
  everbose = FALSE
)
```

Arguments

`votes`	A numeric matrix: one row per ballot, one column per candidate
`astart`	Starting number of ballots (min 2)
`ainc`	Number of ballots to be added in each step. Must be non-negative.
`arep`	Number of repetitions of the test on each step. Required to be non-'NULL' if 'ainc=0' && is.null(trep)'.
`trep`	Limit on the total number of simulated elections. Required to be non-'NULL' if 'ainc=0 && is.null(arep)'.
`rankMethod`	"safeRank" (default), "elected", or "rank". "rank" is a total ranking of the candidates, with ties broken at random. "elected" assigns rank=1 to elected candidates, rank=2 for eliminated candidates.
`countMethod`	countMethod "stv" (default) or "condorcet"
`countArgs`	List of args to be passed to 'countMethod' (in addition to 'votes')
`exptName`	stem-name of experimental units *e.g.* "E". If 'NULL', then a 3-character string of capital letters is chosen at random.
`equiet`	'TRUE' to suppress all experimental output
`everbose`	'TRUE' to produce diagnostic output from the experiment

Value

SafeRankExpt object of experimental results.

Examples

```
data(food_election)
testFraction(food_election, countMethod="condorcet",
             countArgs=list(safety=0.5,complete.ranking=TRUE))
testFraction(dublin_west, astart=20, ainc=10, arep=2, trep=3,
             countMethod="stv", rankMethod="elected", equiet=FALSE)
```

`translate.ties` *Undocumented internal method from original code*

Description

Undocumented internal method from original code

Usage

```
translate.ties(ties, method)
```

Arguments

`ties`	undocumented
`method`	'f' for forward, 'b' for backward

Value

undocumented

`uk_labour_2010` *UK Labour Party Leader 2010*

Description

These are the ballots cast by Labour MPs and MEPs in an election of their party's leader in 2010, as published by the Manchester Guardian. The names of the electors have been suppressed in this file, but are available at rangevoting.org, along with extensive commentary on the election.

Usage

```
data(uk_labour_2010)
```

Format

A data frame with 266 observations and 5 candidates.

view *generic view() for classes defined in this package*

Description

generic view() for classes defined in this package

Usage

```
view(object, ...)
```

Arguments

`object`	election object to be viewed
`...`	additional parameters, passed to formattable::formattable()

Value

html-formatted object, with side-effect in RStudio's Viewer pane

view.SafeVote.approval *view method for approval object*

Description

view method for approval object

Usage

```
## S3 method for class 'SafeVote.approval'
view(object, ...)
```

Arguments

`object, ...`	undocumented

Value

undocumented

view.SafeVote.condorcet

view method for SafeVote.condorcet

Description

view method for SafeVote.condorcet

Usage

```
## S3 method for class 'SafeVote.condorcet'
view(object, ...)
```

Arguments

`object`	of type SafeVote.condorcet
`...`	see view.SafeVote.approval

Value

view object

view.SafeVote.plurality

view method for plurality object

Description

view method for plurality object

Usage

```
## S3 method for class 'SafeVote.plurality'
view(object, ...)
```

Arguments

`object, ...`	undocumented

Value

undocumented

`view.SafeVote.score` *view method for score object*

Description

view method for score object

Usage

```
## S3 method for class 'SafeVote.score'
view(object, ...)
```

Arguments

`object, ...`	undocumented

Value

undocumented

`view.SafeVote.stv` *view method for the result of an stv() ballot-count*

Description

view method for the result of an stv() ballot-count

Usage

```
## S3 method for class 'SafeVote.stv'
view(object, ...)
```

Arguments

`object`	object to be viewed
`...`	additional parameters, passed to formattable::formattable()

Value

html-formatted object

winnerMargin *Find a winner and their margin of victory*

Description

Find a winner and their margin of victory

Usage

```
winnerMargin(votes)
```

Arguments

`votes`	cleaned ballots

Value

length-2 vector: the index of a winning candidate, and their margin of victory (0 if a tie, NA if no losers)

yale_ballots *Yale Faculty Senate 2016*

Description

This data follows the structure of a 2016 Yale Faculty Senate election, with candidate names anonymised and permuted. Imported to SafeVote from STV v1.0.2, after applying the 'STV::cleanBallots' method to remove the ten empty rows.

Usage

```
data(yale_ballots)
```

Format

A data frame with 479 observations and 44 candidates.

Index