Package ‘elo’

January 14, 2020

Title  Elo Ratings
Version  2.1.1
Date  2020-01-14
Description  A flexible framework for calculating Elo ratings and resulting rankings of any two-team-per-matchup system (chess, sports leagues, 'Go', etc.). This implementation is capable of evaluating a variety of matchups, Elo rating updates, and win probabilities, all based on the basic Elo rating system. It also includes methods to benchmark performance, including logistic regression and Markov chain models.

Depends  R (>= 3.3.0), stats
Imports  Rcpp, pROC
Suggests  knitr, testthat, rmarkdown
VignetteBuilder  knitr
License  GPL (>= 2)

URL  https://github.com/eheinzen/elo,
     https://cran.r-project.org/package=elo,
     https://eheinzen.github.io/elo/

BugReports  https://github.com/eheinzen/elo/issues
RoxygenNote  7.0.2
LazyData  true
LinkingTo  Rcpp
Encoding  UTF-8
NeedsCompilation  yes

Author  Ethan Heinzen [aut, cre]
Maintainer  Ethan Heinzen <heinzen.ethan@mayo.edu>
Repository  CRAN
Date/Publication  2020-01-14 17:00:03 UTC
R topics documented:

auc.elo .................................................. 2
elo .......................................................... 3
eolo.calc .................................................. 4
eolo.colley ............................................... 5
eolo glm ................................................... 6
eolo.markovchain ....................................... 8
eolo.model.frame ....................................... 9
eolo mov ................................................. 10
eolo mse .................................................. 11
eolo prob ................................................ 12
eolo run .................................................. 13
eolo run.helpers ....................................... 15
eolo update ............................................... 16
eolo winpct ............................................... 17
favored.elo ............................................. 18
fitted.elo ............................................... 19
players .................................................. 20
predict.elo ............................................. 21
rank.teams ............................................. 23
score .................................................... 24
summary.elo ........................................... 25
tournament ............................................. 26

Index 27

---

auc.elo  Calculate AUC on an elo.run object

Description

Calculate AUC on an elo.run object

Usage

## S3 method for class 'elo.run'
auc(object, ...)

## S3 method for class 'elo.glm'
auc(object, ...)

## S3 method for class 'elo.running'
auc(object, running = TRUE, ...)

## S3 method for class 'elo.markovchain'
auc(object, ...)


## S3 method for class 'elo.winpct'
auc(object, ...)

## S3 method for class 'elo.colley'
auc(object, ...)

**Arguments**

object  
An object of class `elo.run`.  

...  
Other arguments (not used at this time).  

running  
logical, denoting whether to use the running predicted values.

**Value**

The AUC of the predicted Elo probabilities and the actual win results.

**References**


**See Also**

`pROC::auc`, `elo.run`.

---

**elo**  
*The Elo Package*

**Description**

An implementation of Elo ratings for general use in 'R'.

**Functions**

Listed below are the most useful functions available in `elo`:

- `elo.prob`: Calculate the probability that team A beats team B.  
- `elo.update`: Calculate the update value for a given Elo matchup.  
- `elo.calc`: Calculate post-update Elo values.  
- `elo.run`: Calculate Ellos for a series of matches.  
- `score`: Create a 1/0/0.5 win "indicator" based on two teams' scores.

**Data**

- `tournament`: Mock data for examples.
References

Examples

library(elo)

elo.calc(wins.A, ...)

## Default S3 method:

## S3 method for class 'formula'
elo.calc(formula, data, na.action, subset, k = NULL, ...)

Arguments

wins.A Numeric vector of wins by team A.
...
Other arguments (not in use at this time).
k A constant k-value (or a vector, where appropriate).
formula A formula. See the help page for formulas for details.
data A data.frame in which to look for objects in formula.
na.action A function which indicates what should happen when the data contain NAs.
subset An optional vector specifying a subset of observations.

Value
A data.frame with two columns, giving the new Elo values after each update.

See Also
elo.prob, elo.update, elo.model.frame
Examples
elo.calc(c(1, 0), c(1500, 1500), c(1500, 1600), k = 20)

dat <- data.frame(wins.A = c(1, 0), elo.A = c(1500, 1500),
                  elo.B = c(1500, 1600), k = c(20, 20))
elo.calc(wins.A ~ elo.A + elo.B + k(k), data = dat)

Description
Compute a Colley matrix model for a matchup.

Usage
elo.colley(
  formula,
  data,
  family = "binomial",
  weights,
  na.action,
  subset,
  k = 1,
  ...,
  running = FALSE,
  skip = 0
)

Arguments

- **formula**: A formula. See the help page for formulas for details.
- **data**: A data.frame in which to look for objects in formula.
- **family**: Arguments passed to glm.
- **weights**: A vector of weights. Note that these weights are used in the Colley matrix creation, but not the regression.
- **na.action**: A function which indicates what should happen when the data contain NAs.
- **subset**: An optional vector specifying a subset of observations.
- **k**: The fraction of a win to be assigned to the winning team. See "details".
- **...**: Arguments passed to glm.
- **running**: Logical, denoting whether to calculate "running" projected probabilities. If true, a model is fit for group 1 on its own to predict group 2, then groups 1 and 2 to predict 3, then groups 1 through 3 to predict 4, etc. Groups are determined in formula. Omitting a group term re-runs a glm model to predict each observation (a potentially time-consuming operation!)
skip Integer, denoting how many groups to skip before fitting the running models. This is helpful if groups are small, where glm would have trouble converging for the first few groups. The predicted values are then set to 0.5 for the skipped groups.

Details

See the vignette for details on this method. The differences in assigned scores (from the coefficients of the Colley matrix regression) are fed into a logistic regression model to predict wins or (usually) a linear model to predict margin of victory. In this setting, 'k' indicates the fraction of a win to be assigned to the winning team (and the fraction of a loss to be assigned to the losing team); setting k = 1 (the default) emits the "Bias Free" ranking method presented by Colley. It is also possible to adjust the regression by setting the second argument of adjust(). As in elo.glm, the intercept represents the home-field advantage. Neutral fields can be indicated using the neutral() function, which sets the intercept to 0.

References


See Also

glm, summary.elo.colley, score.mov, elo.model.frame

Examples


elo.glm elo.glm

Description

Compute a (usually logistic) regression model for a matchup.

Usage

eolo.glm(
    formula,
    data,
    family = "binomial",
    weights,
    na.action,
    subset,
    ..., 
    running = FALSE,
    skip = 0
)
Arguments

- **formula**: A formula. See the help page for formulas for details.
- **data**: A data.frame in which to look for objects in formula.
- **family**, **weights**, **...**: Arguments passed to glm.
- **na.action**: A function which indicates what should happen when the data contain NAs.
- **subset**: An optional vector specifying a subset of observations.
- **running**: Logical, denoting whether to calculate “running” projected probabilities. If true, a model is fit for group 1 on its own to predict group 2, then groups 1 and 2 to predict 3, then groups 1 through 3 to predict 4, etc. Groups are determined in formula. Omitting a group term re-runs a glm model to predict each observation (a potentially time-consuming operation!)
- **skip**: Integer, denoting how many groups to skip before fitting the running models. This is helpful if groups are small, where glm would have trouble converging for the first few groups. The predicted values are then set to 0.5 for the skipped groups.

Details

The formula syntax is the same as other elo functions. A data.frame of indicator variables is built, where an entry is 1 if a team is home, 0 if a team didn’t play, and -1 if a team is a visitor. Anything passed to adjust() in formula is also put in the data.frame. A glm model is then run to predict wins or margin of victory.

With this setup, the intercept represents the home-field advantage. Neutral fields can be indicated using the neutral() function, which sets the intercept to 0.

Note that any weights specified in players() will be ignored.

Value

An object of class c("elo.glm","glm"). If running==TRUE, the class "elo.glm.running" is prepended.

See Also

- glm, summary.elo.glm, score, mov, elo.model.frame

Examples

```r
data(tournament)
elo.glm(mov(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = tournament, family = "gaussian")
```
elo.markovchain

Description

Compute a Markov chain model for a matchup.

Usage

elo.markovchain(
  formula,
  data,
  family = "binomial",
  weights,
  na.action,
  subset,
  k = NULL,
  ..., 
  running = FALSE,
  skip = 0
)

Arguments

formula A formula. See the help page for formulas for details.
data A data.frame in which to look for objects in formula.
family Arguments passed to glm.
weights A vector of weights. Note that these weights are used in the Markov Chain model, but not the regression.
na.action A function which indicates what should happen when the data contain NAs.
subset An optional vector specifying a subset of observations.
k The probability that the winning team is better given that they won. See details.
... Arguments passed to glm.
running Logical, denoting whether to calculate "running" projected probabilities. If true, a model is fit for group 1 on its own to predict group 2, then groups 1 and 2 to predict 3, then groups 1 through 3 to predict 4, etc. Groups are determined in formula. Omitting a group term re-runs a glm model to predict each observation (a potentially time-consuming operation!)
skip Integer, denoting how many groups to skip before fitting the running models. This is helpful if groups are small, where glm would have trouble converging for the first few groups. The predicted values are then set to 0.5 for the skipped groups.
Details

See the vignette for details on this method. The probabilities we call 'k' purely for convenience. The differences in assigned scores (from the stationary distribution pi) are fed into a logistic regression model to predict wins or (usually) a linear model to predict margin of victory. It is also possible to adjust the regression by setting the second argument of adjust(). As in elo.glm, the intercept represents the home-field advantage. Neutral fields can be indicated using the neutral() function, which sets the intercept to 0.

Note that by assigning probabilities in the right way, this function emits the Logistic Regression Markov Chain model (LRMC).

References


See Also

glm, summary.elo.markovchain, score.mv, elo.model.frame

Examples

elo.markovchain(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = tournament, subset = points.Home != points.Visitor, k = 0.7)

elo.markovchain(mov(points.Home, points.Visitor) ~ team.Home + team.Visitor, family = "gaussian", data = tournament, k = 0.7)

Elo.model.frame

Interpret formulas in elo functions

Description

A helper function to create the model.frame for many elo functions.

Usage

elo.model.frame(
  formula,
  data,
  na.action,
  subset,
  k = NULL,
  ...,
  required.vars = "elos",
  warn.k = TRUE,
  ncol.k = 1
)
Arguments

- **formula**: A formula. See the help page for formulas for details.
- **data**: A data.frame in which to look for objects in formula.
- **na.action**: A function which indicates what should happen when the data contain NAs.
- **subset**: An optional vector specifying a subset of observations.
- **k**: A constant k-value (or a vector, where appropriate).
- **...**: Other arguments (not in use at this time).
- **required.vars**: One or more of c("wins", "elos", "k", "group", "regress"), denoting which variables are required to appear in the final model.frame.
- **warn.k**: Should a warning be issued if k is specified as an argument and in formula?
- **ncol.k**: How many columns (NCOL) should k have?

See Also

e-lo.run, elo.calc, elo.update, elo.prob

e-lo.mov

Create a "margin of victory" column

Description

Create a "margin of victory" based on two teams' scores

Usage

e-lo.mov(score.A, score.B = 0)

Arguments

- **score.A**: Numeric; the score of the first team. Alternatively, this can be a pre-computed margin of victory which will get compared to 0.
- **score.B**: Numeric; the score of the second team; default is 0, in case score.A is already a margin of victory.

Value

An object with class "elo.mov", denoting score.A = score.B.

See Also

e-lo

Examples

- mov(12, 10)
- mov(10, 10)
- mov(10, 12)
elo.mse

Calculate the mean square error

Description

Calculate the mean square error (Brier score) for a model.

Usage

brier(object, subset, ...)

mse(object, subset, ...)

## S3 method for class 'elo.run'
mse(object, subset, ...)

## S3 method for class 'elo.glm'
mse(object, subset, ...)

## S3 method for class 'elo.running'
mse(object, subset, running = TRUE, ...)

## S3 method for class 'elo.markovchain'
mse(object, subset, ...)

## S3 method for class 'elo.winpct'
mse(object, subset, ...)

## S3 method for class 'elo.colley'
mse(object, subset, ...)

Arguments

object An object
subset (optional) A vector of indices on which to calculate the MSE.
... Other arguments (not in use at this time).
running logical, denoting whether to use the running predicted values.

Details

Even though logistic regressions don’t use the MSE on the y=0/1 scale, it can still be informative. Note that the S3 method is mse.
elo.prob

Elo functions

Description
Calculate the probability that team A beats team B. This is vectorized.

Usage
elo.prob(elo.A, ...)

## Default S3 method:
elo.prob(elo.A, elo.B, ..., elos = NULL, adjust.A = 0, adjust.B = 0)

## S3 method for class 'formula'
elo.prob(formula, data, na.action, subset, ..., elos = NULL)

Arguments
elo.A, elo.B Numeric vectors of elo scores, or else vectors of teams.
... Other arguments (not in use at this time).
elos An optional named vector containing Elo ratings for all teams in formula or elo.A and elo.B.
formula A formula. See the help page for formulas for details.
data A data.frame in which to look for objects in formula.
na.action A function which indicates what should happen when the data contain NAs.
subset An optional vector specifying a subset of observations.

Details
Note that formula can be missing the wins.A component. If present, it’s ignored by elo.model.frame.

Value
A vector of Elo probabilities.

See Also
elo.update, elo.calc, elo.model.frame
Examples

 elo.prob(1500, 1500)
 elo.prob(c(1500, 1500), c(1500, 1600))

dat <- data.frame(wins.A = c(1, 0), elo.A = c(1500, 1500),
                    elo.B = c(1500, 1600), k = c(20, 20))
 elo.prob(~ elo.A + elo.B, data = dat)

## Also works to include the wins and k:
 elo.prob(wins.A ~ elo.A + elo.B + k(k), data = dat)

## Also allows teams
 elo.prob(c("A", "B"), c("C", "C"), elos = c(A = 1500, B = 1600, C = 1500))

Description

Calculate Elos for a series of matches.

Usage

 elo.run(formula, data, na.action, subset, k = NULL, initial.elos = NULL, ...)

 elo.run2(
   formula, data, subset, k = NULL, initial.elos = NULL,
   ..., prob.fun = elo.prob, update.fun = elo.update
 )

Arguments

 formula A formula. See the help page for formulas for details.
 data A data.frame in which to look for objects in formula.
 na.action A function which indicates what should happen when the data contain NAs.
 subset An optional vector specifying a subset of observations.
 k A constant k-value (or a vector, where appropriate).
 initial.elos An optional named vector containing initial Elo ratings for all teams in formula.
... Other arguments (not used at this time).

prob.fun A function with at least 4 arguments: elo.A, elo.B, adjust.A, and adjust.B. It should return a predicted probability that team A wins. The values passed in will be scalars, and a scalar is expected as output.

update.fun A function with at least 6 arguments: the same as elo.update.default. The function takes in the Elos, the win indicator, k, and any adjustments, and returns a value by which to update the Elos. The values passed in will be scalars, and a scalar is expected as output.

Details

elo.run and elo.run2 by default return the exact same thing. elo.run uses C++ and may be up to 50 times faster, while elo.run2 uses R but also supports custom update functions. Prefer the first unless you really need a custom update function.

Value

An object of class "elo.run" or class "elo.run.regressed".

See Also

score, elo.run.helperselo.run helpers, elo.calc, elo.update, elo.prob, elo.model.frame.

Examples

data(tournament)
elo.run(score(points.Home, points.Visitor) ~ team.Home + team.Visitor,
data = tournament, k = 20)

# Create non-constant 'k'
elo.run(score(points.Home, points.Visitor) ~ team.Home + team.Visitor +
k(20*log(abs(points.Home - points.Visitor) + 1)), data = tournament)

# Adjust Elo for, e.g., home-field advantage
elo.run(score(points.Home, points.Visitor) ~ adjust(team.Home, 30) + team.Visitor,
data = tournament, k = 20)

tournament$home.field <- 30
elo.run(score(points.Home, points.Visitor) ~ adjust(team.Home, home.field) + team.Visitor,
data = tournament, k = 20)

# Regress the Elos back toward 1500 at the end of the half-season
elo.run(score(points.Home, points.Visitor) ~ adjust(team.Home, 30) +
team.Visitor + regress(half, 1500, 0.2), data = tournament, k = 20)
Description

as.matrix converts an Elo object into a matrix of running Elos.

Usage

## S3 method for class 'elo.run'
as.matrix(x, ..., group = x$group)

## S3 method for class 'elo.run.regressed'
as.matrix(x, ..., group = x$group)

## S3 method for class 'elo.run'
as.data.frame(x, ...)

final.elos(x, ...)

## S3 method for class 'elo.run'
final.elos(x, ...)

## S3 method for class 'elo.run.regressed'
final.elos(x, regressed = FALSE, ...)

Arguments

x  An object of class "elo.run" or class "elo.run.regressed".

...  Other arguments (Not in use at this time).

group  A grouping vector, telling which rows to output in the matrix.

regressed  Logical, denoting whether to use the post-regressed (TRUE) or pre-regressed (FALSE) final Elos. Note that TRUE only makes sense when the final Elos were regressed one last time (i.e., if the last element of the regress() vector yields TRUE).

Details

as.data.frame converts the "elos" component of an object from elo.run into a data.frame.

final.elos is a generic function to extract the last Elo per team.

Value

A matrix, a data.frame, or a named vector.
See Also

elo.run

Examples

e <- elo.run(score(points.Home, points.Visitor) ~ team.Home + team.Visitor + group(week),
data = tournament, k = 20)
head(as.matrix(e))
str(as.data.frame(e))
final.elos(e)

description

Calculate the update value for a given Elo matchup. This is used in elo.calc, which reports the
post-update Elo values. This is vectorized.

Usage

elo.update(wins.A, ...)

## Default S3 method:

## S3 method for class
quotesingle.Var
elo.update(formula, data, na.action, subset, k = NULL, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wins.A</td>
<td>Numeric vector of wins by team A.</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments (not in use at this time).</td>
</tr>
<tr>
<td>elo.A</td>
<td>Numeric vectors of elo scores.</td>
</tr>
<tr>
<td>elo.B</td>
<td>Numeric vectors of elo scores.</td>
</tr>
<tr>
<td>k</td>
<td>A constant k-value (or a vector, where appropriate).</td>
</tr>
<tr>
<td>formula</td>
<td>A formula. See the help page for formulas for details.</td>
</tr>
<tr>
<td>data</td>
<td>A data.frame in which to look for objects in formula.</td>
</tr>
<tr>
<td>na.action</td>
<td>A function which indicates what should happen when the data contain NAs.</td>
</tr>
<tr>
<td>subset</td>
<td>An optional vector specifying a subset of observations.</td>
</tr>
</tbody>
</table>
**elo.winpct**

**Value**
A vector of Elo updates.

**See Also**
elo.prob, elo.calc, elo.model.frame

**Examples**
elo.update(c(1, 0), c(1500, 1500), c(1500, 1600), k = 20)
dat <- data.frame(wins.A = c(1, 0), elo.A = c(1500, 1500), elo.B = c(1500, 1600), k = c(20, 20))elo.update(wins.A ~ elo.A + elo.B + k(k), data = dat)

---

**elo.winpct elo.winpct**

**Description**
Compute a (usually logistic) regression based on win percentage for a matchup.

**Usage**
elo.winpct(
  formula,
data,
family = "binomial",
weights,
na.action,
subset,
...,running = FALSE,
skip = 0
)

**Arguments**
formula A formula. See the help page for formulas for details.
data A data.frame in which to look for objects in formula.
family Arguments passed to glm.
weights A vector of weights. Note that these are used in calculating wins and losses but not in the regression.
na.action A function which indicates what should happen when the data contain NAs.
subset An optional vector specifying a subset of observations.
... Arguments passed to glm.
running Logical, denoting whether to calculate "running" projected probabilities. If true, a model is fit for group 1 on its own to predict group 2, then groups 1 and 2 to predict 3, then groups 1 through 3 to predict 4, etc. Groups are determined in formula. Omitting a group term re-runs a glm model to predict each observation (a potentially time-consuming operation!)

skip Integer, denoting how many groups to skip before fitting the running models. This is helpful if groups are small, where glm would have trouble converging for the first few groups. The predicted values are then set to 0.5 for the skipped groups.

Details

Win percentages are first calculated. Anything passed to adjust() in formula is also put in the data.frame. A glm model is then run to predict wins or margin of victory.

With this setup, the intercept represents the home-field advantage. Neutral fields can be indicated using the neutral() function, which sets the intercept to 0.

See Also

glm, summary.elo.winpct, score.mov, elo.model.frame

Examples


elo.winpct(mov(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = tournament, family = "gaussian")

favored.elo

Classify teams that are favored to win

Description

Classify teams that are favored to win

Usage

favored(x, ...)

## S3 method for class 'elo.run'
favored(x, ...)

## S3 method for class 'elo.glm'
favored(x, ...)

## S3 method for class 'elo.running'
favored(x, running = TRUE, ...)

## S3 method for class 'elo.markovchain'
favored(x, ...)

## S3 method for class 'elo.winpct'
favored(x, ...)

## S3 method for class 'elo.colley'
favored(x, ...)

## Default S3 method:
favored(x, p.A, ...)

Arguments

x

An object from elo.run or elo.glm, or for the default method a vector representing wins.A.

... Other arguments (not in use at this time).

running logical, denoting whether to use the running predicted values.

p.A A vector of predicted win probabilities.

fitted.elo Extract model values

Description

Extract model values from elo functions.

Usage

## S3 method for class 'elo.run'
fitted(object, ...)

## S3 method for class 'elo.run'
residuals(object, ...)

## S3 method for class 'elo.running'
fitted(object, running = TRUE, ...)

## S3 method for class 'elo.glm'
fitted(object, ...)

## S3 method for class 'elo.markovchain'
fitted(object, ...)
## S3 method for class 'elo.winpcct'
fitted(object, ...)

## S3 method for class 'elo.colley'
fitted(object, ...)

### Arguments

- **object**: An object.
- **...**: Other arguments
- **running**: logical, denoting whether to use the running predicted values.

---

### Description

Details on Elo formulas and the special functions allowed in them to change functions’ behaviors.

### Usage

- `players(..., weights = NULL)`
- `k(x, y = NULL)`
- `adjust(x, adjustment)`
- `regress(x, to, by, regress.unused = TRUE)`
- `group(x)`
- `neutral(x)`

### Arguments

- **...**: Vectors to be coerced to character, which comprise of the players of a team.
- **weights**: A vector giving the weights of Elo updates for the players in .... Ignored for `elo.glm`.
- **x, y**: A vector.
- **adjustment**: A single value or a vector of the same length as x: how much to adjust the Elos in x.
- **to**: Numeric: what Elo to regress to. Can be a single value or named vector the same length as the number of teams.
- **by**: Numeric: by how much should Elos be regressed toward to.
- **regress.unused**: Logical: whether to continue regressing teams which have stopped playing.
Details

In the functions in this package, formula is usually of the form wins.A ~ elo.A + elo.B, where elo.A and elo.B are vectors of Elos, and wins.A is between 0 and 1, denoting whether team A (Elo A) won or lost (or something between). elo.prob also allows elo.A and elo.B to be character or factors, denoting which team(s) played. elo.run requires elo.A to be a vector of teams or a players matrix from players() (sometimes denoted by "team.A"), but elo.B can be either a vector of teams or players matrix ("team.B") or else a numeric column (denoting a fixed-Elo opponent). elo.glm requires both to be a vector of teams or players matrix. elo.markovchain requires both to be a vector of teams.

formula accepts six special functions in it:

- k() allows for complicated Elo updates. For constant Elo updates, use the k = argument instead of this special function. Note that elo.markovchain uses this function (or argument) as a convenient way of specifying transition probabilities. elo.colley uses this to indicate the fraction of a win to be assigned to the winning team.

- adjust() allows for Elos to be adjusted for, e.g., home-field advantage. The second argument to this function can be a scalar or vector of appropriate length. This can also be used in elo.glm and elo.markovchain as an adjuster to the logistic regressions.

- regress() can be used to regress Elos back to a fixed value after certain matches. Giving a logical vector identifies these matches after which to regress back to the mean. Giving any other kind of vector regresses after the appropriate groupings (see, e.g., duplicated(...,fromLast = TRUE)). The other three arguments determine what Elo to regress to (to =), by how much to regress toward that value (by =), and whether to continue regressing teams which have stopped playing (regress.unused, default = TRUE).

- group() is used to group matches (by, e.g., week). It is fed to as.matrix.elo.run to produce only certain rows of matrix output. It also determines how many models to run (and on what data) for elo.glm and elo.markovchain when running=TRUE.

- neutral() is used in elo.glm and elo.markovchain to determine the intercept. In short, the intercept is 1 -neutral(), denoting home-field advantage. Therefore, the column passed should be 0 (denoting home-field advantage) or 1 (denoting a neutral game). If omitted, all matches are assumed to have home field advantage.

- players() is used for multiple players on a team contributing to an overall Elo. The Elo updates are then assigned based on the specified weights. The weights are ignored in elo.glm.
predict.elo

Usage

```r
## S3 method for class 'elo.run'
predict(object, newdata, ...)

## S3 method for class 'elo.run.regressed'
predict(object, newdata, regressed = FALSE, ...)

## S3 method for class 'elo.glm'
predict(object, newdata, type = "response", ...)

## S3 method for class 'elo.running'
predict(object, newdata, running = TRUE, ...)

## S3 method for class 'elo.markovchain'
predict(object, newdata, ...)

## S3 method for class 'elo.colley'
predict(object, newdata, ...)

## S3 method for class 'elo.winpct'
predict(object, newdata, ...)
```

Arguments

- `object` An model from which to get predictions.
- `newdata` A new dataset containing the same variables as the call that made `object`. If missing, the predicted win probabilities from `object` will be returned.
- `...` Other arguments.
- `regressed` See the note on `final.elos`.
- `type` See `predict.glm`
- `running` logical, denoting whether to use the running predicted values. Only makes sense if `newdata` is missing.

Details

Note that the "elo.glm.running" objects will use a model fit on all the data to predict.

Value

A vector of win probabilities.

Examples

```r
data(tournament)
t1 <- head(tournament, -3)
t2 <- tail(tournament, 3)
results <- elo.run(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, 
data = t1, k = 20)
```
rank.teams

predict(results)
predict(results, newdata = t2)

results <- elo.glm(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = t1,
                     subset = points.Home != points.Visitor)
predict(results)
predict(results, newdata = t2)

results <- elo.markovchain(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = t1,
                           subset = points.Home != points.Visitor, k = 0.7)
predict(results)
predict(results, newdata = t2)

results <- elo.colley(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = t1,
                       subset = points.Home != points.Visitor)
predict(results)
predict(results, newdata = t2)

results <- elo.winpct(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = t1,
                      subset = points.Home != points.Visitor, k = 0.7)
predict(results)
predict(results, newdata = t2)

---

**rank.teams**

**Rank teams**

**Description**

Extract the rankings from Elo objects.

**Usage**

```r
rank.teams(object, ties.method = "min", ...)
```

## S3 method for class 'elo.run'

```r
rank.teams(object, ties.method = "min", ...)
```

## S3 method for class 'elo.run.regressed'

```r
rank.teams(object, ties.method = "min", regressed = FALSE, ...)
```

## S3 method for class 'elo.glm'

```r
rank.teams(object, ties.method = "min", ...)
```

## S3 method for class 'elo.markovchain'

```r
rank.teams(object, ties.method = "min", ...)
```

## S3 method for class 'elo.winpct'

```r
rank.teams(object, ties.method = "min", ...)
```
## S3 method for class 'elo.colley'
rank.teams(object, ties.method = "min", ...)

Arguments

- **object**: An object.
- **ties.method**: Passed to `rank`.
- **...**: Other arguments
- **regressed**: Passed to `final.elos`.

---

### score

Create a 1/0/0.5 win "indicator"

#### Description

Create a 1/0/0.5 win "indicator" based on two teams' scores, and test for "score-ness".

#### Usage

```r
score(score.A, score.B)
```

```r
is.score(x)
```

#### Arguments

- **score.A**: Numeric; the score of the first team (whose wins are to be denoted by 1).
- **score.B**: Numeric; the score of the second team (whose wins are to be denoted by 0).
- **x**: An R object.

#### Value

For `score`, a vector containing 0, 1, and 0.5 (for ties). For `is.score`, TRUE or FALSE depending on whether all values of `x` are between 0 and 1 (inclusive).

#### See Also

- `score`

#### Examples

- `score(12, 10)`
- `score(10, 10)`
- `score(10, 12)`
**Summary.elo**

**Summarize an elo Object**

**Description**
Summarize an elo Object

**Usage**

```r
## S3 method for class 'elo.run'
summary(object, ...)

## S3 method for class 'elo.glm'
summary(object, ...)

## S3 method for class 'elo.markovchain'
summary(object, ...)

## S3 method for class 'elo.colley'
summary(object, ...)

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

**Arguments**

- `object` An object to summarize.
- `...` Other arguments

**Value**
A summary of object.

**See Also**
`favored, auc.elo.run, mse`

**Examples**

```r
summary(elo.run(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, 
data = tournament, k = 20))
summary(elo.glm(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, 
data = tournament))
m <- elo.markovchain(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, 
data = tournament, subset = points.Home != points.Visitor, k = 0.7)
s <- elo.colley(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, 
data = tournament, subset = points.Home != points.Visitor)
```
```
summary(co)
wp <- elo.winpct(score(points.Home, points.Visitor) ~ team.Home + team.Visitor,
  data = tournament, subset = points.Home != points.Visitor, k = 0.7)
summary(wp)
```

tournament: Mock data for examples

Description

A fake dataset containing results from "animal-ball" matches.

Format

A data frame with 56 observations on the following 4 variables:

- `team.Home` The home team for the match
- `team.Visitor` The visiting team for the match
- `points.Home` Number of points scored by the home team
- `points.Visitor` Number of points scored by the visiting team
- `week` Week Number
- `half` The half of the season in which the match was played

Examples

```
data(tournament)
str(tournament)
```
Index

adjust, 6, 7, 9, 18
adjust (players), 20
as.data.frame.elo.run
   (elo.run.helpers), 15
as.matrix.elo.run, 21
as.matrix.elo.run (elo.run.helpers), 15
auc, 3
auc.elo, 2
auc.elo.run, 25

brier (elo.mse), 11
duplicated, 21
 elo, 3
elo.calc, 3, 4, 10, 12, 14, 16, 17
elo.colley, 5, 21
elo.glm, 6, 6, 9, 19–21
elo.markovchain, 8, 21
elo.model.frame, 6, 7, 9, 12, 14, 18
elo.mov, 10
elo.mse, 11
elo.prob, 3, 4, 10, 12, 14, 17
elo.run, 3, 10, 13, 15, 16, 19
elo.run.helpers, 14, 15
elo.run2 (elo.run), 13
elo.update, 3, 4, 10, 12, 14, 16
elo.update.default, 14
elo.winpct, 17

favored, 25
favored (favored.elo), 18
favored.elo, 18
final.elos, 22, 24
final.elos (elo.run.helpers), 15
fitted.elo, 19
formula.specials (players), 20
glm, 5–9, 17, 18
group (players), 20

is.score (score), 24

k (players), 20

mov, 6, 7, 9, 18
mov (elo.mov), 10
mse, 25
mse (elo.mse), 11

neutral, 6, 7, 9, 18
neutral (players), 20

players, 20
predict.elo, 21
predict.glm, 22

rank, 24
rank.teams, 23
regress (players), 20
residuals.elo.run (fitted.elo), 19

score, 3, 6, 7, 9, 10, 14, 18, 24, 24
summary.elo, 25
summary.elo.colley, 6
summary.elo.glm, 7
summary.elo.markovchain, 9
summary.elo.winpct, 18

the help page for formulas, 4, 5, 8, 10, 12, 13, 16, 17
tournament, 3, 26