Package 'fbRanks'

July 2, 2014

Type Package

Title Association Football (Soccer) Ranking via Poisson Regression

Version 2.0

Date 2013-10-21

Depends stats

Imports igraph, stringr

Suggests RJSONIO, RCurl, httr, XML, Rlab, xtable, tcltk, speedglm,glmnet

Author Eli Holmes

Maintainer E Holmes <eeholmes@u.washington.edu>

Description This package uses time dependent Poisson regression and a record of goals scored in matches to rank teams via estimated attack and defense strengths. The statistical model is based on Dixon and Coles (1997) Modeling Association Football Scores and Inefficiencies in the Football Betting Market, Applied Statistics, Volume 46, Issue 2, 265-280. The package has a some webscrapers to assist in the development and updating of a match database. If the match database contains unconnected clusters (i.e. sets of teams that have only played each other and not played teams from other sets), each cluster is ranked separately relative to the median team strength in the cluster. The package contains functions for predicting and simulating tournaments and leagues from estimated models. The package allows fitting via the glm(), speedglm(), and glmnet() functions. The latter allows fast and efficient fitting of very large numbers of teams. The fitting algorithm will analyze the match data and determine which teams form a cluster (a set of teams where there is a path of matches connecting every team) and fit each cluster separately.

License GPL-2

URL http://www.lastplanetranking.blogspot.com/

LazyData yes

BuildVignettes yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-10-23 07:59:42

R topics documented:

`B00data` *Match and Team Data for B00 Youth Soccer Matches*

Description

Example data set for soccer rating and prediction. These are a set of ca 2000 select soccer matches for boys age Aug 2001 to Jul 2000 in Washington, Oregon, and British Columbia during the 2012-2013 season.

Usage

```
data(B00data)
```

Format

The data are supplied as a scores data frame and a teams data frame. The score data frame is the raw match data taken from multiple leagues and tournaments and teams appear under slightly different names in different tournaments. The teams data frame has the display name for a team and then a series of columns called alt.name.1, alt.name.2, etc with the alternate names used by the team. Teams in this area compete in leagues governed by different associations and only play each other during open tournaments. Teams within an association will meet teams from different leagues within the association cups.

Source

The match results were recorded from the league and tournament websites during the 2012-2013 season.

Examples

```
str(B00.scores)
```

`coef.fbRanks` *Return Coefficients from fbRanks Objects*

Description

The `rank.teams` function outputs fbRanks objects. `coef(fbRanks)`, where fbRanks is one's output from a `rank.teams` call, will return a list with vectors of coefficients for each fit and lists of coefficients for each term in a model. The fit element of a fbRanks object is a list with a fit (class "glm", "speedglm" or "glmnet") for each cluster. A cluster is a collection of teams with a path, represented by a series of games, between every team in the cluster. "attack" and "defend" are always model terms, but the user may have added on additional predictor variables (such as "surface").

Usage

```
## S3 method for class 'fbRanks'
coef(object, ...)
```

Arguments

`object`	A fbRanks object.
`...`	Not used.

Value

A list with elements `coef.vector` and `coef.list`. `coef.vector` is like the familar vector returned by a coef() call except that these vectors are held in a list where each element of the list is for the fit for a different cluster. `coef.list` is the vector separated into the coefficients for each model term. At the minimum, `coef.list` is a list with elements "attack" and "defend" because these terms always appear in the fbRanks model. Additional terms added by the user may also appear.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

Examples

```
## Not run:
#This will load two dataframes: B00.scores and B00.teams
data(B00data)

#fit a model using a particular date range for B00 teams
#set the data range to just be the summer games of WA teams
x=rank.teams(scores=B00.scores, teams=B00.teams,
             min.date="2012-5-1", max.date="2012-9-8", silent=TRUE)

#there are multiple clusters
names(coef(x)$coef.list)

#show the coefficients for cluster 1 as a list
#notice that fewer goals are scored on Turf
coef(x)$coef.list[[1]]

## End(Not run)
```

---

construct.team.file *Helper Function to Construct a Team File*

Description

Helper function.

Usage

```
construct.team.file(scores.file="scores.csv",
                    add.to.name="",
                    file="team_template.csv",
                    ignore="")
```

Arguments

scores.file	A single file name or vector of file names which have the match data. The function will throw an error if the column headers are not the same.
file	Name to give your team file.
ignore	A list of strings to ignore when matching teams. See example.
add.to.name	Any text to append to the display names.

Details

This will determine all the unique team names that differ only by 3 letters in the match files. It will choose 1 as the display name and put the rest in the alt.name columns. The result is a a template of the teams.csv file. This will then need to be edited heavily.

Value

Nothing is output. Only a .csv file is written.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

Examples

```
## Not run:
# A list of league match files to read in
file_list = c(
  "RCL D1.csv", "RCL D2.csv", "RCL D3.csv", "RCL D4.csv",
  "tournaments.csv")
ignore=c()
#use the current directory
construct.team.file(scores.file=file_list, file="team_template.csv", ignore=ignore)

## End(Not run)
```

---

create.fbRanks.dataframes

Helper Function to Construct the Scores and Teams Data Frames from csv files.

Description

Helper function. Reads in .csv files to create the scores, team.resolver, and teams data.frames.

Usage

```
create.fbRanks.dataframes(scores.file,
                          team.resolver=NULL,
                          teams.file=NULL,
                          date.format="%Y-%m-%d", na.remove=FALSE)
```

Arguments

`scores.file`	A single file name or vector of file names (csv files) which have the match data. Must have columns named "date","home.team","home.score", and "away.team","away.score". Extra columns can be added, e.g. "surface" or "attack.adv" (home, away, neutral), and these can then be used as response variables.
`team.resolver`	Optional. A team name resolver. It gives a unique team name (display name) associated with the team names appearing in the scores file home.team and away.team columns. This needs to have 2 columns. One column is "name" and each name in this column must appear as in the "name" column of the

teams.file. The second column is "alt.name" and these are the names that appear in the scores file. If a team uses different names in different leagues and tournaments, then each alternate name is one row in the team.resolver file. If not passed in, it will be constructed from the scores file home.team and away.team columns.

`teams.file` Optional. A single file name or vector of file names (csv files) which have the team names and team data. One column must be called "name" and is the display name for the team. The names in the "name" column must match those appearing in the "name" column of the team.resolver file, however there can be extra teams in the team file that do not appear in the team.resolver file. Features of teams, like age, state, club, etc, can also be added to the file and then can be used for filtering for printing or using as response variables. If left off, a teams data.frame will be constructed from the home.team and away.team names in the scores file.

`date.format` What format the dates in the scores file are in. Some common formats are "%m/%d%Y" for 12/31/2012, "%m/%d/%y" for 12/31/12, "%d.%m.%Y" for 31.12.2012. The default is 2012-12-31.

`na.remove` Remove matches where both home.score and away.score are missing (NaN). Do not set to TRUE if you are going to use the data frame for prediction of future matches. However, setting to TRUE can significantly reduce the size of the data object if you are only ranking teams.

Details

This creates scores and teams dataframes from the scores files and team files If use.display.names=TRUE, then the names in scores data frame will be the display name in the teams file

Value

This returns a list with elements `scores` and `teams` which are dataframes ready for the `rank.teams` function. `scores` has the team names replaced with the display name, while `raw.scores` has the original names (in scores file).

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

Examples

```
## Not run:
#This shows how a series of .csv files can be imported to create
#a scores and teams data frames

#this is the list of files with the league data
match.files = c(
  "OPL NPL.csv", "OPL 1st Div.csv",
  "OPL 2nd Div.csv", "OPL 3rd Div.csv","OPL 4th Div.csv",
)
```

```
#In this case the team file is a single file, but it could be a list
team.file="teams_b00.csv"
#This does error-checking and outputs dataframes in the proper format
b00_data=create.fbRanks.dataframes(scores.file=match.files, teams.file=team.file, date.format="


## End(Not run)
```

`create.newdata.dataframe` *Helper Function to Replace Team Names in Score File with a Uniform Name from Team Data Frame.*

Description

Helper function not exported for users. Used by predict.fbRanks() to construct the newdata data frame.

Usage

```
create.newdata.dataframe(x, newdata, min.date, max.date, ...)
```

Arguments

`x`	A fbRanks object.
`newdata`	A data.frame of data to use for predicting. At the minimum home.team, away.team and any predictors in x (the fbRanks model) are required.
`max.date, min.date`	Used for filtering the matches in newdata.
`...`	Other arguments for to use for filtering the scores data.frame or newdata data.frame. You can use any column in the scores or teams data frames. Name of the argument should be the column name and values passed in as a vector.

Details

This is taking a list or data frame passed in in the `newdata` argument of a `predict.fbRanks` call and constructing a data frame that can be passed to glm or lmer. It ensures that the uniform names (name column of `x$teams`) are used. It uses `resolve.team.names`.

Value

A list with elements `scores` and `teams`.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

fbRanks.utility.functions

Helper Functions in the fbRanks Package

Description

Helper functions not exported for users.

Usage

```
str_strip.white(string, sub = " ")
str_proper(string)
str_remove.nonascii(string, sub = "")
str_remove(string, start = 1L, end = -1L, sub = "")
detect.normality.outliers(x,alpha=0.05)
scale.for.ranks(type = 1, base = 2, silent = TRUE, str.disp.mult=1)
```

Arguments

string	A character vector.
start, end	Location of where to start and end for removing characters.
x	A vector of numbers
alpha	A significance level.
type	Type of scale.
sub	What to replace the removed characters with.
base	The base to which to scale the total strength. By default it is base(2). You want this to match the base used in your print call.
silent	Whether to output the scale to the console (silent=FALSE).
str.disp.mult	Multiply the strength numbers in the column names by this number.

Details

`str_strip.white` strips ending and internal extra white space. `str_proper` does proper capitalization. `str_remove.nonascii` remove non-ascii characters from strings. `str_remove` remove characters from strings. Like sub but removes the specified characters between start and end. `detect.normality.outliers` detects the values that violate normality to a particular alpha level. `scale.for.ranks` prints a scale for fbRanks ratings.

Value

`str_strip.white` returns a vector with the extra white space striped.

`detect.normality.outliers` returns a vector of TRUE, FALSE values.

`scale.for.ranks` will return the simulations used to construct the scale if the output is assigned to a variable. `team.name.select` calls up a GUI that allows the user to select the team name from a list or input a new team name.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

`plot.fbRanks` *Plotting function for fbRanks Objects*

Description

The `rank.teams` function outputs fbRanks objects. `plot(fbRanks)`., where fbRanks is one's output from a `rank.teams` call, will plot the rankings for each cluster of teams in the match data. Specific teams, regions, or leagues can be labeled on the plots.

Usage

```
## S3 method for class 'fbRanks'
plot(
              x, ...,
              which = "residuals",
              annotate = list(title = TRUE),
              team.resids = NULL,
              min.date = NULL,
              max.date = NULL)
```

Arguments

`x`	A fbRanks object.
`...`	Extra elements to filter the ranks with, e.g. country. Must be column names in scores or teams dataframes.
`which`	What to plot. Options are "residuals" and "hist". If "hist" then annotate needs Name element.
`annotate`	Info to add to title. If title=TRUE, then the team name is added to the plot.
`team.resids`	If which="residuals", the team residuals are needed. This is provided by residuals(x). This is time-consuming, so depending how plot is called, you might want to pass in the result of residuals(x) in the plot call.
`min.date`	Minimum date on the x-axis of a residuals plot.
`max.date`	Maximum date on the x-axis of a residuals plot.

Value

Some plots of team residuals and ranks relative to other teams.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

`predict.fbRanks` *Predict function for fbRanks Objects*

Description

The `rank.teams` function outputs fbRanks objects. `predict(fbRanks)`, where fbRanks is one's output from a `rank.teams` call, will predict the result (win, loss, tie) and goals for a set of matches between teams.

Usage

```
## S3 method for class 'fbRanks'
predict(object, ...,
       newdata=list(home.team="foo", away.team="bar"),
       max.date="2100-6-1", min.date="1900-5-1",
       rnd=TRUE, silent=FALSE, show.matches=TRUE,
       verbose=FALSE, remove.outliers=TRUE, n=100000)
```

Arguments

`object`	A fbRanks object.
`...`	Other arguments for to use for filtering the scores data.frame or newdata data.frame. You can use any column in the scores or teams data frames. Name of the argument should be the column name and values passed in as a vector. See examples.
`newdata`	A data.frame of data to use for predicting. At the minimum home.team, away.team and any predictors in x (the fbRanks model) are required.
`max.date, min.date`	Used for filtering the scores data.frame or newdata data.frame.
`rnd`	whether to round the output of predictions.
`silent`	whether to print anything
`show.matches`	whether to print the match predicitons
`verbose`	whether to print some prediction performances stats.
`remove.outliers`	whether to show predictions for teams whose estimates violate normality.
`n`	the number of simulate matches to use when computing the probabilities of win, tie, loss

Value

A prediction for a set of matches based on an estimated model of attack and defense strengths for each team.

Author(s)

Eli Holmes, Seattle, USA. eeholmes(at)u(dot)washington(dot)com

See Also

rank.teams, simulate.fbRanks

Examples

```
## Not run:
#This will load two dataframes: B00.scores and B00.teams
data(B00data)

#fit a model using a particular date range for B00 teams
#set the data range to just be the summer games of WA teams
x=rank.teams(scores=B00.scores, teams=B00.teams,
             min.date="2012-5-1", max.date="2012-9-8", silent=TRUE)

#x is a fbRanks object with the fitted model and ranks, and the scores and teams data.frames
#the scores data.frames has all the data, not just the summer data

#Now we can use the summer data to predict the fall RCL D1 games (top B00 league in WA)
predict(x, venue="RCL D1")

#We can also predict all the games for a particular team
predict(x, name="Seattle United Copa B00")

## End(Not run)
```

---

print.fbRanks *Printing function for fbRanks Objects*

Description

The rank.teams function outputs fbRanks objects. print(fbRanks)., where fbRanks is one's output from a rank.teams call, will print out the rankings for each cluster of teams in the match data.

Usage

```
## S3 method for class 'fbRanks'
print(x, ..., scaling="median total", base=2,
       log.transform.attack=FALSE, header=TRUE, silent=FALSE, type="", file="")
```

Arguments

x	A fbRanks object.
...	Other filters to apply when printing. These must match column names in either x$teams or x$scores. For example, if x$teams has a column named 'country' with values 'UK','Canada' and 'Germany', you can pass in country="UK" to only show UK ranks.
scaling	How to scale the ranks. By default, they are scaled with 0 at the median team in the database. Pass in a number to scale to that number.
base	By default, the ranks are shown base(2) but you can other bases. Pass in exp(1) to change to base e.
log.transform.attack	Whether to show the attack and defense scores as logged transformed.
header	Whether to add some header information to the print out.
type	If type="html", html of the ranks table is output.
file	If type="html", then the output can be output to a file.
silent	If FALSE no output but the ranks are returned.

Value

A print out of team ranks.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

Examples

```
#This loads two data frames: B00.scores and B00.teams
data(B00data)
#rank the teams in the RCL D1 league
x=rank.teams(scores=B00.scores, teams=B00.teams, venue="RCL D1")

#Just a printing of the ranks with no extra info shown.
#some 2 team clusters are shown.  Those are teams with just 1 game (against each other)
print(x)

## Not run:
#rank all teams
x=rank.teams(scores=B00.scores, teams=B00.teams, venue="RCL D1")

#You can also do filtered prints
#print the ranks with the regions shown
print(x, region="all")

#print ranks for just a region
print(x, region="WA")
```

```
#print ranks for a league; fall.league is a column in the teams data frame
print(x, fall.league=c("RCL D1 U12","RCL D2 U12"))

#print ranks for teams in a tournament; venue is a column in the scores data frame
print(x, venue="Baker Blast")

## End(Not run)
```

---

rank.teams *Team Ranks Via Poission Regression*

Description

Creates ranks using a dataframe of match records. This, along with print.fbRanks and predict.fbRanks, are the main functions in the package. Use vignette("Basic_team_ranking",package="fbRanks") at the command line to open a vignette that walks through a basic ranking using a data.frame or comma-delimited text file.

Usage

```
rank.teams(scores=NULL, teams=NULL,
       family="poisson", fun="glm",
       max.date="2100-6-1", min.date="1900-5-1", date.format="%Y-%m-%d",
       time.weight.eta=0, add=NULL, silent=FALSE, ...)
```

Arguments

scores	A data frame of match results. Must have columns "date", "home.team", "home.score", "away.team", "away.score". Missing scores must be denoted NaN. Extra columns to be used for filtering in print results or as explanatory variables can be included, e.g. surface or attack.adv.
teams	A data frame with the team data. Must have columns "name" and "alt.name.x", where x can be anything, e.g. 1. Extra columns to be used for filtering in print results or as explanatory variables can be included, e.g. age. None of the column names in the teams data frame are allowed to be the same as names in the scores data frame.
family	Passed to glm or glmer. If you are using speedglm, use the glm notation for family, e.g. "poisson", rather than the equivalent speedglm notation, e.g. poisson(log).
fun	"glm", "glmnet", or "speedglm"
max.date	Latest match date to use when fitting the model.
min.date	Earliest match date to use when fitting the model.
date.format	The date formate for max.date, min.date and dates in the scores dataframe.
add	Vector of explanatory variables to add to the model. Must be a character vector that corresponds to names of columns in scores or teams dataframes.

`time.weight.eta`
: How much time weighting to include. 0 is no weighting. 0.1 would weight the most recent games quite strongly.

`silent`
: Suppresses printing.

`...`
: Other filters to apply when ranking. These must match column names in either `teams` or `scores`. For example, if `teams` has a column named 'country' with values 'UK','Canada' and 'Germany', you can pass in `country="UK"` to only rank using the matches for UK teams.

Details

The function uses Dixon and Coles time-weighted poisson model to estimate attack and defense strengths using the `glm` function. Extra explanatory variables (factors or continuous) can be added to the model. The output is a print out of the attack+defense strengths (in total column) and the exp(attack) and exp(defense) strengths in separate columns. Take the ratio of team A's attack strength to the team B's defense strength to get the expected goals scored by team A in a match between A and B.

The 2 raised to the difference in total strengths of two teams is the relative scoring rate of team A to team B. Thus if the difference in total strength is 1 (team A - team B = 1) then team A scores 2^1 times faster than team B (in a match up between the two) and expected to score 2 goals for every 1 of team B. If the diffence is 3, then team A is expected to score 2^3 = 8 goals for every 1 of team B.

The model can be fit with three different glm functions: glm(), speedglm(), and glmnet() using the `fun` argument. speedglm() requires the speedglm package. It is considerably faster than glm() and returns identical coefficient values. glmnet() requires the glmnet package. It uses a different algorithm and does not return exactly the same values as glm(). lambda=0 is passed to the glmnet() (in the rank.teams code) to force it to return the OLS values, yet for the poisson family there are some small differences. However, it is exceedingly fast and uses little RAM. It is required for ranking large datasets with 1000s of teams because glm() and speedglm() have extreme RAM requirements for models with 1000s of teams and take excessive amounts of time to fit. In addition, testing suggests that glmnet() coefficients are more robust (closer to true values) for these large models.

Value

A list of class `fbRanks` with the following components:

`fit`
: A list with the glm fit for each cluster.

`graph`
: A list with some information about the graph describing the interconnectedness of the teams from the `igraph` package. The elements are graph (output from `graph.edgelist`), membership (output from `cluster`), csize (output from `cluster`), no (output from `cluster`), names (output from `get.vertex.attribute`).

`scores`
: The scores dataframe with all team names replaced with display names.

`teams`
: The teams data.frame.

`max.date`
: The most recent match used in model fit.

`min.date`
: The oldest match used in model fit.

`time.weight.eta`
: The time weighting used.

`date.format`
: The date format to use when displaying output using the fbRanks object.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

References

Dixon and Coles (1997) Modeling Association Football Scores and Inefficiencies in the Football Betting Market, Applied Statistics, Volume 46, Issue 2, 265-280

See Also

print.fbRanks, create.fbRanks.dataframes, predict.fbRanks, coef.fbRanks

Examples

```
#load the example data set
data(B00data)

#rank teams in the RCL D1 league using just the league data
x=rank.teams(scores=B00.scores, teams=B00.teams, venue="RCL D1")

#repeat with surface (turf, grass) as an explanatory variable
ranks2=rank.teams(scores=B00.scores, teams=B00.teams, venue="RCL D1", add=c("surface","adv"))

#Slightly fewer goals per game are scored on turf
coef(ranks2)$coef.list$cluster.1$surface.f

#Slightly more goals per game are scored at home
coef(ranks2)$coef.list$cluster.1$adv.f

#get the ranks based on summer data
# x=rank.teams(scores=B00.scores, teams=B00.teams,
#              min.date="2012-5-1", max.date="2012-9-8", silent=TRUE)

# See the vignette Basic Team Ranking for more examples
```

---

residuals.fbRanks *Residuals function for fbRanks Objects*

Description

The rank.teams function outputs fbRanks objects. residuals(fbRanks)., where fbRanks is one's output from a rank.teams call, returns the model response residuals (data minus fitted values) for each match in the fbRanks$scores data frame. The output is organized as a list by team. If you just want the response residuals for each match, that is in fbRanks$scores$home.residuals and $away.residuals.

Usage

```
## S3 method for class 'fbRanks'
residuals(object, ...)
```

Arguments

object	A fbRanks object.
...	Other filters to apply when returning residuals. These must match column names in either x$teams or x$scores. For example, if x$teams has a column named 'country' with values 'UK','Canada' and 'Germany', you can pass in country="UK" to only show UK residuals.

Value

A list of the response residuals (data minus fitted values) for each team. It's rather verbose. You probably want to assign it to a variable and work with that. Look at predict.fbRanks. That provides similar output in the $scores variable.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

See Also

predict.fbRanks

Examples

```
data(B00data) #load a set of games
#fit a model using just the matches from the RCL D1 league
x=rank.teams(scores=B00.scores, teams=B00.teams, venue="RCL D1")

#Show the predicted versus actual scores
b00.resids=residuals(x)

## Not run:
#you can compare to the residuals for a model to fit to all the data (including RCL D1)
x=rank.teams(scores=B00.scores, teams=B00.teams)

#Just a printing of the ranks with no extra info shown.
b00.resids=residuals(x, venue="RCL D1")

## End(Not run)
```

`resolve.team.names` *Helper Function to Create Uniform Team Names in Score Data Frame*

Description

Helper function not exported for users. Used primarily in rank.teams() function to construct the team predictor variable.

Usage

```
resolve.team.names(scores, team.resolver, team.data=NULL, use.team.select=TRUE)
```

Arguments

`scores`	A scores data frame.
`team.resolver`	A team name resolver. It gives a unique team name (display name) associated with the team name used in leagues and tournaments.
`team.data`	The team info data frame. Only needed if team select GUI is used.
`use.team.select`	Whether to call up the GUI `team.name.select` to select names missing from the team name resolver.

Details

This replaces the team names in scores data frame with a unique name from the team name resolver (in name column). It is used primarily in the `create.fbRanks.dataframes` function. The `scores` element of a fbRanks object is the score data frame with the uniform names. The `raw.scores` element is the scores with the original names (as used in the match).

Value

Scores data frame with unique and uniform names for the teams in `home.team` and `away.team` columns.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

scrape.matches *Webscraping Match Scores*

Description

Webscrapers for various types of match score content managers.

Usage

```
scrape.korrio(url, file="Korrio", url.date.format="%B %Y %a %d",
     date.format="%Y-%m-%d", append=FALSE, get.surface=FALSE, ...)
scrape.demosphere(url, file="Demosphere", url.date.format="%B %d %Y",
    date.format="%Y-%m-%d", table.style=1, year=NULL, append=FALSE,
    get.surface=FALSE, ...)
scrape.demosphere.main(url, div.resolver, name="Demosphere", basedir=".",
    url.date.format="%B %d %Y", date.format="%Y-%m-%d", U12=2001,
    table.style=1, append=FALSE, get.surface=FALSE, ...)
scrape.gotsport(url, file="GotSport", tb.num=10, url.date.format="%m/%d/%Y",
    table.style=1, date.format="%Y-%m-%d", append=FALSE, ...)
scrape.gotsport.main(url, name="test", basedir=".", tb.num=10,
    url.date.format="%m/%d/%Y", table.style=1, date.format="%Y-%m-%d",
    U12=2001, append=FALSE, ...)
scrape.sportaffinity(url,file="SportAffinity", url.date.format="%B %d, %Y",
    date.format="%Y-%m-%d", append=FALSE, ...)
scrape.sportaffinity.brackets(url, file, venue=NULL, ...)
scrape.sportaffinity.main(url, name="SportAffinity", basedir=".",
    url.date.format="%B %d, %Y", date.format="%Y-%m-%d", U12=2001,
    append=FALSE, add.to.base.url="tour/public/info/", ...,
    U.designation="Under ", name.delimiter="Under ", name.skip=3)
scrape.scoreboard(html.file, file="ScoreBoard", url.date.format="%a %m/%d/%Y",
     date.format="%Y-%m-%d", append=FALSE, get.surface=FALSE, ...)
scrape.custom1(url, file="Custom1", weeks=NULL, first.td.tag=3, last.td.tag=7,
    td.per.row=5, append=FALSE, ...)
scrape.custom2(url, file="Custom2", year=NULL, date.format="%Y-%m-%d", append=FALSE, ...)
scrape.custom3(url, file="Custom3", year=NULL, date.format="%Y-%m-%d", append=FALSE, ...)
scrape.custom4(url, file="Custom4", year=NULL, date.format="%Y-%m-%d", append=FALSE, ...)
scrape.json1(url, file="Json1", date.format="%Y-%m-%d", append=FALSE, ...)
scrape.usclub(url, file="USClub", url.date.format="%A%m/%d/%Y",
    date.format="%Y-%m-%d", append=FALSE, ...)
```

Arguments

url	URL to the webpage with the match information.
html.file	the html file(s) to scrape.
file	file where the match data is to be saved. Will be saved as a comma-delimined flat file. This should include the directory if needed (i.e. not to be saved in the working directory.

`name` name of the file to be saved. Needed when the name of the file need to be dynamically created for scrapers that scrape all age groups.

`basedir` base directory where gender-age files are to be saved. Needed when the name of the file need to be dynamically created for scrapers that scrape all age groups.

`U12` The year that a U12 player is associated with. Needed when multiple ages are being scraped and the gender-age must be computed.

`append` whether to append the match data to the existing file.

`date.format` the date format to be used in the date column in the outputted match file.

`get.surface` Some websites have surface (turf, grass) information and this can be scraped if desired.

`tb.num` the table number to scrape. Some websites put have the table in different places.

`url.date.format`
the date format on the webpage.

`table.style` the table style. Some content managers use different formats on different pages.

`first.td.tag, last.td.tag, td.per.row`
custom information for dealing with badly formed table html.

`weeks` dates for webpages that show week number instead of a date. Must be in YYYY-mm-dd format.

`year` The year to associate with the match dates since the match dates are not shown with the year on the webpage.

`venue` For scraping sport affinity brackets, the user should use the column name venue if the bracket name should be added to the score data. Otherwise no bracket info is added, only the scores are scraped.

`div.resolver` For scrape.demosphere.main. tells what division names are on the Demosphere page for scraping the main page

`U.designation` For scrape.sportaffinity.main. U.designation is the text right before the age (a 2 digit number); so if the ages are denoted like "Boys Under 11", U.designation = "Under " or "der ". If ages were denoted "BU11", U.designation="U".

`name.delimiter` For scrape.sportaffinity.main. name.delimiter and name.skip help form the division name. Say divisions are like "Boys Under 11 Foobar" and you want to use Foobar as the division. Then name.delimiter="Under " and name.skip=3. This says the delimiter is "Under " and after that skip another 3 spaces to find the division name. If you want 11 Foobar as your division, use name.skip=0. If you want Under 11 Foobar, name.delimiter="Boys" name.skip=1

`name.skip` For scrape.sportaffinity.main. See above.

`add.to.base.url`
For scrape.sportaffinity.main. The scraper constructs the urls for the individual ages. It needs to know if it shoud add anything to the base url of the main page to the info it gets from the age links.

`...` Other columns to append to the match file. For example, a column denoting the league or venue name.

Details

These webscrapers are customized for various match delivery platforms: Korrio, Demosphere, GotSport, SportAffinity and ScoreBoard. Look at the bottom of the match website to determine the platform and thus scraper to use. These scrapers will go out of date quickly as website structure is changed. Thus you will probably need to modify the scrapers for your own purposes. The custom1 scraper shows how to scrape a page with improperly formed html. Custom2, Custom3 and Custom4 are for scores not provided by a standard content provider. The json1 scraper shows how to scrape JSON data.

Type the scraper name (e.g. scrape.custom1) at the command line to see comments and a url example for each scraper.

Value

The scrapers output a comma-delimited file with the match data. For examples, see the vignettes.

Author(s)

Eli Holmes, Seattle, USA. ee(dot)holmes(at)u(dot)washington(dot)edu

---

simulate.fbRanks *Simulate function for fbRanks Objects*

Description

The `rank.teams` function outputs fbRanks objects. `simulate(fbRanks)`, where fbRanks is one's output from a `rank.teams` call, will simulate games and produce predicted standings based on different points rules.

Usage

```
## S3 method for class 'fbRanks'
simulate(
         object,
         nsim=100000,
         seed=NULL, ...,
         newdata=list(home.team="foo", away.team="bar"),
         bracket.names=NULL,
         max.date="2100-6-1", min.date="1900-5-1", silent=FALSE,
         points.rule="tournament 10pt",tie.rule=list(tie.rule.gd.max=10),
         non.equal.games.rule=list(n.games=3,rule="proportional"),
         show.matches=FALSE, groups.column=NULL)
```

Arguments

object	A fbRanks object.
...	Other arguments for to use for filtering the scores data.frame or newdata data.frame. See examples.
newdata	An (optional) data.frame of data to use for predicting. At the minimum home.team, away.team and any predictors in x (the fbRanks model) are required. You can also pass in arguments to specify that newdata be constructed from the scores data.frame in the fbRanks object. See examples.
bracket.names	An optional list with the team in each bracket. If not used, all teams are assumed to be in the same bracket unless groups.column is passed in. In that case, groups.column is used to set the brackets. See examples.
max.date, min.date	Used for filtering the scores data.frame or newdata data.frame.
silent	whether to print anything
nsim	the number of simulations.
points.rule	A text rule or a list with pwin, pdraw, pshutout, pgoal, pgoal.max. See details.
tie.rule	Currently only a GD rule with max GD per game is specified.
non.equal.games.rule	How to deal with teams with unequal number of games.
show.matches	whether to print the match predicitons
groups.column	if your scores data.frame or newdata data.frame has a column that indicates the bracket information. See examples.
seed	not used

Details

The following points rules are available. "tournament10pt": 6 points for a win, 3 points for a draw, 1 point for each goal scored (up to a maximum of 3 per game for both teams), and 1 point for a shutout. In the event of a 0-0 tie, both teams are awarded 4 points. "league3pt": 3 points for a win, 1 points for a draw, 0 point for each goal scored, and 0 point for a shutout. A custom points rule can be specified by passing in a list with the elements pwin, pdraw, pshutout, pgoal, pgoal.max to specify points for win, draw, shutout, and per goal up to a max of pgoal.max goals.

If teams have non-equal number of games, if non.equal.games.rule="proportional", the default, then the total points are divided by the number of games played.

If specifying what to simulate using newdata, ensure that the data format is the same as that in object$date.format. Also if there are any NA attack or defense strengths for your teams, predict will throw an error and exit.

Value

A simulation based on an estimated model of attack and defense strengths for each team.

Author(s)

Eli Holmes, Seattle, USA. eeholmes(at)u(dot)washington(dot)com

Examples

```
#When specifying filters such as age, name, or league, these must correspond to column names in the
#scores or team files.
data(B00data) #load a set of games
#fit a model using a particular date range for B00 teams
#add predictors surface and adv (home/away advantage)
## Not run:
x=rank.teams(scores=B00.scores, teams=B00.teams, min.date="2012-5-1", max.date="2012-9-8",
    silent=TRUE, add=c("surface","adv"))

#simulate a league
#in this case, since newdata is not passed in, simulate tries to construct
#newdata from x$scores using the venue information
simulate(x, venue="RCL D1", points.rule="league3pt")

## End(Not run)

#simulate a tournament
#B00data includes a dataframe B00.founders with the brackets for
#the preliminary rounds of the Founders B00 2012 tournament
#fit a model with all data up to the start of the Founders Cup
## Not run:
x=rank.teams(scores=B00.scores, teams=B00.teams, min.date="2012-5-1", max.date="2012-12-14",
    silent=TRUE, add=c("surface","adv"))
simulate(x, newdata=B00.founders, groups.column="venue")

## End(Not run)
```

---

team.and.score.filters

Helper Function to Determine Teams or Scores to Include

Description

Helper function not exported for users. This creates a vector of teams names that match the filter information passed in in ... and a vector of TRUE, FALSE for each row of x$scores to include. This function is used throughout the package whenever filtering on scores or teams columns is requested.

Usage

```
team.and.score.filters(x, ...)
```

Arguments

x	A list with elements scores which is a scores data frame with uniform team names and teams which is a teams data frame. Typically is a fbRanks object in most function calls.
...	Other filters to apply when printing. These must match column names in either x$teams or x$scores. For example, if x$teams has a column named 'country' with values 'UK','Canada' and 'Germany', you can pass in country="UK" to only show UK ranks.

Value

A list with include.teams, a vector of team names, and include.scores, a vector of TRUE, FALSE for each row in x$scores.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

---

team.name.select *Helper Function to Create Uniform Team Names in Score Data Frame*

Description

Helper function. Calls up a GUI to help enter team info. Requires tcltk package. Only edit.team.data is exported.

Usage

```
team.name.select(newname, team.resolver, team.data, scores, type="alt.name")
edit_team_data(team.resolver, team.data, scores)
```

Arguments

newname	A list of unknown alt.names or display names.
team.resolver	A team name resolver. It gives a unique team name (display name) associated with the team name used in leagues and tournaments.
team.data	The team info data frame. Only needed if team select GUI is used.
scores	A scores data frame.
type	"alt.name"/"disp.name": Whether the newname is a list of bad alt.names (team name in match file that does not appear in the alt.name column of team resolver) or display names (name column of team resolver that does not match a name in the name column of the team file).

Details

This calls up a GUI to use to select teams from the team data file to use for unknown alt.names (team names in the match file) or team resolver display names (in the name column of team resolver). This will return updated data.frames for team.resolver and team.data. These should be saved if you do not want to go through the process of entering team data again.

Value

team.resolver and team.data data frames with with updated info. ok and updated also returned to indicate that there were no errors and whether team.resolver and team.data were updated.

Author(s)

Eli Holmes, Seattle, USA.

eeholmes(at)u(dot)washington(dot)com

Index