Package 'vote'

May 8, 2026

Type Package

Title Election Vote Counting

Version 2.5-2

Date 2026-02-27

Description Counting election votes and determining election results by different methods, including the single transferable vote or ranked choice, approval, score, plurality, condorcet and two-round runoff methods (Raftery et al., 2021 <doi:10.32614/RJ-2021-086>).

Depends R (>= 3.5.0)

Imports formattable, knitr, data.table, fields

Suggests ggplot2, testthat

License GPL (>= 2)

NeedsCompilation no

Author Hana Sevcikova [aut, cre],
Bernard Silverman [aut],
Adrian Raftery [aut]

Maintainer Hana Sevcikova <hanas@uw.edu>

Repository CRAN

Date/Publication 2026-02-27 17:20:02 UTC

Contents

vote-package *Election Vote Counting*

Description

Counting election votes and determining election results by different methods, including the single transferable vote (ranked choice), approval, score, plurality, condorcet and two-round runoff methods. Details about the methods and package functions can be found in Raftery et al. (2021). Versions of all methods where votes are weighted be given weights are also implemented.

Details

The main function of the package is called `count.votes`. If no specific method is passed, it decides on the basis of the available data which method is the most appropriate. Specific methods can also be invoked explicitly. The following voting methods are available:

- stv: Single transferable vote (STV) where voters rank candidates in order. It is also known as ranked choice voting or instant runoff.
- score: Range voting where each voter gives each candidate a score within a specific range.
- approval: Voters give each candidate one (approve) or zero (not approve).
- plurality: Each voter chooses one candidate.
- condorcet: Voters rank candidates in order. The winner is determined in pairwise comparisons.
- tworound.runoff: Two-round majority system with ranked ballots. If no candidate gets the majority, there is a run-off between the top two candidates.

Output of these functions can be viewed using `summary` methods, or in a browser using `view` methods. The `summary` methods return a data frame which can be stored in a file, see Example below. Outputs of the `stv` method can be plotted in a graph. The joint and marginal distributions of ranked votes (for `stv`, `condorcet` and `tworound.runoff`) can be visualized in an image plot.

Functions `invalid.votes`, `valid.votes` and `corrected.votes` can be used to check the validity of ballots for the various methods, including corrections made within the methods. Function `correct.ranking` can be used to make ballot corrections to ranked data, including ballots with equal preferences.

Example datasets are included. The `ims_election` dataset contains anonymized ballots from a past Council election of the Institute of Mathematical Statistics (IMS) which uses the STV method. Modifications of this dataset are available (`ims_approval`, `ims_score`, `ims_plurality`) as examples of data required by the various methods. The `food_election` dataset taken from Wikipedia can be used to test the STV method. Similarly, methods for ranked voting can be applied to the `dublin_west` dataset which contains election ballots from the 2002 election to the Dublin West constituency in Ireland.

Author(s)

Hana Sevcikova, Bernard Silverman, Adrian Raftery

Maintainer: Hana Sevcikova

References

Raftery, A.E., Sevcikova, H. and Silverman, B.W. (2021). The vote Package: Single Transferable Vote and Other Electoral Systems in R. The R Journal, 13(2), 673-696. doi:10.32614/RJ2021086.

Examples

```
data(ims_election)
res <- count.votes(ims_election, method = "stv", nseats = 5)
summary(res)

# View invalid votes
invalid.votes(res)

## Not run:
# View results in a browser
view(res)

# Write election results into a csv file
s <- summary(res)
write.csv(s, "IMSstvresults.csv")
## End(Not run)
```

`approval` *Approval and Plurality Vote Count*

Description

Count votes using the approval and plurality methods. Each voter can select candidates using 1 for a selection and 0 otherwise. In the approval method, any number of candidates can be selected by a voter, while in the plurality method only one candidate can be chosen by a voter. Thus, plurality voting is a special case of approval voting. The winner(s) in either method is/are the most-approved candidate(s). Each vote can be weighted.

Usage

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

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

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

plurality(votes, nseats = 1, fsep = "\t", weight.column = NULL,
    quiet = FALSE, ...)

## S3 method for class 'vote.plurality'
```

```
summary(object, ...)

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

Arguments

votes	Matrix or data frame of zeros and ones containing the votes. Rows correspond to the votes, columns correspond to the candidates. If it is a character string it is interpreted as a file name from which the votes are to be read. Missing values (NA) are interpreted as zeros. The dataset can have an extra column containing a weight for each vote. Name of this column must be passed into the argument weight.column.
nseats	Number of candidates to be elected.
fsep	If votes is a file name, this argument gives the column separator in the file.
weight.column	Name of a column in the votes dataset that represent weights. If given, a weighted voting is performed.
quiet	If TRUE no output is printed.
...	Not used.
object	Object of class vote.approval or vote.plurality.

Value

Functions approval and plurality return an object of class vote.approval and vote.plurality, respectively, both of which are lists with the following objects:

elected	Vector of names of the elected candidates in the order in which they were elected.
totals	Vector of total votes in the same order as candidates (columns) in the input data.
data	Input data with invalid votes removed. Weights are attached as an attribute called "weights".
invalid.votes	Matrix of invalid votes that were removed from the original dataset.

Author(s)

Hana Sevcikova, Adrian Raftery

References

https://en.wikipedia.org/wiki/Approval_voting

https://en.wikipedia.org/wiki/Plurality_voting_method

See Also

count.votes

Examples

```
# Example using the IMS Council dataset modified for approval voting
data(ims_approval)
approval(ims_approval) # Li wins

# Increase the weight of voters who did not vote for Li
weighted.approval <- cbind(ims_approval, weight = 1)
weighted.approval$weight[ims_approval$Li == 0] <- 2
approval(weighted.approval, weight.column = "weight") # now Jasper wins

# Example using the IMS Council dataset modified for plurality voting
data(ims_plurality)
pl.ims <- plurality(ims_plurality)
invalid.votes(pl.ims)

# Can we get Wang to win by increasing the weight of its votes?
weighted.plurality <- cbind(ims_plurality, weight = 1)
weighted.plurality$weight[ims_plurality$Wang == 1] <- 10
plurality(weighted.plurality, weight.column = "weight") # now Wang wins
```

---

condorcet *Condorcet Vote Count*

Description

Count votes using the Condorcet voting method.

Usage

```
condorcet(votes, runoff = FALSE, fsep = '\t', weight.column = NULL,
    quiet = FALSE, ...)

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

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

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

Arguments

votes	Matrix or data frame containing the votes. Rows correspond to the votes, columns correspond to the candidates. If it is a character string it is interpreted as a file name from which the votes are to be read. The dataset can have an extra column containing a weight for each vote. Name of this column must be passed into the argument `weight.column`.

runoff	Logical. If TRUE and no condorcet winner exists, the election goes into a run-off, see below for details.
fsep	If votes is a file name, this argument gives the column separator in the file.
weight.column	Name of a column in the votes dataset that represent weights. If given, a weighted Condorcet is performed.
quiet	If TRUE no output is printed.
object, x	Object of class vote.condorcet.
...	Additional arguments passed to the underlying functions. For the image function, see arguments for image.vote.stv, especially xpref, ypref, all.pref and proportion.

Details

The Condorcet method elects the candidate that wins a majority of the ranked vote in every head-to-head election against each of the other candidates. I.e., the Condorcet winner is a candidate that beats all other candidates in pairwise comparisons. Analogously, a Condorcet loser is a candidate that loses against all other candidates. If weights are used, each win is multiplied by the corresponding weight. Neither Condorcet winner nor loser might exist.

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 $1, 2, \ldots, r, 0, 0, 0, 0$, 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. An additional column of weights can be present. In such a case its name must be passed to the weight.column argument.

Note that if equal preferences are used, they are automatically converted into a format where for each preference i that does not have any duplicate, there must be exactly $i - 1$ preferences j with $0 < j < i$. It is the same ranking as one would obtain with rank(x, ties.method = "min"). If a conversion of a vote occurs, a warning is issued. That is done internally by calling the correct.ranking function.

The image function visualizes the joint distribution of two preferences (if all.pref=FALSE) given by 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).

Value

Function condorcet returns an object of class vote.condorcet which is a list with the following objects:

elected	The Condorcet winner if exists, otherwise NULL.
loser	The Condorcet loser if exists, otherwise NULL.

`totals` nc x nc matrix where nc is the number of candidates. Element ij = 1 if i won against j, otherwise 0.

`runoff.winner` The run-off winner if exists and if the `runoff` argument was set to `TRUE`, otherwise `NULL`.

`runoff.participants` List of run-off participants if the `runoff` argument was set to `TRUE`, otherwise `NULL`.

`data` Input data (possibly corrected) with invalid votes removed. Weights are attached as an attribute called "weights".

`invalid.votes` Matrix of invalid votes that were removed from the original dataset.

Author(s)

Hana Sevcikova, Salvatore Barbaro

References

Condorcet, Marquis de (1785). Essai sur l'application de l'analyse a la probabilite des decisions rendues a la probabilite des voix. Paris: De l'imprimerie royale.

https://en.wikipedia.org/wiki/Condorcet_method

Sen A. (2017). Collective Choice and Social Welfare. Harvard University Press, Cambridge, Massachusetts (Chapter A4*).

Examples

```
data(food_election)
cdc.food <- condorcet(food_election)
summary(cdc.food)
# show the marginal distribution of the preferences
par(mai=c(1, 1.2, 0.8, 0.4)) # expand the left margin
image(cdc.food, all.pref = TRUE)

# Example with a runoff
votes <- matrix(c(2, 1, 3, 4,
                  2, 1, 3, 4,
                  4, 3, 2, 1,
                  4, 3, 2, 1,
                  1, 4, 3, 2), byrow = TRUE, nrow = 5)
colnames(votes) <- LETTERS[1:4]
cdc.v <- condorcet(votes, runoff = TRUE)

# Weighted condorcet
wvotes <- cbind(votes, weight = c(1, 1, 1.5, 1.5, 1))
wcdc.v <- condorcet(wvotes, runoff = TRUE, weight.column = "weight")
```

count.votes *Count Votes*

Description

Count votes using one of five methods. View valid, invalid and corrected ballots.

Usage

```
count.votes(votes, method = c("auto", "plurality", "approval", "stv",
    "score", "condorcet", "tworound.runoff"), fsep = "\t", ...)

invalid.votes(object)
valid.votes(object)
corrected.votes(object)
```

Arguments

votes	Matrix or data frame containing the votes. Rows correspond to the votes, columns correspond to the candidates. If it is a character string it is interpreted as a file name from which the votes are to be read. The dataset can have an extra column containing a weight for each vote. Name of this column must be passed into the argument weight.column which is supported by all methods.
method	Voting method to use. If "auto", the input data is passed through a checker for each of the methods and the one with the largest number of valid votes is used. In case of the same number of valid votes, it goes by their ordering in the function definition.
fsep	If votes is a file name, this argument gives the column separator in the file.
...	Additional arguments passed to the underlying functions, e.g. nseats, max.score etc.
object	Object returned by one of the functions plurality, approval, stv, score, condorcet, tworound.runoff.

Value

Depending which method is used, count.votes returns an object of class vote.plurality, vote.approval, vote.stv, vote.score, vote.condorcet, or vote.tworound.runoff.

The data component in this object contains the (possibly corrected) input dataset of votes which is an object of class vote.matrix. This is a matrix with an additional attribute weights containing a weight for each vote.

Functions valid.votes and invalid.votes return a subset of the input data with valid records and invalid records, respectively.

Function corrected.votes can be used when votes are automatically corrected (as in stv and condorcet). It returns a list with the uncorrected votes (item original), the corrected votes (item new), and its indices within the original votes dataset (item index).

Author(s)

Hana Sevcikova, Bernard Silverman

See Also

stv, approval, score, condorcet

Examples

```
# Example using the IMS Council dataset modified for score voting
data(ims_score)
# should recognize that it is a dataset with score voting data
count.votes(ims_score, max.score = 9, larger.wins = FALSE)

# All records with score larger than 8 are excluded
res <- count.votes(ims_score, method = "score", max.score = 8)
head(invalid.votes(res))

summary(res)

# For a corrected.votes() example see ?stv
```

---

dublin_west *Election Dataset to Dublin West Constituency*

Description

Dataset containing ranked votes for the Dublin West constituency in 2002, Ireland. Results of that STV elections can be viewed at https://en.wikipedia.org/wiki/Dublin_West#2002_general_election. They can be reproduced via the stv function, see Example below.

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.

References

https://en.wikipedia.org/wiki/Dublin_West#2002_general_election

Examples

```
data(dublin_west)
head(dublin_west)

## Not run:
# produce similar results as in the Wikipedia link above
dwstv <- stv(dublin_west, nseats = 3, eps = 1, constant.quota = TRUE)

# plot results
plot(dwstv)
image(dwstv)
image(dwstv, all.pref = TRUE)
## End(Not run)
```

---

food_election *Example Dataset*

Description

Dataset on food election which serves as a simple example for the STV method taken from Wikipedia.

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.

Source

https://en.wikipedia.org/wiki/Single_transferable_vote#Example

Examples

```
data(food_election)
head(food_election)
```

ims_election *Datasets on IMS Election*

Description

Datasets containing anonymized votes for a past Council election of the Institute of Mathematical Statistics (IMS). The dataset `ims_election` (named also `ims_stv`) is the original dataset used with single transferable vote, where candidate names have been changed. Each of the other datasets is a modified version of the original data to be used as an example for each of the other voting methods.

Usage

```
data("ims_election")
data("ims_stv")

data("ims_approval")
data("ims_score")
data("ims_plurality")
```

Format

A data frame with 620 observations and 10 candidates (names were made up). Each record corresponds to one ballot. Values depend on the voting method. 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.

Source

The original dataset (which was randomized and anonymized, with write-in votes removed) was obtained from the the Institute of Mathematical Statistics.

References

https://imstat.org/elections/single-transferable-voting-system/

Examples

```
data(ims_election)
head(ims_election)
```

score *Score Vote Count*

Description

Count votes using the score (or range) method. Voters give each candidate a score, the scores are added and the candidate(s) with the highest (or lowest) totals is/are elected. Votes can also be weighted.

Usage

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

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

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

Arguments

votes	Matrix or data frame containing the votes which should be numbers between 0 and max.score. Rows correspond to the votes, columns correspond to the candidates. If it is a character string it is interpreted as a file name from which the votes are to be read. Missing values (NA) are interpreted as zeros. The dataset can have an extra column containing a weight for each vote. Name of this column must be passed into the argument weight.column.
nseats	Number of candidates to be elected.
max.score	Maximum score allowed. It is used to remove invalid votes. If not given, the maximum value contained in the data is taken and thus, all non-negative votes are valid.
larger.wins	Logical argument indicating whether the winners are the candidates with the highest scores (default) or the lowest scores.
fsep	If votes is a file name, this argument gives the column separator in the file.
weight.column	Name of a column in the votes dataset that represent weights. If given, a weighted voting is performed.
quiet	If TRUE no output is printed.
...	Not used.
object	Object of class vote.score.

Value

Function score returns an object of class vote.score which is a list with the following objects:

elected	Vector of names of the elected candidates in the order in which they were elected.
totals	Vector of total votes in the same order as candidates (columns) in the input data.
larger.wins	Input argument of the same name.
data	Input data with invalid votes removed. Weights are attached as an attribute called "weights".
invalid.votes	Number of invalid votes that were removed from the original dataset.

Author(s)

Hana Sevcikova, Adrian Raftery

References

https://en.wikipedia.org/wiki/Range_voting

See Also

count.votes

Examples

```
# Example using the IMS Council dataset modified for score voting
data(ims_score)
score.ims <- score(ims_score, max.score = 9)
summary(score.ims)

# weighted version (assign random weight)
wvotes <- cbind(ims_score, weight = round(runif(nrow(ims_score), 0.5, 10), 1))
wscore.ims <- score(wvotes, max.score = 9, weight.column = "weight")
```

---

stv *Single Transferable Vote*

Description

Count votes using the single transferable voting method, also known as ranked choice voting or instant runoff. Raftery et al. (2021) describes the functionality in great detail.

Usage

```
stv(votes, nseats = NULL, eps = 0.001, equal.ranking = FALSE,
  fsep = '\t', ties = c("f", "b"), constant.quota = FALSE,
  quota.hare = FALSE, group.nseats = NULL, group.members = NULL,
  weight.column = NULL, complete.ranking = FALSE,
  invalid.partial = FALSE, impute.missing = FALSE, verbose = FALSE,
  seed = 1234, quiet = FALSE, digits = 3, ...)

## S3 method for class 'vote.stv'
summary(object, ..., complete.ranking = FALSE, digits = 3)

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

## S3 method for class 'vote.stv'
plot(x, xlab = "Count", ylab = "Preferences", point.size = 2, xmin = 0, ...)

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

## S3 method for class 'vote.stv'
complete.ranking(object, ...)

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

impute.ranking(votes, equal.ranking = FALSE, quiet = TRUE)

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

ordered.tiebreak(vmat, seed = NULL)

ordered.preferences(vmat)
```

Arguments

votes	Matrix or data frame containing the votes. Rows correspond to the votes, columns correspond to the candidates. The stv function allows to be a character string which is interpreted as a file name from which the votes are to be read. The stv function also allows an extra column containing a weight for each vote. Name of this column must be passed into the argument weight.column.
nseats	Number of candidates to be elected. By default it is half the number of candidates standing.
eps	Value added to the quota. I.e. the STV default Droop quota is computed as number_of_first_preferences/(number_of_seats + 1) + eps.
equal.ranking	If TRUE equal preferences are allowed, see below.
fsep	If votes is a file name, this argument gives the column separator in the file.

ties	Method used to break ties. By default the forwards tie-breaking is used ("f"). Value "b" invokes the backwards tie-breaking method, see O'Neill (2004).
constant.quota	Logical determining if the quota should be kept constant for all counts.
quota.hare	Changes quota calculation method from (default) Droop (FALSE) to Hare (TRUE). STV Hare quota method is computed as number_of_first_preferences/number_of_seats + eps. The actual Hare formula would entail eps = 0.
group.nseats	Minimum number of candidates to be elected who are members of a given group. I.e., number of reserved seats for a subset of candidates defined by the group.members argument.
group.members	Vector of candidate names or indices who are eligible for reserved seats given by group.nseats. If it is a vector of indices, the order of candidates is assumed to correspond to the columns of votes.
weight.column	Name of a column in the votes dataset that represent weights. If given, a weighted STV is performed.
impute.missing	Logical. If TRUE and if the data contains values of -1, those ranks are imputed while all other ranks that are equal or larger than the imputed value are shifted.
verbose	Logical. If TRUE the progress of the count will be printed.
seed	Integer. Seed of the random number generator. Only used if there are ties that cannot be resolved by the tie-breaking method. If set to NULL, the RNG is not initialized.
quiet	If TRUE no output is printed.
object, x	Object of class vote.stv.
complete.ranking	Logical. If TRUE a complete ranking is generated conditioned on the number of seats nseats.
invalid.partial	Logical. If TRUE, partially invalid votes are corrected by removing ranking starting with the first incorrect rank, see Details below.
digits	How many significant digits to be used in the output table.
xlab, ylab	Labels of the x- and y-axis.
point.size	Size of the points in the plot.
xmin	Minimum count to appear on the plot. By default, counts start at 0.
xpref, ypref	Preference for the x- and y-axis, respectively, for showing the joined distribution of the votes. It is not used if all.pref is TRUE.
all.pref	Logical. If TRUE the marginal distribution of all preferences is shown in the image. Otherwise, the joint distribution of xpref and ypref is shown.
proportion	If TRUE the preferences are shown as proportions across the x-axis, otherwise raw vote counts are shown. Only available when all.pref is FALSE.
...	Additional arguments passed to the underlying functions.
partial	Logical. The same meaning as invalid.partial.
can	Vector of candiate name(s) or indices to be removed from the set of votes.
vmat	Matrix of valid votes.

Details

For a description of the single transferable vote system see https://imstat.org/elections/single-transferable-voting-system/.

The input data `votes` is structured as follows: Row i contains the preferences of voter i numbered $1, 2, \ldots, r, 0, 0, 0, 0$, in some order. 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, representing lower preferences that were not expressed.

By default the preferences are not allowed to contain duplicates per vote. However, if the argument `equal.ranking` is set to `TRUE`, votes 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 $1, 1, 3, 4, \ldots$, or $1, 2, 3, 3, 3, 6, \ldots$, but NOT $1, 1, 2, 3, \ldots$, or NOT $1, 2, 3, 3, 3, 5, 6, \ldots$. 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 alowed (`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.

The `correct.ranking` function does the above corrections for all records, regardless if they contain duplicates or not. Its argument `partial` determines if ballots are partially set to 0 (`TRUE`), or if it is complete re-ranking, as allowed when `equal.ranking = TRUE`. It can either be used by calling it explicitly, otherwise it is called by `stv` if `equal.ranking = TRUE` or `invalid.partial = TRUE`. The function is also called from within the `condorcet` function. The `remove.candidate` function removes the given candidate(s) and adjusts the ranked votes accordingly by calling the `correct.ranking` function.

The function allows the user to impute missing values. It can be used for example, if a voter has a conflict of interest with one or more candidates and not voting for them would unfairly decrease the chances of those candidates being elected. (Note that missing values are not to be confused with lower preferences that are not expressed.) Preferences to be imputed should be set to -1 and the argument `impute.missing` to `TRUE`. Each such preference is imputed using the median rank value over the remaining votes. When computing the median rank across the votes, any value of zero is replaced by the median of the ranks not used in the corresponding vote. For example, for a ballot $1, 2, 3, 0, 0, 0$, the three zeros are replaced by the median of $4, 5, 6$, i.e. by 5, which is then used to compute the missing median rank. If the final imputed rank is larger than the number of non-zero preferences (e.g. if in a ballot $1, 2, 0, -1, 0$ the imputed value for the fourth candidate would be larger than 3), the preference is set to zero and a warning is issued. The described functionality is implemented in the `impute.ranking` function, which is called automatically from `stv` if `impute.missing = TRUE`. It can be used explicitly as well.

Note that neither `correct.ranking`, `remove.candidate` nor `impute.ranking` allow a column of weights to be present in the dataset of votes.

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 1st

preferences. Equal ranks are resolved by moving to the number of 2nd preferences, then 3rd 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`.

The `plot` function shows the evolution of the total score for each candidate as well as the quota. The `image` function visualizes the joint distribution of two preferences (if `all.pref=FALSE`) as well as the marginal distribution of all preferences (if `all.pref=TRUE`). The joint distribution can be shown either as proportions (if `proportion=TRUE`) or raw vote counts (if `proportion=FALSE`).

Method `complete.ranking` produces a complete ranking of the candidates, conditioned on the number of seats selected in the `nseats` argument. It is called from the `summary` function if the `complete.ranking` argument is set to `TRUE`.

Value

Function `stv` returns an object of class `vote.stv` which is a list with the following objects:

`elected` Vector of names of the elected candidates in the order in which they were elected.

`preferences` Matrix of preferences. Columns correspond to the candidates and rows to the counts (i.e. voting rounds).

`quotas` Vector of quotas, one for each count.

`elect.elim` Matrix of the same shape as `preferences`. Value 1 means that the corresponding candidate was elected in that round; value -1 means an elimination.

`equal.pref.allowed` Input argument `equal.ranking`.

`ties` Character vector indicating if and what tie-break happened in each count. Possible values: "" (no tie-break), "f" (forward tie-breaking method only), "fo" (forward method and ordered method), "fos" (forward method and ordered method and sampling). If the backwards tie-breaking method is used, these values are "b", "bo" and "bos".

`data` Input data (possibly corrected) with invalid votes removed. Weights are attached as an attribute called "weights".

`invalid.votes` Matrix of invalid votes that were removed from the original dataset.

`corrected.votes` List containing data about corrected votes if any. It has three or four elements, `original` (matrix of the raw votes that were corrected), (optionally) `imputed` (imputed values if any), `new` (the corrected values), `index` (index of those votes within the input `votes` dataset).

`reserved.seats` Number of reserved seats (`group.nseats`), or `NULL` if none.

`group.members` Vector of candidates eligible for reserved seats, or `NULL` if none.

The summary function returns a data frame where columns are counts and transfers, and rows are the quota, the candidates, ties and the elected and eliminated candidates. Various attributes of the data frame contain more information about the results.

The correct.ranking (impute.ranking) function returns a matrix of votes with corrected (imputed) preferences.

remove.candidate returns a matrix of votes with the given candidates removed and preferences corrected.

complete.ranking returns a data frame with a full ordering of the candidates.

ordered.preferences returns a matrix with number of preferences for each candidate and preference. These are the same values as seen by image(..., all.pref = TRUE).

ordered.tiebreak returns the ranking for each candidate based on ordered.preferences(), with the highest number being the best and the lowest number being the worst. Its attribute "sampled" indicates if there was random sampling involved in ranking each candidate.

Author(s)

Bernard Silverman, Hana Sevcikova, Adrian Raftery

References

Raftery, A.E., Sevcikova, H. and Silverman, B.W. (2021). The vote Package: Single Transferable Vote and Other Electoral Systems in R. The R Journal, 13(2), 673-696. doi:10.32614/RJ2021086.

R.A. Newland and F.S. Britton (1997). How to conduct an election by the Single Transferable Vote. ERS 3rd Edition. https://electoral-reform.org.uk/latest-news-and-research/publications/how-to-conduct-an-election-by-the-single-transferable-vote-3rd-edition/

https://imstat.org/elections/single-transferable-voting-system/

https://en.wikipedia.org/wiki/Single_transferable_vote

J.C. O'Neill (2004). Tie-Breaking with the Single Transferable Vote. Voting Matters, 18, 14-17. https://www.votingmatters.org.uk/ISSUE18/I18P6.PDF

Examples

```
# Reproducing example from Wikipedia (version from Sep 2019)
# https://en.wikipedia.org/wiki/Single_transferable_vote#Example
# Uses eps=1
data(food_election)
stv.food <- stv(food_election, nseats = 3, eps = 1)
summary(stv.food)
## Not run:
view(stv.food)
## End(Not run)

# Example of the IMS Council voting
data(ims_election)
stv.ims <- stv(ims_election, nseats = 5)
## Not run:
view(stv.ims)
plot(stv.ims)
```

```
image(stv.ims)

# write election results into a csv file
s <- summary(stv.ims)
write.csv(s, "myfile.csv")
## End(Not run)

# produce complete ranking
summary(stv.ims, complete.ranking = TRUE)

## Not run:
# Example of Dublin West 2002 elections
# https://en.wikipedia.org/wiki/Dublin_West#2002_general_election
data(dublin_west)
stv(dublin_west, nseats = 3, eps = 1)
## End(Not run)

# Example of a small committee dataset
# with four candidates (C) and four
# voting committee members (uses tie-breaking)
votes <- data.frame(C1=c(3,2,1,3), C2=c(2,1,2,4),
                    C3=c(4,3,3,1), C4=c(1,4,4,2))
stv(votes, nseats = 2, verbose = TRUE)

# Example of weighted votes
wvotes <- cbind(votes, weight = c(1, 2, 1, 2.5))
stv(wvotes, nseats = 2, weight.column = "weight", verbose = TRUE)

# Example with equal ranking and correction
votes <- data.frame(C1=c(3,2,1,3), C2=c(1,1,2,0),
                    C3=c(4,3,3,1), C4=c(1,4,2,2))
stv(votes, nseats = 2, equal.ranking = TRUE)
# vote #3 was corrected by stv which used this data:
correct.ranking(votes, quiet = TRUE)

# Example of imputing preferences
# (third voter has a conflict of interest with candidate C2)
votes <- data.frame(C1=c(3,2,1,3), C2=c(2,1,-1,0),
                    C3=c(4,3,3,1), C4=c(1,4,2,2))
res <- stv(votes, nseats = 2, impute.missing = TRUE)
corrected.votes(res)
# imputed rank 2, as it is the median(c(2, 1, 4))
# where the last 4 was derived as the median of missing ranks
# in vote four. The imputation can be also performed via
impute.ranking(votes)

# Example of using reserved seats:
# e.g. reserve two seats for students
stv(ims_election, nseats = 5, group.nseats = 2,
    group.members = c("Declan", "Claire", "Oscar")) # students

# Example of removing candidates from original votes
stv(remove.candidate(ims_election, c("Jasper", "Tilmann")), nseats = 5)
```

```
# Example of accepting partially invalid ballots
res <- stv(ims_election, invalid.partial = TRUE)

# There are now 24 invalid votes instead of 29,
# because 5 were corrected (ranking before the first
# gap/tie is valid, after that it is 0)
corrected.votes(res)
invalid.votes(res)
```

`tworound.runoff` *Two-Round Runoff Vote Count*

Description

Count votes using the two-round voting method with ranked ballots. If no candidate reaches the majority, the top two candidates go into a run-off. Votes can be weighted.

Usage

```
tworound.runoff(votes, fsep = '\t', weight.column = NULL,
    seed = NULL, quiet = FALSE, ...)

## S3 method for class 'vote.tworound.runoff'
summary(object, ...)

## S3 method for class 'vote.tworound.runoff'
view(object, ...)

## S3 method for class 'vote.tworound.runoff'
image(x, ...)
```

Arguments

`votes`	Matrix or data frame containing the votes. Rows correspond to the votes, columns correspond to the candidates. If it is a character string it is interpreted as a file name from which the votes are to be read. The dataset can have an extra column containing a weight for each vote. Name of this column must be passed into the argument `weight.column`.
`fsep`	If `votes` is a file name, this argument gives the column separator in the file.
`weight.column`	Name of a column in the `votes` dataset that represent weights. If given, a weighted voting is performed.
`seed`	Integer. Seed of the random number generator (RNG). Only used if there are ties either between candidates to enter the run-off, or between the two run-off contenders. If set to `NULL`, the RNG is not initialized.
`quiet`	If `TRUE` no output is printed.

object, x	Object of class vote.tworound.runoff.
...	Additional arguments passed to the underlying functions. For the image function, see arguments for image.vote.stv, especially xpref, ypref, all.pref and proportion.

Details

First, the number of first preferences is counted and possibly weighted if weights are given. If there is a candidate with more than 50%, that candidate gets elected. Otherwise, there is a runoff between the top two candidates.

The input data votes is structured the same way as for the stv method: Row i contains the preferences of voter i numbered $1, 2, \ldots, r, 0, 0, 0, 0$, in some order. Equal preferences are not 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.

The image function visualizes the joint distribution of two preferences (if all.pref=FALSE) given by 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).

Value

Function tworound.runoff returns an object of class vote.tworound.runoff which is a list with the following objects:

elected	The elected candidate.
totals	Vector of total votes in the same order as candidates (columns) in the input data.
totals2r	Vector of total votes from the run-off (second round).
coin.toss.winner	TRUE if the winner was sampled between candidates with the same score, otherwise FALSE.
coin.toss.runoff	TRUE if the run-off contenders were sampled from candidates with the same score. Otherwise it is FALSE.
data	Input data (possibly corrected) with invalid votes removed. Weights are attached as an attribute called "weights".
invalid.votes	Matrix of invalid votes that were removed from the original dataset.

Author(s)

Hana Sevcikova, Salvatore Barbaro

References

Sen A. (2017). Collective Choice and Social Welfare. Harvard University Press, Cambridge, Massachusetts, Chapter 10*3 (p. 243ff).

https://en.wikipedia.org/wiki/Two-round_system

Examples

```
data(ims_election)
trr <- tworound.runoff(ims_election)
summary(trr)
```

Index