Package 'PlayerRatings'

March 1, 2020

Version 1.1-0

Date 2020-02-28

Title Dynamic Updating Methods for Player Ratings Estimation

Author Alec Stephenson and Jeff Sonas.

Maintainer Alec Stephenson <alec_stephenson@hotmail.com>

Depends R (>= 3.5.0)

Description Implements schemes for estimating player or team skill based on dynamic updating. Implemented methods include Elo, Glicko, Glicko-2 and Stephenson. Contains pdf documentation of a reproducible analysis using approximately two million chess matches. Also contains an Elo based method for multi-player games where the result is a placing or a score. This includes zero-sum games such as poker and mahjong.

LazyData yes

License GPL-3

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-03-01 15:50:06 UTC

R topics documented:

`aflodds` *Australian Football Game Results and Odds*

Description

The `aflodds` data frame has 675 rows and 9 variables. It shows the results and betting odds for 675 Australian football games played by 18 teams from 26th March 2009 until 24th June 2012.

Usage

```
aflodds
```

Format

This data frame contains the following columns:

Date A date object showing the date of the game.

Week The number of weeks since 25th March 2009.

HomeTeam The home team name.

AwayTeam The away team name.

HomeScore The home team score.

AwayScore The home team score.

Score A numeric vector giving the value one, zero or one half for a home win, an away win or a draw respectively.

HomeOdds The best decimal odds offered for the home team. This is missing for some earlier games.

AwayOdds The best decimal odds offered for the away team. This is missing for some earlier games.

Source

Wikipedia and www.oddsportal.com.

elo *The Elo Rating System*

Description

Implements the Elo rating system for estimating the relative skill level of players in two-player games such as chess.

Usage

```
elo(x, status = NULL, init = 2200, gamma = 0, kfac = 27,
    history = FALSE, sort = TRUE, ...)
```

Arguments

x	A data frame containing four variables: (1) a numeric vector denoting the time period in which the game took place (2) a numeric or character identifier for player one (3) a numeric or character identifier for player two and (4) the result of the game expressed as a number, typically equal to one for a player one win, zero for a player two win and one half for a draw.
status	A data frame with the current status of the system. If not NULL, this needs to be a data frame in the form of the ratings component of the returned list, containing variables named Player, Rating, and optionally Games, Win, Draw, Loss and Lag, which are set to zero if not given.
init	The rating at which to initialize a new player not appearing in status. Must be a single number. If different initializations for different players are required, this can be done using status.
gamma	A player one advantage parameter; either a single value or a numeric vector equal to the number of rows in x. Positive values favour player one, while negative values favour player two. This could represent the advantage of playing at home, or the advantage of playing white for chess. Note that this is not passed to predict.rating, which has its own gamma parameter.
kfac	The K factor parameter. Can be a single number or a vectorized function of two arguments, the first being the ratings and the second being the number of games played. See kfide, kgames and krating for examples.
history	If TRUE returns the entire history for each period in the component history of the returned list.
sort	If TRUE sort the results by rating (highest to lowest). If FALSE sort the results by player.
...	Passed to the function kfac.

Details

The Elo rating system is a simple method for evaluating the skill of players. It has been used since around 1960 and is still employed in various settings. Although the basic form uses only the ratings, additional complexity is commonly introduced by adding a player one advantage parameter and by using different K factors. A player one advantage parameter has been added to the original definition in the reference. A player one advantage parameter is also used for prediction purposes in `predict.rating`.

This implementation has a simple initialization, and allows the K factor to depend on both the ratings and the number of games played. Default values are roughly optimized the chess data analyzed in the file doc/ChessRatings.pdf, using the binomial deviance criterion and considering only constant K factors. See the function `fide` for a different implementation.

Value

A list object of class `"rating"` with the following components

`ratings`	A data frame of the results at the end of the final time period. The variables are self explanatory except for `Lag`, which represents the number of time periods since the player last played a game. This is equal to zero for players who played in the latest time period, and is also zero for players who have not yet played any games.
`history`	A three dimensional array, or `NULL` if `history` is `FALSE`. The row dimension is the players, the column dimension is the time periods. The third dimension gives different parameters.
`gamma`	The player one advantage parameter.
`kfac`	The K factor or K factor function.
`type`	The character string `"Elo"`.

References

Elo, Arpad (1978) The Rating of Chessplayers, Past and Present. Arco. ISBN 0-668-04721-6.

See Also

`fide`, `glicko`, `kfide`

Examples

```
afl <- aflodds[,c(2,3,4,7)]
robj <- elo(afl)
robj

robj <- elo(afl[afl$Week==1,])
for(i in 2:max(afl$Week)) robj <- elo(afl[afl$Week==i,], robj$ratings)
robj
```

elom *Multi-player Elo Based Rating System*

Description

Implements the Elo based rating system for for multi-player games where the result is a placing or a score. This includes zero-sum games such as poker and mahjong. The default arguments used here are those used by Tenhou for riichi mahjong.

Usage

```
elom(x, nn = 4, exact = TRUE, base = c(30,10,-10,-30), status = NULL,
    init = 1500, kfac = kriichi, history = FALSE, sort = TRUE, ...,
placing = FALSE)
```

Arguments

x	A data frame containing 2nn+1 variables, where nn is the number of players in a single game: (col 1) a numeric vector denoting the time period in which the game took place, (cols 2 to nn+1) numeric or character identifiers for each of the nn players, (cols nn+2 to 2nn+1) the result of the game expressed as a number, typically a score for each player (e.g. the number of remaining chips in poker). Negative numbers are allowed. Alternatively, the result can be a placing (e.g. 1 for first place, 2 for second place), in which case the placing argument MUST be set to TRUE. Placings must be integers: in the event of a tie, multiple players can be given the same placing.
nn	Number of players in a single game. If the number of players varies, then this argument should be set to the maximum number of players in any game, and the exact argument should be set to FALSE. Unused player identifiers in x must then be set to the missing value NA. The game score for NA player identifiers is ignored and therefore can also be set to NA.
exact	If TRUE (the default), then every game always has exactly nn players. If TRUE, then x cannot have missing values.
base	The base values used for the rating. Can be a numeric vector of length equal to nn, a numeric matrix with nrow(x) rows and nn columns, or a vectorized function of the game score. If a numeric vector, then the person with the highest score gets base[1], the person with the second highest score gets base[2], and so on. In the event of a tie on the game score, tied players are given the largest available base value. For games with less than nn players, see Details. If base is a matrix, then the ith row is used for the ith game in x. If base is a vectorized function, then each player gets the result of the function applied to the game score. In Riichi mahjong, where players start with 25000 points, a typical example might be function(x) (x-25000)/250.
status	A data frame with the current status of the system. If not NULL, this needs to be a data frame in the form of the ratings component of the returned list, containing variables named Player, Rating, and optionally Games, 1st, 2nd, 3rd and so on, and finally Lag, which are all set to zero if not given.

`init`	The rating at which to initialize a new player not appearing in `status`. Must be a single number. If different initializations for different players are required, this can be done using `status`.
`kfac`	The K factor parameter. Can be a single number or a vectorized function of two arguments, the first being the ratings and the second being the number of games played. See `kriichi` for an example.
`history`	If `TRUE` returns the entire history for each period in the component `history` of the returned list.
`sort`	If `TRUE` sort the results by rating (highest to lowest). If `FALSE` sort the results by player.
`...`	Passed to the function `kfac`.
`placing`	If the results are given as placings (e.g. 1 for first place, 2 for second place) then this argument MUST be set to `TRUE`, otherwise the placings will be interpreted as game scores.

Details

For multi-player games there is no player one advantage parameter (e.g. a home advantage in football or a white advantage in chess).

If the sum of the vector `base` is not zero, or if `base` is a function which is not zero when evaluated at the starting chip/points value, then you may observe unusual behaviour and/or substantial ratings inflation/deflation.

The two-player Elo system is based on game outcomes in the interval [0,1] and therefore uses a different scaling. As a result, the K factors here should be smaller. The default (as used by Tenhou) is a K factor of 0.2 for players that have played a large number of games (see `kriichi`).

If the number of players varies and `base` is a vector (of length `nn`), then if the game has less than `nn` players, the vector is reduced by successively removing the centre value (for odd lengths) or by averaging both centre values (for even lengths). For example, if the `x` data frame contains both four-player and three-player mahjong games, then under the default values the three-player base vector becomes `c(30,0,-30)`, which is consistent with the vector that Tenhou uses for three-player mahjong.

A numeric matrix can be used to allocate different `base` vectors to different games. For example, in Riichi mahjong, games can be Tonpuusen (East round only) or Hanchan (East and South rounds), and you may wish to allocate different base vectors to each type.

Value

A list object of class `"rating"` with the following components

`ratings`	A data frame of the results at the end of the final time period. The variables are self explanatory except for `Lag`, which represents the number of time periods since the player last played a game. This is equal to zero for players who played in the latest time period, and is also zero for players who have not yet played any games.
`history`	A three dimensional array, or `NULL` if `history` is `FALSE`. The row dimension is the players, the column dimension is the time periods. The third dimension gives different parameters.

nn	The number of players for a single game.
kfac	The K factor or K factor function.
type	The character string "EloM".

References

Elo, Arpad (1978) The Rating of Chessplayers, Past and Present. Arco. ISBN 0-668-04721-6.

See Also

elo, fide, glicko, kriichi

Examples

```
robj <- elom(riichi)
robj

ut <- unique(riichi$Time)
robj <- elom(riichi[riichi$Time == ut[1],])
for(i in 2:length(ut)) {
  robj <- elom(riichi[riichi$Time == ut[i],], status = robj$ratings)
}
robj
```

---

fide *The Elo Rating System Employed By The FIDE*

Description

Implements the Elo rating system for estimating the relative skill level of players in two-player games such as chess, implementing a version similar to that employed by the FIDE.

Usage

```
fide(x, status = NULL, init = 2200, gamma = 0, kfac = kfide,
    history = FALSE, sort = TRUE, ...)
```

Arguments

x	A data frame containing four variables: (1) a numeric vector denoting the time period in which the game took place (2) a numeric or character identifier for player one (3) a numeric or character identifier for player two and (4) the result of the game expressed as a number, typically equal to one for a player one win, zero for a player two win and one half for a draw.
status	A data frame with the current status of the system. If not NULL, this needs to be a data frame in the form of the ratings component of the returned list, containing variables named Player, Rating, and optionally Games, Win, Draw, Loss Lag and Elite, which are set to zero if not given, and Opponent, which is set to the player rating if not given.

init	The rating at which to initialize a new player not appearing in status. Must be a single number. If different initializations for different players are required, this can be done using status.
gamma	A player one advantage parameter; either a single value or a numeric vector equal to the number of rows in x. Positive values favour player one, while negative values favour player two. This could represent the advantage of playing at home, or the advantage of playing white for chess. Note that this is not passed to predict.rating, which has its own gamma parameter.
kfac	The K factor parameter. Can be a single number or a vectorized function of three arguments, the first being the ratings, the second being the number of games played, and the third being a binary indicator for whether or not a player has ever achieved a rating above 2400. See kfide, kgames and krating for examples. The function kfide is used by default.
history	If TRUE returns the entire history for each period in the component history of the returned list.
sort	If TRUE sort the results by rating (highest to lowest). If FALSE sort the results by player.
...	Passed to the function kfac.

Details

The Elo rating system is a simple method for evaluating the skill of players. It has been used since around 1960 and is still employed in various settings. Although the basic form uses only the ratings, additional complexity is commonly introduced by adding a player one advantage parameter and by using different K factors. A player one advantage parameter has been added to the original definition in the reference. A player one advantage parameter is also used for prediction purposes in predict.rating.

This implementation uses default arguments that are consistent with the implementation of FIDE for rating chess players. It does not employ the initialization used by FIDE. For the chess data analyzed in the file doc/ChessRatings.pdf, prediction performance is poor because the default values of the K factors are too low. This can be altered using the kv argument which is passed to the function kfide.

Value

A list object of class "rating" with the following components

ratings	A data frame of the results at the end of the final time period. The variables are self explanatory except for Lag, which represents the number of time periods since the player last played a game, Elite, which is a binary indicator for whether or not a player has ever reached 2400, and Opponent, which gives the average rating of all opponents. The Lag variable is equal to zero for players who played in the latest time period, and is also zero for players who have not yet played any games. The Elite variable is required due to the K factor dependency in the FIDE implementation. The Opponent variable is not currently used in the updating algorithm.

history	A three dimensional array, or NULL if history is FALSE. The row dimension is the players, the column dimension is the time periods. The third dimension gives different parameters.
gamma	The player one advantage parameter.
kfac	The K factor or K factor function.
type	The character string "Elo".

References

Elo, Arpad (1978) The Rating of Chessplayers, Past and Present. Arco. ISBN 0-668-04721-6.

See Also

elo, kfide

Examples

```
afl <- aflodds[,c(2,3,4,7)]
robj <- fide(afl)
robj

robj <- fide(afl[afl$Week==1,])
for(i in 2:max(afl$Week)) robj <- fide(afl[afl$Week==i,], robj$ratings)
robj
```

---

glicko *The Glicko Rating System*

Description

Implements the Glicko rating system for estimating the relative skill level of players in two-player games such as chess. It extends the Elo method by including a deviation parameter for each player, representing uncertainty on the rating.

Usage

```
glicko(x, status = NULL, init = c(2200,300), gamma = 0, cval = 15,
    history = FALSE, sort = TRUE, rdmax = 350, ...)
```

Arguments

x	A data frame containing four variables: (1) a numeric vector denoting the time period in which the game took place (2) a numeric or character identifier for player one (3) a numeric or character identifier for player two and (4) the result of the game expressed as a number, typically equal to one for a player one win, zero for a player two win and one half for a draw.

status	A data frame with the current status of the system. If not NULL, this needs to be a data frame in the form of the ratings component of the returned list, containing variables named Player, Rating, Deviation, and optionally Games, Win, Draw, Loss and Lag, which are set to zero if not given.
init	The rating vector at which to initialize a new player not appearing in status. Must be a vector of length two giving the initial rating and initial deviation respectively. If different initializations for different players are required, this can be done using status. The initial deviation cannot be greater than rdmax.
gamma	A player one advantage parameter; either a single value or a numeric vector equal to the number of rows in x. Positive values favour player one, while negative values favour player two. This could represent the advantage of playing at home, or the advantage of playing white for chess. Note that this is not passed to predict.rating, which has its own gamma parameter.
cval	The c parameter, which controls the increase in the player deviations across time. Must be a single non-negative number.
history	If TRUE returns the entire history for each period in the component history of the returned list.
sort	If TRUE sort the results by rating (highest to lowest). If FALSE sort the results by player.
rdmax	The maximum value allowed for the rating deviation.
...	Not used.

Details

The Glicko rating system is a method for evaluating the skill of players. It is more complex than Elo but typically yields better predictions. Default values are roughly optimized for the chess data analyzed in the file doc/ChessRatings.pdf, using the binomial deviance criterion. A player one advantage parameter has been added to the original definition in the reference. A player one advantage parameter is also used for prediction purposes in predict.rating. In this implementation, rating deviances increase at the beginning of the updating period, and decrease at the end. This is slightly different from the Glicko-2 implementation, where deviance increases for active players may occur at the end of the previous period. In both implementations there will be an initial increase for existing but previously inactive players.

Value

A list object of class "rating" with the following components

ratings	A data frame of the results at the end of the final time period. The variables are self explanatory except for Lag, which represents the number of time periods since the player last played a game. This is equal to zero for players who played in the latest time period, and is also zero for players who have not yet played any games.
history	A three dimensional array, or NULL if history is FALSE. The row dimension is the players, the column dimension is the time periods. The third dimension gives different parameters.
gamma	The player one advantage parameter.

cval	The c parameter.
type	The character string "Glicko".

References

Glickman, M.E. (1999) Parameter estimation in large dynamic paired comparison experiments. J. R. Stat. Soc. Ser. C: Applied Statistics, 48(3), 377-394.

See Also

elo, glicko2, steph

Examples

```
afl <- aflodds[,c(2,3,4,7)]
robj <- glicko(afl)
robj

robj <- glicko(afl[afl$Week==1,])
for(i in 2:max(afl$Week)) robj <- glicko(afl[afl$Week==i,], robj$ratings)
robj
```

---

glicko2 *The Glicko-2 Rating System*

Description

Implements the Glicko-2 rating system for estimating the relative skill level of players in two-player games such as chess. It extends the Glicko method by including a volatility parameter for each player, representing the degree of expected fluctuation in the rating. Volatility is therefore a measure of consistency of performance.

Usage

```
glicko2(x, status = NULL, init = c(2200,300,0.15), gamma = 0,
    tau = 1.2, history = FALSE, sort = TRUE, rdmax = 350, ...)
```

Arguments

x	A data frame containing four variables: (1) a numeric vector denoting the time period in which the game took place (2) a numeric or character identifier for player one (3) a numeric or character identifier for player two and (4) the result of the game expressed as a number, typically equal to one for a player one win, zero for a player two win and one half for a draw.
status	A data frame with the current status of the system. If not NULL, this needs to be a data frame in the form of the ratings component of the returned list, containing variables named Player, Rating, Deviation, Volatility, and optionally Games, Win, Draw, Loss and Lag, which are set to zero if not given.

init	The rating vector at which to initialize a new player not appearing in status. Must be a vector of length three giving the initial rating, initial deviation and initial volatility respectively. If different initializations for different players are required, this can be done using status. The initial deviation cannot be greater than rdmax. The initial volatility cannot be greater than rdmax divided by 400/log(10).
gamma	A player one advantage parameter; either a single value or a numeric vector equal to the number of rows in x. Positive values favour player one, while negative values favour player two. This could represent the advantage of playing at home, or the advantage of playing white for chess. Note that this is not passed to predict.rating, which has its own gamma parameter.
tau	The tau parameter, which controls the change in the player volatility across time. Smaller values prevent the volatility measures from changing by large amounts. Must be a single number. Mark Glickman suggests a value between 0.3 and 1.2. A non-positive value can be specified, in which case the volatilities are never updated.
history	If TRUE returns the entire history for each period in the component history of the returned list.
sort	If TRUE sort the results by rating (highest to lowest). If FALSE sort the results by player.
rdmax	The maximum value allowed for the rating deviation. The maximum value allowed for the volatility is rdmax divided by 400/log(10).
...	Not used.

Details

The Glicko-2 rating system is a method for evaluating the skill of players. It is more complex than Glicko because it includes a volatility for each player. It requires a single parameter optimization for each player within each time period. We use the R function optimize in preference to the root-finding approaches suggested in Glickman (2001) and Glickman (2013). Default values are roughly optimized for the chess data analyzed in the file doc/ChessRatings.pdf, using the binomial deviance criterion. A player one advantage parameter has been added to the original definition in the reference. A player one advantage parameter is also used for prediction purposes in predict.rating.

Value

A list object of class "rating" with the following components

ratings	A data frame of the results at the end of the final time period. The variables are self explanatory except for Lag, which represents the number of time periods since the player last played a game. This is equal to zero for players who played in the latest time period, and is also zero for players who have not yet played any games.
history	A three dimensional array, or NULL if history is FALSE. The row dimension is the players, the column dimension is the time periods. The third dimension gives different parameters.
gamma	The player one advantage parameter.

tau	The tau parameter.
type	The character string "Glicko-2".

References

Glickman, M.E. (2001) Dynamic paired comparison models with stochastic variances. Journal of Applied Statistics, 28, 673-689.

Glickman, M.E. (2013) Example of the Glicko-2 system.

See Also

elo, glicko, steph

Examples

```
initstate <- data.frame(Player=1:4, Rating = c(1500,1400,1550,1700),
  Deviation = c(200,30,100,300), Volatility = 0.06)
games <- data.frame(Week = 1, Payer1 = 1, Player2 = 2:4, Score = c(1,0,0))
robj <- glicko2(games, status = initstate, tau = 0.5, sort = FALSE)
print(robj, cols = 1:4, digits = 6)

afl <- aflodds[,c(2,3,4,7)]
robj <- glicko2(afl)
robj

robj <- glicko2(afl[afl$Week==1,])
for(i in 2:max(afl$Week)) robj <- glicko2(afl[afl$Week==i,], robj$ratings)
robj
```

---

hist.rating *Histogram Plotting for a Rating Object*

Description

Plot histograms of estimated ratings or other features, including full history progressions.

Usage

```
## S3 method for class 'rating'
hist(x, which = "Rating", tng=15, history = FALSE, log = FALSE,
  xlab = which, main = paste(x$type," Ratings System"), density = FALSE,
  add = FALSE, ...)
```

Arguments

x	An object of class "rating".
which	The variable to be plotted.
tng	A single value. If the number of games played by the player is below this value, the player is not depicted on the plot.
history	If TRUE, a histogram is plotted for every single time point. Only available if the history was retained in x.
log	The log(x+1) transform. May be useful if plotting e.g. the number of games.
xlab,main	Graphical parameters.
density	If TRUE, plot a density estimate rather than a histogram.
add	Add to an existing plot? Only relevant for density estimates.
...	Other parameters to be passed through to plotting functions.

See Also

plot.rating

Examples

```
afl <- aflodds[,c(2,3,4,7)]
robj <- steph(afl)
hist(robj, xlim = c(1900,2500), density=TRUE)

afl <- aflodds[,c(2,3,4,7)]
robj <- steph(afl, history=TRUE)
hist(robj, history=TRUE, xlim = c(1900,2500), density=TRUE)

robj <- elom(riichi)
hist(robj, xlim = c(1100,1900))

robj <- elom(riichi, history=TRUE)
hist(robj, history=TRUE, xlim = c(1100,1900))
```

---

kfide *The K Factor Function Used By FIDE*

Description

Calculates the K factor for the Elo rating system based on player rating, number of games played, and optionally a binary elite player identifier.

Usage

```
kfide(rating, games, elite = NULL, kv = c(10,15,30))
```

Arguments

rating	A numeric vector of player ratings.
games	A numeric vector with the number of games played by each player.
elite	If not NULL, then a binary identifier for elite players.
kv	The three different K factors that the function can produce.

Details

This function is designed to be used for the kfac argument of either fide or elo. It returns kv[1] for elite players, kv[2] for non-elite players with 30 games or more, and kv[3] for non-elite players with less than 30 games. The default is the current FIDE implementation which uses the K factors 10, 15 and 30. The K factor of 30 was changed from 25 in the year 2011. In this context, elite players are defined by FIDE as being those who have reached the rating 2400 or more at any time in the past.

Value

A numeric vector of K factors.

See Also

fide

---

kgames *A K Factor Function With Dependence On Number Of Games*

Description

Calculates the K factor for the Elo rating system based on number of games played.

Usage

```
kgames(rating, games, elite = NULL, gv = 30, kv = c(32,26))
```

Arguments

rating	A numeric vector of player ratings. The K factor does not depend on this quantity.
games	A numeric vector with the number of games played by each player.
elite	Not used.
gv	A numeric vector of length one less than kv giving the thresholds for the number of games played.
kv	A numeric vector of length one more than gv giving the different K factors that the function can produce.

Details

This function is designed to be used for the kfac argument of either fide or elo. It returns kv[i] for players who have played a total number of games within the intervals defined by gv (closed on the right).

Value

A numeric vector of K factors.

See Also

elo, fide

---

krating *A K Factor Function With Dependence On Rating*

Description

Calculates the K factor for the Elo rating system based on the player rating.

Usage

```
krating(rating, games, elite = NULL, rv = 2300, kv = c(32,26))
```

Arguments

rating	A numeric vector of player ratings.
games	A numeric vector with the number of games played by each player. The K factor does not depend on this quantity.
elite	Not used.
rv	A numeric vector of length one less than kv giving the thresholds for the ratings.
kv	A numeric vector of length one more than gv giving the different K factors that the function can produce.

Details

This function is designed to be used for the kfac argument of either fide or elo. It returns kv[i] for players who have a rating within the intervals defined by rv (closed on the right).

Value

A numeric vector of K factors.

See Also

elo, fide

kriichi *A multi-player K Factor Function for Riichi Mahjong*

Description

Calculates the K factor for the rating system employed by Tenhou for Riichi mahjong.

Usage

```
kriichi(rating, games, gv = 400, kv = 0.2)
```

Arguments

rating	A numeric vector of player ratings. The K factor does not depend on this quantity.
games	A numeric vector with the number of games played by each player.
gv	A value giving the threshold for the number of games played.
kv	The K factor if the number of games played is greater than or equal to gv.

Details

This function is designed to be used for the kfac argument of elom. It returns kv for players who have played at least gv games, and returns 1-(1-kv)N/gv otherwise, where N is the number of games played.

Value

A numeric vector of K factors.

See Also

elom, fide

metrics *Prediction Evaluation*

Description

Returns measures that assess prediction performance.

Usage

```
metrics(act, pred, cap = c(0.01,0.99), which = 1:3, na.rm = TRUE,
  sort = TRUE, digits = 3, scale = TRUE)
```

Arguments

act	A numeric vector of actual values. Typically equal to one for a player one win, zero for a player two win, and one half for a draw.
pred	A numeric vector of predictions, typically values between zero and one. A matrix can also be given, in which case the jth column contains the predictions for model j.
cap	A numeric vector of length two giving values at which to cap the binomial deviance.
which	Select metrics using any subset of 1:3. All are produced by default.
na.rm	Remove missing values in predictions. The default is to remove missing values because the default predict method will predict missing values for games with new players.
sort	By default output is ordered from best to worst using the first metric specified.
digits	Round to this number of digits.
scale	If TRUE (the default), all metrics are scaled so that a value of 100 corresponds to predicting 0.5 for every game.

Details

The preferred metric for assessing predictions in chess is the capped binomial deviance. Mean squared error and mean absolute error metrics are also produced. By default all metrics are scaled so that the value 100 represents the zero information case. If not scaled, then all metrics are multiplied by 100.

Value

A numeric vector.

See Also

predict.rating

Examples

```
afl <- aflodds[,c(2,3,4,7)]
train <- afl[afl$Week <= 80,]
test <- afl[afl$Week > 80,]
robj <- elo(train)
metrics(test$Score, predict(robj, test))
metrics(test$Score, predict(robj, test), scale = FALSE)
```

plot.rating *Plot Player Features Across Time for a Rating Object*

Description

Plot line traces of estimated ratings or other features for selected players. This function can only be used if the full history is retained in the object x.

Usage

```
## S3 method for class 'rating'
plot(x, which = "Rating", players = NULL, t0 = 1, tv = NULL,
  npl = 10, random = FALSE, xlab = "Time Period", ylab = paste(x$type," Ratings"),
   main = paste(x$type," Ratings System"), inflation = FALSE, add=FALSE, ...)
```

Arguments

x	An object of class "rating".
which	The variable to be plotted.
players	If not NULL, should be a vector of player identifiers to explicitly select players to be plotted.
t0	The time index at which to begin. Note that unless players are specified explicitly, players who do not play at time index t0 will not be selected for the plot. Can also be a vector of length two, in which case the second value is the time index at which to end.
tv	If not NULL, then a vector of values to be used on the x-axis instead of the time index.
npl	The number of players to select.
random	If TRUE, npl players are selected at random from those who played at time t0. If FALSE (the default), the npl players who played most games at t0 are selected. Ignored if players is not NULL.
xlab,ylab,main	Graphical parameters.
inflation	If TRUE, plot the average rating of the best npl players at each time point. This is designed to investigate ratings inflation.
add	Add to an existing plot.
...	Other parameters to be passed through to plotting functions.

Details

Note that the argument random is not used by default, since it can produce flat profiles from randomly selected players who play few games. The default selection is non-random and selects more active players, however they may be more likely to improve over time than the general population.

See Also

hist.rating

Examples

```
afl <- aflodds[,c(2,3,4,7)]
robj <- steph(afl, history=TRUE)
plot(robj)

robj <- elom(riichi, history = TRUE)
pl <- robj$ratings$Player[robj$ratings$Games >= 80]
plot(robj, players = pl)
```

---

predict.rating *Predict Result Of Games Based On Player Ratings*

Description

Predict the result of two-player or multi-player games, given the estimated ratings for each player.

Usage

```
## S3 method for class 'rating'
predict(object, newdata, tng=15, trat=NULL, gamma=30,
  thresh, placing = FALSE, ...)
```

Arguments

object	An object of class "rating".
newdata	For two player games, a dataframe containing three variables: (1) a numeric vector denoting the time period in which the game is taking place (2) a numeric or character identifier for player one (3) a numeric or character identifier for player two. The time period can contain missing values as it is not used for the prediction. For N-player games (i.e. for objects created by the elom function), the player identifiers should be in columns 2 to N-1. This argument cannot be missing; if predictions on the original dataset are required, then this dataset must be passed to the prediction function.
tng	A single value. If the number of games played by any player is below this value, then either the prediction will be a missing value, or the prediction will be based on trat.
trat	A single number (for Elo and EloM), or a vector of length two (for Glicko or Glicko-2 or Stephenson) giving the rating and deviation parameters to be used for players who have played less than tng games. If NULL then these predictions will be missing. The volatility parameter in Glicko-2 is not needed for predictions.

gamma	A player one advantage parameter; either a single value or a numeric vector equal to the number of rows in newdata. Positive values favour player one, while negative values favour player two. This could represent the advantage of playing at home, or the advantage of playing white for chess. The default value is roughly optimal for chess ratings. Ignored for multi-player.
thresh	A single value. If given, a binary vector is returned indicating whether the prediction is greater than this value. Ignored for multi-player.
placing	For multi-player only. If TRUE, predicted placings are given rather than expected base scores.
...	Not used.

Details

The function predicts the expectation of the game result. In two-player games, if the value of one is a win for player one, and the value of zero is a win for player two, and there are no other possibilities, then the prediction is the probability of a win for player one. This is not the case when draws are a possibility.

For multi-player predictions using objects produced by elom, expected base scores are given for each player. These are simply the difference between the rating of a player and the average of all players in the game, divided by 40.

Value

A numeric vector of predictions of two-player games, or a matrix of expected base scores for multi-player games, either of which may contain missing values.

See Also

elo, elom, metrics

Examples

```
afl <- aflodds[,c(2,3,4,7)]
train <- afl[afl$Week <= 80,]
test <- afl[afl$Week > 80,]
robj <- elo(train)
pvals <- predict(robj, test)

train <- riichi[riichi$Time <= 250,]
test <- riichi[riichi$Time > 250,]
robj <- elom(train)
predict(robj, test, trat = 1400, placing = TRUE)
```

`riichi` *Riichi Mahjong Game Results*

Description

The `aflodds` data frame has 540 rows and 9 variables. It shows the results for 540 Riichi Mahjong games played by 69 players at the Melbourne Mahjong Club in 2019. Player identifiers are randomly assigned.

Usage

```
aflodds
```

Format

This data frame contains the following columns:

Time The day number within the year 2019.

Play1 Player 1 identifier.

Play2 Player 2 identifier.

Play3 Player 3 identifier.

Play4 Player 4 identifier.

Score1 Player 1 score.

Score2 Player 2 score.

Score3 Player 3 score.

Score4 Player 4 score.

Details

Players start the game with 25000 points. Mahjong is a zero-sum game, therfore the sum of all four scores is always 100000. Negative scores are possible. The largest recorded score is 93900. The smallest recorded score is -24600.

Source

Hand collected by The Melbourne Mahjong Club.

steph *The Stephenson Rating System*

Description

Implements the Stephenson rating system for estimating the relative skill level of players in two-player games such as chess. It extends the Glicko method by including a second parameter controlling player deviation across time, a bonus parameter, and a neighbourhood parameter.

Usage

```
steph(x, status = NULL, init = c(2200,300), gamma = 0, cval = 10,
  hval = 10, bval = 0, lambda = 2, history = FALSE, sort = TRUE,
  rdmax = 350, ...)
```

Arguments

x	A data frame containing four variables: (1) a numeric vector denoting the time period in which the game took place (2) a numeric or character identifier for player one (3) a numeric or character identifier for player two and (4) the result of the game expressed as a number, typically equal to one for a player one win, zero for a player two win and one half for a draw.
status	A data frame with the current status of the system. If not NULL, this needs to be a data frame in the form of the ratings component of the returned list, containing variables named Player, Rating, Deviation, and optionally Games, Win, Draw, Loss and Lag, which are set to zero if not given.
init	The rating vector at which to initialize a new player not appearing in status. Must be a vector of length two giving the initial rating and initial deviation respectively. If different initializations for different players are required, this can be done using status. The initial deviation cannot be greater than rdmax.
gamma	A player one advantage parameter; either a single value or a numeric vector equal to the number of rows in x. Positive values favour player one, while negative values favour player two. This could represent the advantage of playing at home, or the advantage of playing white for chess. Note that this is not passed to predict.rating, which has its own gamma parameter.
cval	The c parameter, which controls the increase in the player deviations across time. Must be a single non-negative number. Note that both cval and hval increase player deviations, so if hval is not zero then cval should typically be lower than the corresponding parameter in glicko.
hval	The h parameter, which also controls the increase in the player deviations across time. Must be a single non-negative number.
bval	The bonus parameter, which gives a per game bonus to each player on the basis that players who play more often may improve irrespective of whether they win or lose. A single non-negative number. Note that this will create ratings inflation (i.e. ratings will increase over time).

lambda	The neighbourhood parameter, which shrinks player ratings towards their opponents. A single non-negative number.
history	If TRUE returns the entire history for each period in the component history of the returned list.
sort	If TRUE sort the results by rating (highest to lowest). If FALSE sort the results by player.
rdmax	The maximum value allowed for the rating deviation.
...	Not used.

Details

The Stephenson rating system is a method for evaluating the skill of players. It was developed by Alec Stephenson in 2012 as a variant of his winning entry in a competition to find the most useful practical chess rating system, organized by Jeff Sonas on Kaggle, a platform for data prediction competitions. The precise details are given in the file doc/ChessRatings.pdf.

This implementation is written so that Glicko is obtained as a special case upon setting all of the parameters hval, bval and lambda to zero. Default values are roughly optimized for the chess data analyzed in the file doc/ChessRatings.pdf, using the binomial deviance criterion.

Value

A list object of class "rating" with the following components

ratings	A data frame of the results at the end of the final time period. The variables are self explanatory except for Lag, which represents the number of time periods since the player last played a game. This is equal to zero for players who played in the latest time period, and is also zero for players who have not yet played any games.
history	A three dimensional array, or NULL if history is FALSE. The row dimension is the players, the column dimension is the time periods. The third dimension gives different parameters.
gamma	The player one advantage parameter.
cval	The c parameter.
hval	The h parameter.
bval	The bonus parameter.
lambda	The neighbourhood parameter.
type	The character string "Stephenson".

References

Glickman, M.E. (1999) Parameter estimation in large dynamic paired comparison experiments. J. R. Stat. Soc. Ser. C: Applied Statistics, 48(3), 377-394.

Glickman, M.E. (2001) Dynamic paired comparison models with stochastic variances. Journal of Applied Statistics, 28, 673-689.

See Also

glicko

Examples

```
afl <- aflodds[,c(2,3,4,7)]
robj <- steph(afl)
robj

robj <- steph(afl[afl$Week==1,])
for(i in 2:max(afl$Week)) robj <- steph(afl[afl$Week==i,], robj$ratings)
robj
```

Index