Package 'elo'

January 21, 2019

Title Elo Ratings

Version 1.1.0

Date 2019-01-20

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.

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

BugReports https://github.com/eheinzen/elo/issues

RoxygenNote 6.1.1

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 2019-01-21 05:20:02 UTC

R topics documented:

`auc.elo.run` *Calculate AUC on an* `elo.run` *object*

Description

Calculate AUC on an `elo.run` object

Usage

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

Arguments

`object`	An object of class `elo.run`.
`...`	Other arguments (not used at this time).

Value

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

References

Adapted from code here: https://stat.ethz.ch/pipermail/r-help/2005-September/079872.html

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 Elos 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

Elo, A. E. 1978. The Rating of Chess Players, Past and Present. New York: Arco.

Examples

```
library(elo)
```

elo.calc *Elo functions*

Description

Calculate post-update Elo values. This is vectorized.

Usage

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

## Default S3 method:
elo.calc(wins.A, elo.A, elo.B, k, ..., adjust.A = 0,
  adjust.B = 0)

## 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).
`elo.A, elo.B`	Numeric vectors of elo scores.
`k`	A constant k-value (or a vector, where appropriate).
`adjust.A, adjust.B`	Numeric vectors to adjust `elo.A` and `elo.B` by.
`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)
```

---

elo.glm elo.glm

Description

Compute a logistic regression model for a matchup.

Usage

```
elo.glm(formula, data, na.action, subset, family = "binomial", ...,
  rm.ties = TRUE)
```

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.
family, ...	Arguments passed to glm.
rm.ties	Logical, denoting whether to remove ties on the left-hand side.

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. A glm model is then run to predict wins.

With this setup, usually one model term will be NA, as the input is linearly dependent. Consider this team's skill to be zero relative to the other teams. The intercept represents the home-field advantage.

Value

An object of class c("elo.glm", "glm").

See Also

score, elo.model.frame

Examples

```
data(tournament)
elo.glm(score(points.Home, points.Visitor) ~ team.Home + team.Visitor, data = tournament)
```

---

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")
```

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.

See Also

elo.run, elo.calc, elo.update, elo.prob

---

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.
adjust.A	Numeric vectors to adjust elo.A and elo.B by.
adjust.B	Numeric vectors to adjust elo.A and elo.B by.
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))
```

`elo.run` `elo.run`

Description

Calculate Elos for a series of matches.

Usage

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

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

## S3 method for class 'elo.run.regressed'
print(x, ...)
```

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

Value

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

See Also

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

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)
```

elo.run.helpers *Helper functions for* elo.run

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)
```

---

elo.update *Elo functions*

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:
elo.update(wins.A, elo.A, elo.B, k, ..., adjust.A = 0,
  adjust.B = 0)

## S3 method for class 'formula'
elo.update(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).
elo.A	Numeric vectors of elo scores.
elo.B	Numeric vectors of elo scores.
k	A constant k-value (or a vector, where appropriate).
adjust.A	Numeric vectors to adjust elo.A and elo.B by.
adjust.B	Numeric vectors to adjust elo.A and elo.B by.
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 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)
```

---

players *Details on* elo *formulas and the specials therein*

Description

Details on elo functions and the special functions allowed in them to change functions' behaviors.

Usage

```
players(..., weights = NULL)

k(x)

adjust(x, adjustment)

regress(x, to, by, regress.unused = TRUE)

group(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 ....
x	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.

`formula` accepts five special functions in it:

`k()` allows for complicated Elo updates. For constant Elo updates, use the `k =` argument instead of this special function.

`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.

`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.

`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.

---

predict.elo.run *Make Predictions on an* `elo.run` *Object*

Description

Make Predictions on an `elo.run` Object

Usage

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

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

Arguments

object	An object of class "elo.run".
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 to be passed to elo.prob.
regressed	See the note on final.elos.

Value

A vector of win probabilities.

Examples

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

---

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

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

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).

Examples

```
score(12, 10)
score(10, 10)
score(10, 12)
```

---

summary.elo.run *Summarize an* elo.run *Object*

Description

Summarize an elo.run Object

Usage

```
favored(x, ...)

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

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

mse(object, subset)

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

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

Arguments

x	An object of class "summary.elo.run".
...	Other arguments
object	An object resulting from elo.run.
subset	A vector of indices on which to calculate the MSE.

Value

A summary of object.

Examples

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

tournament *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