Package 'BradleyTerry2'

April 10, 2025

Title Bradley-Terry Models

Version 1.1.3

Description Specify and fit the Bradley-Terry model, including structured versions in which the parameters are related to explanatory variables through a linear predictor and versions with contest-specific effects, such as a home advantage.

License GPL (>= 2)

URL https://github.com/hturner/BradleyTerry2

BugReports https://github.com/hturner/BradleyTerry2/issues

Depends R (>= 2.10)

Imports brglm, gtools, lme4 (>= 1.0), qvcalc, stats

Suggests bookdown, knitr, litedown, prefmod, testthat

Enhances gnm

VignetteBuilder knitr, litedown

Encoding UTF-8

Language en-GB

LazyData yes

RoxygenNote 7.3.2

NeedsCompilation no

Author Heather Turner [aut, cre],
David Firth [aut]

Maintainer Heather Turner <ht@heatherturner.net>

Repository CRAN

Date/Publication 2025-04-10 19:20:44 UTC

Contents

`add1.BTm` *Add or Drop Single Terms to/from a Bradley Terry Model*

Description

Add or drop single terms within the limit specified by the `scope` argument. For models with no random effects, compute an analysis of deviance table, otherwise compute the Wald statistic of the parameters that have been added to or dropped from the model.

Usage

```
## S3 method for class 'BTm'
add1(object, scope, scale = 0, test = c("none", "Chisq", "F"), x = NULL, ...)
```

Arguments

`object`	a fitted object of class inheriting from `"BTm"`.
`scope`	a formula specifying the model including all terms to be considered for adding or dropping.
`scale`	an estimate of the dispersion. Not implemented for models with random effects.

test	should a p-value be returned? The F test is only appropriate for models with no random effects for which the dispersion has been estimated. The Chisq test is a likelihood ratio test for models with no random effects, otherwise a Wald test.
x	a model matrix containing columns for all terms in the scope. Useful if add1 is to be called repeatedly. **Warning:** no checks are done on its validity.
...	further arguments passed to add1.glm().

Details

The hierarchy is respected when considering terms to be added or dropped: all main effects contained in a second-order interaction must remain, and so on.

In a scope formula '.' means 'what is already there'.

For drop1, a missing scope is taken to mean that all terms in the model may be considered for dropping.

If scope includes player covariates and there are players with missing values over these covariates, then a separate ability will be estimated for these players in *all* fitted models. Similarly if there are missing values in any contest-level variables in scope, the corresponding contests will be omitted from all models.

If formula includes random effects, the same random effects structure will apply to all models.

Value

An object of class "anova" summarizing the differences in fit between the models.

Author(s)

Heather Turner

See Also

BTm(), anova.BTm()

Examples

```
result <- rep(1, nrow(flatlizards$contests))
BTmodel1 <- BTm(result, winner, loser,
                ~ throat.PC1[..] + throat.PC3[..] + (1|..),
                data = flatlizards,
                tol = 1e-4, sigma = 2, trace = TRUE)

drop1(BTmodel1)

add1(BTmodel1, ~ . + head.length[..] + SVL[..], test = "Chisq")

BTmodel2 <- update(BTmodel1, formula = ~ . + head.length[..])

drop1(BTmodel2, test = "Chisq")
```

anova.BTm *Compare Nested Bradley Terry Models*

Description

Compare nested models inheriting from class "BTm". For models with no random effects, compute analysis of deviance table, otherwise compute Wald tests of additional terms.

Usage

```
## S3 method for class 'BTm'
anova(object, ..., dispersion = NULL, test = NULL)
```

Arguments

object	a fitted object of class inheriting from "BTm".
...	additional "BTm" objects.
dispersion	a value for the dispersion. Not implemented for models with random effects.
test	optional character string (partially) matching one of "Chisq", "F" or "Cp" to specify that p-values should be returned. The Chisq test is a likelihood ratio test for models with no random effects, otherwise a Wald test. Options "F" and "Cp" are only applicable to models with no random effects, see stat.anova().

Details

For models with no random effects, an analysis of deviance table is computed using anova.glm(). Otherwise, Wald tests are computed as detailed here.

If a single object is specified, terms are added sequentially and a Wald statistic is computed for the extra parameters. If the full model includes player covariates and there are players with missing values over these covariates, then the NULL model will include a separate ability for these players. If there are missing values in any contest-level variables in the full model, the corresponding contests will be omitted throughout. The random effects structure of the full model is assumed for all sub-models.

For a list of objects, consecutive pairs of models are compared by computing a Wald statistic for the extra parameters in the larger of the two models.

The Wald statistic is always based on the variance-covariance matrix of the larger of the two models being compared.

Value

An object of class "anova" inheriting from class "data.frame".

Warning

The comparison between two or more models will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and 's default of `na.action = na.omit` is used. An error will be returned in this case.

The same problem will occur when separate abilities have been estimated for different subsets of players in the models being compared. However no warning is given in this case.

Author(s)

Heather Turner

See Also

`BTm()`, `add1.BTm()`

Examples

```
result <- rep(1, nrow(flatlizards$contests))
BTmodel <- BTm(result, winner, loser, ~ throat.PC1[..] + throat.PC3[..] +
               head.length[..] + (1|..), data = flatlizards,
               trace = TRUE)
anova(BTmodel)
```

---

baseball *Baseball Data from Agresti (2002)*

Description

Baseball results for games in the 1987 season between 7 teams in the Eastern Division of the American League.

Usage

```
baseball
```

Format

A data frame with 42 observations on the following 4 variables.

home.team a factor with levels `Baltimore`, `Boston`, `Cleveland`, `Detroit`, `Milwaukee`, `New York`, `Toronto`.

away.team a factor with levels `Baltimore`, `Boston`, `Cleveland`, `Detroit`, `Milwaukee`, `New York`, `Toronto`.

home.wins a numeric vector.

away.wins a numeric vector.

Note

This dataset is in a simpler format than the one described in Firth (2005).

Source

Page 438 of Agresti, A. (2002) *Categorical Data Analysis* (2nd Edn.). New York: Wiley.

References

Firth, D. (2005) Bradley-Terry models in R. *Journal of Statistical Software*, **12**(1), 1–12.

Turner, H. and Firth, D. (2012) Bradley-Terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, **48**(9), 1–21.

See Also

`BTm()`

Examples

```
##  This reproduces the analysis in Sec 10.6 of Agresti (2002).
data(baseball) # start with baseball data as provided by package

##  Simple Bradley-Terry model, ignoring home advantage:
baseballModel1 <- BTm(cbind(home.wins, away.wins), home.team, away.team,
                      data = baseball, id = "team")

##  Now incorporate the "home advantage" effect
baseball$home.team <- data.frame(team = baseball$home.team, at.home = 1)
baseball$away.team <- data.frame(team = baseball$away.team, at.home = 0)
baseballModel2 <- update(baseballModel1, formula = ~ team + at.home)

##  Compare the fit of these two models:
anova(baseballModel1, baseballModel2)
```

BTabilities *Estimated Abilities from a Bradley-Terry Model*

Description

Computes the (baseline) ability of each player from a model object of class `"BTm"`.

Usage

```
BTabilities(model)
```

Arguments

model	a model object for which inherits(model, "BTm") is TRUE

Details

The player abilities are either directly estimated by the model, in which case the appropriate parameter estimates are returned, otherwise the abilities are computed from the terms of the fitted model that involve player covariates only (those indexed by model$id in the model formula). Thus parameters in any other terms are assumed to be zero. If one player has been set as the reference, then predict.BTm() can be used to obtain ability estimates with non-player covariates set to other values, see examples for predict.BTm().

If the abilities are structured according to a linear predictor, and if there are player covariates with missing values, the abilities for the corresponding players are estimated as separate parameters. In this event the resultant matrix has an attribute, named "separate", which identifies those players whose ability was estimated separately. For an example, see flatlizards().

Value

A two-column numeric matrix of class c("BTabilities", "matrix"), with columns named "ability" and "se"; has one row for each player; has attributes named "vcov", "modelcall", "factorname" and (sometimes — see below) "separate". The first three attributes are not printed by the method print.BTabilities.

Author(s)

David Firth and Heather Turner

References

Firth, D. (2005) Bradley-Terry models in R. *Journal of Statistical Software*, **12**(1), 1–12.

Turner, H. and Firth, D. (2012) Bradley-Terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, **48**(9), 1–21.

See Also

BTm(), residuals.BTm()

Examples

```
### citations example

##  Convert frequencies to success/failure data
citations.sf <- countsToBinomial(citations)
names(citations.sf)[1:2] <- c("journal1", "journal2")

##  Fit the "standard" Bradley-Terry model
citeModel <- BTm(cbind(win1, win2), journal1, journal2, data = citations.sf)
BTabilities(citeModel)
```

```
### baseball example

data(baseball) # start with baseball data as provided by package

##  Fit mode with home advantage
baseball$home.team <- data.frame(team = baseball$home.team, at.home = 1)
baseball$away.team <- data.frame(team = baseball$away.team, at.home = 0)
baseballModel2 <- BTm(cbind(home.wins, away.wins), home.team, away.team,
                      formula = ~ team + at.home, id = "team",
                      data = baseball)
##  Estimate abilities for each team, relative to Baltimore, when
##  playing away from home:
BTabilities(baseballModel2)
```

BTm *Bradley-Terry Model and Extensions*

Description

Fits Bradley-Terry models for pair comparison data, including models with structured scores, order effect and missing covariate data. Fits by either maximum likelihood or maximum penalized likelihood (with Jeffreys-prior penalty) when abilities are modelled exactly, or by penalized quasi-likelihood when abilities are modelled by covariates.

Usage

```
BTm(
  outcome = 1,
  player1,
  player2,
  formula = NULL,
  id = "..",
  separate.ability = NULL,
  refcat = NULL,
  family = "binomial",
  data = NULL,
  weights = NULL,
  subset = NULL,
  na.action = NULL,
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = NULL,
  br = FALSE,
  model = TRUE,
  x = FALSE,
  contrasts = NULL,
```

```
  ...
)
```

Arguments

`outcome`	the binomial response: either a numeric vector, a factor in which the first level denotes failure and all others success, or a two-column matrix with the columns giving the numbers of successes and failures.
`player1`	either an ID factor specifying the first player in each contest, or a data.frame containing such a factor and possibly other contest-level variables that are specific to the first player. If given in a data.frame, the ID factor must have the name given in the `id` argument. If a factor is specified it will be used to create such a data.frame.
`player2`	an object corresponding to that given in `player1` for the second player in each contest, with identical structure – in particular factors must have identical levels.
`formula`	a formula with no left-hand-side, specifying the model for player ability. See details for more information.
`id`	the name of the ID factor.
`separate.ability`	(if `formula` does not include the ID factor as a separate term) a character vector giving the names of players whose abilities are to be modelled individually rather than using the specification given by `formula`.
`refcat`	(if `formula` includes the ID factor as a separate term) a character specifying which player to use as a reference, with the first level of the ID factor as the default. Overrides any other contrast specification for the ID factor.
`family`	a description of the error distribution and link function to be used in the model. Only the binomial family is implemented, with either`"logit"`, `"probit"` , or `"cauchit"` link. (See `stats::family()` for details of family functions.)
`data`	an optional object providing data required by the model. This may be a single data frame of contest-level data or a list of data frames. Names of data frames are ignored unless they refer to data frames specified by `player1` and `player2`. The rows of data frames that do not contain contest-level data must correspond to the levels of a factor used for indexing, i.e. row 1 corresponds to level 1, etc. Note any rownames are ignored. Objects are searched for first in the `data` object if provided, then in the environment of `formula`. If `data` is a list, the data frames are searched in the order given.
`weights`	an optional numeric vector of 'prior weights'.
`subset`	an optional logical or numeric vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when any contest-level variables contain `NA`s. The default is the `na.action` setting of `options`. See details for the handling of missing values in other variables.
`start`	a vector of starting values for the fixed effects.
`etastart`	a vector of starting values for the linear predictor.
`mustart`	a vector of starting values for the vector of means.

offset	an optional offset term in the model. A vector of length equal to the number of contests.
br	logical. If TRUE fitting will be by penalized maximum likelihood as in Firth (1992, 1993), using brglm::brglm(), rather than maximum likelihood using glm(), when abilities are modelled exactly or when the abilities are modelled by covariates and the variance of the random effects is estimated as zero.
model	logical: whether or not to return the model frame.
x	logical: whether or not to return the design matrix for the fixed effects.
contrasts	an optional list specifying contrasts for the factors in formula. See the contrasts.arg of model.matrix().
...	other arguments for fitting function (currently either glm(), brglm::brglm(), or glmmPQL())

Details

In each comparison to be modelled there is a 'first player' and a 'second player' and it is assumed that one player wins while the other loses (no allowance is made for tied comparisons).

The countsToBinomial() function is provided to convert a contingency table of wins into a data frame of wins and losses for each pair of players.

The formula argument specifies the model for player ability and applies to both the first player and the second player in each contest. If NULL a separate ability is estimated for each player, equivalent to setting formula = reformulate(id).

Contest-level variables can be specified in the formula in the usual manner, see formula(). Player covariates should be included as variables indexed by id, see examples. Thus player covariates must be ordered according to the levels of the ID factor.

If formula includes player covariates and there are players with missing values over these covariates, then a separate ability will be estimated for those players.

When player abilities are modelled by covariates, then random player effects should be added to the model. These should be specified in the formula using the vertical bar notation of lme4::lmer(), see examples.

When specified, it is assumed that random player effects arise from a $N(0,\sigma^2)$ distribution and model parameters, including σ, are estimated using PQL (Breslow and Clayton, 1993) as implemented in the glmmPQL() function.

Value

An object of class c("BTm", "x"), where "x" is the class of object returned by the model fitting function (e.g. glm). Components are as for objects of class "x", with additionally

id	the id argument.
separate.ability	the separate.ability argument.
refcat	the refcat argument.
player1	a data frame for the first player containing the ID factor and any player-specific contest-level variables.

player2	a data frame corresponding to that for player1.
assign	a numeric vector indicating which coefficients correspond to which terms in the model.
term.labels	labels for the model terms.
random	for models with random effects, the design matrix for the random effects.

Author(s)

Heather Turner, David Firth

References

Agresti, A. (2002) *Categorical Data Analysis* (2nd ed). New York: Wiley.

Firth, D. (1992) Bias reduction, the Jeffreys prior and GLIM. In *Advances in GLIM and Statistical Modelling*, Eds. Fahrmeir, L., Francis, B. J., Gilchrist, R. and Tutz, G., pp91–100. New York: Springer.

Firth, D. (1993) Bias reduction of maximum likelihood estimates. *Biometrika* **80**, 27–38.

Firth, D. (2005) Bradley-Terry models in R. *Journal of Statistical Software*, **12**(1), 1–12.

Stigler, S. (1994) Citation patterns in the journals of statistics and probability. *Statistical Science* **9**, 94–108.

Turner, H. and Firth, D. (2012) Bradley-Terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, **48**(9), 1–21.

See Also

countsToBinomial(), glmmPQL(), BTabilities(), residuals.BTm(), add1.BTm(), anova.BTm()

Examples

```
########################################################
##  Statistics journal citation data from Stigler (1994)
##  -- see also Agresti (2002, p448)
########################################################

##  Convert frequencies to success/failure data
citations.sf <- countsToBinomial(citations)
names(citations.sf)[1:2] <- c("journal1", "journal2")

##  First fit the "standard" Bradley-Terry model
citeModel <- BTm(cbind(win1, win2), journal1, journal2, data = citations.sf)

##  Now the same thing with a different "reference" journal
citeModel2 <- update(citeModel, refcat = "JASA")
BTabilities(citeModel2)

######################################################################
##  Now an example with an order effect -- see Agresti (2002) p438
######################################################################
```

```
data(baseball) # start with baseball data as provided by package

##  Simple Bradley-Terry model, ignoring home advantage:
baseballModel1 <- BTm(cbind(home.wins, away.wins), home.team, away.team,
                      data = baseball, id = "team")

##  Now incorporate the "home advantage" effect
baseball$home.team <- data.frame(team = baseball$home.team, at.home = 1)
baseball$away.team <- data.frame(team = baseball$away.team, at.home = 0)
baseballModel2 <- update(baseballModel1, formula = ~ team + at.home)

##  Compare the fit of these two models:
anova(baseballModel1, baseballModel2)

##
## For a more elaborate example with both player-level and contest-level
## predictor variables, see help(chameleons).
##
```

CEMS *Dittrich, Hatzinger and Katzenbeisser (1998, 2001) Data on Management School Preference in Europe*

Description

Community of European management schools (CEMS) data as used in the paper by Dittrich et al. (1998, 2001), re-formatted for use with `BTm()`

Usage

```
CEMS
```

Format

A list containing three data frames, `CEMS$preferences`, `CEMS$students` and `CEMS$schools`.

The `CEMS$preferences` data frame has `303 * 15 = 4505` observations (15 possible comparisons, for each of 303 students) on the following 8 variables:

student a factor with levels `1:303`

school1 a factor with levels `c("Barcelona", "London", "Milano", "Paris", "St.Gallen", "Stockholm")`; the first management school in a comparison

school2 a factor with the same levels as `school1`; the second management school in a comparison

win1 integer (value 0 or 1) indicating whether `school1` was preferred to `school2`

win2 integer (value 0 or 1) indicating whether `school2` was preferred to `school1`

tied integer (value 0 or 1) indicating whether no preference was expressed

win1.adj numeric, equal to `win1 + tied/2`

win2.adj numeric, equal to `win2 + tied/2`

The `CEMS$students` data frame has 303 observations (one for each student) on the following 8 variables:

STUD a factor with levels `c("other", "commerce")`, the student's main discipline of study

ENG a factor with levels `c("good, poor")`, indicating the student's knowledge of English

FRA a factor with levels `c("good, poor")`, indicating the student's knowledge of French

SPA a factor with levels `c("good, poor")`, indicating the student's knowledge of Spanish

ITA a factor with levels `c("good, poor")`, indicating the student's knowledge of Italian

WOR a factor with levels `c("no", "yes")`, whether the student was in full-time employment while studying

DEG a factor with levels `c("no", "yes")`, whether the student intended to take an international degree

SEX a factor with levels `c("female", "male")`

The `CEMS$schools` data frame has 6 observations (one for each management school) on the following 7 variables:

Barcelona numeric (value 0 or 1)

London numeric (value 0 or 1)

Milano numeric (value 0 or 1)

Paris numeric (value 0 or 1)

St.Gallen numeric (value 0 or 1)

Stockholm numeric (value 0 or 1)

LAT numeric (value 0 or 1) indicating a 'Latin' city

Details

The variables `win1.adj` and `win2.adj` are provided in order to allow a simple way of handling ties (in which a tie counts as half a win and half a loss), which is slightly different numerically from the Davidson (1970) method that is used by Dittrich et al. (1998): see the examples.

Author(s)

David Firth

Source

Royal Statistical Society datasets website, at https://rss.onlinelibrary.wiley.com/hub/journal/14679876/series-c-datasets/pre_2016.

References

Davidson, R. R. (1970) Extending the Bradley-Terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association* **65**, 317–328.

Dittrich, R., Hatzinger, R. and Katzenbeisser, W. (1998) Modelling the effect of subject-specific covariates in paired comparison studies with an application to university rankings. *Applied Statistics* **47**, 511–525.

Dittrich, R., Hatzinger, R. and Katzenbeisser, W. (2001) Corrigendum: Modelling the effect of subject-specific covariates in paired comparison studies with an application to university rankings. *Applied Statistics* **50**, 247–249.

Turner, H. and Firth, D. (2012) Bradley-Terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, **48**(9), 1–21.

Examples

```
##
##  Fit the standard Bradley-Terry model, using the simple 'add 0.5'
##  method to handle ties:
##
table3.model <-  BTm(outcome = cbind(win1.adj, win2.adj),
                     player1 = school1, player2 = school2,
                     formula = ~.. , refcat = "Stockholm",
                     data = CEMS)
##  The results in Table 3 of Dittrich et al (2001) are reproduced
##  approximately by a simple re-scaling of the estimates:
table3 <- summary(table3.model)$coef[, 1:2]/1.75
print(table3)
##
##  Now fit the 'final model' from Table 6 of Dittrich et al.:
##
table6.model <-  BTm(outcome = cbind(win1.adj, win2.adj),
                     player1 = school1, player2 = school2,
                     formula = ~ .. +
                         WOR[student] * Paris[..] +
                         WOR[student] * Milano[..] +
                         WOR[student] * Barcelona[..] +
                         DEG[student] * St.Gallen[..] +
                         STUD[student] * Paris[..] +
                         STUD[student] * St.Gallen[..] +
                         ENG[student] * St.Gallen[..] +
                         FRA[student] * London[..] +
                         FRA[student] * Paris[..] +
                         SPA[student] * Barcelona[..] +
                         ITA[student] * London[..] +
                         ITA[student] * Milano[..] +
                         SEX[student] * Milano[..],
                     refcat = "Stockholm",
                     data = CEMS)
##
##  Again re-scale to reproduce approximately Table 6 of Dittrich et
##  al. (2001):
```

```
##
table6 <- summary(table6.model)$coef[, 1:2]/1.75
print(table6)
##
## Not run:
##  Now the slightly simplified model of Table 8 of Dittrich et al. (2001):
##
table8.model <-  BTm(outcome = cbind(win1.adj, win2.adj),
                     player1 = school1, player2 = school2,
                     formula = ~ .. +
                         WOR[student] * LAT[..] +
                         DEG[student] * St.Gallen[..] +
                         STUD[student] * Paris[..] +
                         STUD[student] * St.Gallen[..] +
                         ENG[student] * St.Gallen[..] +
                         FRA[student] * London[..] +
                         FRA[student] * Paris[..] +
                         SPA[student] * Barcelona[..] +
                         ITA[student] * London[..] +
                         ITA[student] * Milano[..] +
                         SEX[student] * Milano[..],
                     refcat = "Stockholm",
                     data = CEMS)
table8 <- summary(table8.model)$coef[, 1:2]/1.75
##
##  Notice some larger than expected discrepancies here (the coefficients
##  named "..Barcelona", "..Milano" and "..Paris") from the results in
##  Dittrich et al. (2001).  Apparently a mistake was made in Table 8 of
##  the published Corrigendum note (R. Dittrich personal communication,
##  February 2010).
##
print(table8)

## End(Not run)
```

`chameleons` *Male Cape Dwarf Chameleons: Measured Traits and Contest Outcomes*

Description

Data as used in the study by Stuart-Fox et al. (2006). Physical measurements made on 35 male Cape dwarf chameleons, and the results of 106 inter-male contests.

Usage

```
chameleons
```

Format

A list containing three data frames: `chameleons$winner`, `chameleons$loser` and `chameleons$predictors`.

The `chameleons$winner` and `chameleons$loser` data frames each have 106 observations (one per contest) on the following 4 variables:

ID a factor with 35 levels `C01`, `C02`, ... , `C43`, the identity of the winning (or losing) male in each contest

prev.wins.1 integer (values 0 or 1), did the winner/loser of this contest win in an immediately previous contest?

prev.wins.2 integer (values 0, 1 or 2), how many of his (maximum) previous 2 contests did each male win?

prev.wins.all integer, how many previous contests has each male won?

The `chameleons$predictors` data frame has 35 observations, one for each male involved in the contests, on the following 7 variables:

ch.res numeric, residuals of casque height regression on `SVL`, i.e. relative height of the bony part on the top of the chameleons' heads

jl.res numeric, residuals of jaw length regression on `SVL`

tl.res numeric, residuals of tail length regression on `SVL`

mass.res numeric, residuals of body mass regression on `SVL` (body condition)

SVL numeric, snout-vent length (body size)

prop.main numeric, proportion (arcsin transformed) of area of the flank occupied by the main pink patch on the flank

prop.patch numeric, proportion (arcsin transformed) of area of the flank occupied by the entire flank patch

Details

The published paper mentions 107 contests, but only 106 contests are included here. Contest number 16 was deleted from the data used to fit the models, because it involved a male whose predictor-variables were incomplete (and it was the only contest involving that lizard, so it is uninformative).

Author(s)

David Firth

Source

The data were obtained by Dr Devi Stuart-Fox, https://devistuartfox.com/, and they are reproduced here with her kind permission.

These are the same data that were used in

Stuart-Fox, D. M., Firth, D., Moussalli, A. and Whiting, M. J. (2006) Multiple signals in chameleon contests: designing and analysing animal contests as a tournament. *Animal Behaviour* **71**, 1263–1271.

Examples

```
##
## Reproduce Table 3 from page 1268 of the above paper:
##
summary(chameleon.model <- BTm(player1 = winner, player2 = loser,
  formula = ~ prev.wins.2 + ch.res[ID] + prop.main[ID] + (1|ID), id = "ID",
  data = chameleons))
head(BTabilities(chameleon.model))
##
## Note that, although a per-chameleon random effect is specified as in the
## above [the term "+ (1|ID)"], the estimated variance for that random
## effect turns out to be zero in this case.  The "prior experience"
## effect ["+ prev.wins.2"] in this analysis has explained most of the
## variation, leaving little for the ID-specific predictors to do.
## Despite that, two of the ID-specific predictors do emerge as
## significant.
##
## Test whether any of the other ID-specific predictors has an effect:
##
add1(chameleon.model, ~ . + jl.res[ID] + tl.res[ID] + mass.res[ID] +
  SVL[ID] + prop.patch[ID])
```

---

citations *Statistics Journal Citation Data from Stigler (1994)*

Description

Extracted from a larger table in Stigler (1994). Inter-journal citation counts for four journals, "Biometrika", "Comm Statist.", "JASA" and "JRSS-B", as used on p448 of Agresti (2002).

Usage

```
citations
```

Format

A 4 by 4 contingency table of citations, cross-classified by the factors `cited` and `citing` each with levels `Biometrika`, `Comm Statist`, `JASA`, and `JRSS-B`.

Details

In the context of paired comparisons, the 'winner' is the cited journal and the 'loser' is the one doing the citing.

Source

Agresti, A. (2002) *Categorical Data Analysis* (2nd ed). New York: Wiley.

References

Firth, D. (2005) Bradley-Terry models in R. *Journal of Statistical Software* **12**(1), 1–12.

Turner, H. and Firth, D. (2012) Bradley-Terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, **48**(9), 1–21.

Stigler, S. (1994) Citation patterns in the journals of statistics and probability. *Statistical Science* **9**, 94–108.

See Also

`BTm()`

Examples

```
##  Data as a square table, as in Agresti p448
citations

##
## Convert frequencies to success/failure data:
##
citations.sf <- countsToBinomial(citations)
names(citations.sf)[1:2] <- c("journal1", "journal2")

## Standard Bradley-Terry model fitted to these data
citeModel <-  BTm(cbind(win1, win2), journal1, journal2,
                  data = citations.sf)
```

`countsToBinomial` *Convert Contingency Table of Wins to Binomial Counts*

Description

Convert a contingency table of wins to a four-column data frame containing the number of wins and losses for each pair of players.

Usage

```
countsToBinomial(xtab)
```

Arguments

`xtab`	a contingency table of wins cross-classified by "winner" and "loser"

Value

A data frame with four columns

`player1`	the first player in the contest.
`player2`	the second player in the contest.
`win1`	the number of times `player1` won.
`win2`	the number of times `player2` won.

Author(s)

Heather Turner

See Also

`BTm()`

Examples

```
##########################################################
##  Statistics journal citation data from Stigler (1994)
##  -- see also Agresti (2002, p448)
##########################################################
citations

## Convert frequencies to success/failure data
citations.sf <- countsToBinomial(citations)
names(citations.sf)[1:2] <- c("journal1", "journal2")
citations.sf
```

---

flatlizards *Augrabies Male Flat Lizards: Contest Results and Predictor Variables*

Description

Data collected at Augrabies Falls National Park (South Africa) in September-October 2002, on the contest performance and background attributes of 77 male flat lizards (*Platysaurus broadleyi*). The results of exactly 100 contests were recorded, along with various measurements made on each lizard. Full details of the study are in Whiting et al. (2006).

Usage

```
flatlizards
```

Format

This dataset is a list containing two data frames: `flatlizards$contests` and `flatlizards$predictors`.

The `flatlizards$contests` data frame has 100 observations on the following 2 variables:

winner a factor with 77 levels `lizard003` ... `lizard189`.

loser a factor with the same 77 levels `lizard003` ... `lizard189`.

The `flatlizards$predictors` data frame has 77 observations (one for each of the 77 lizards) on the following 18 variables:

id factor with 77 levels (3 5 6 ... 189), the lizard identifiers.

throat.PC1 numeric, the first principal component of the throat spectrum.

throat.PC2 numeric, the second principal component of the throat spectrum.

throat.PC3 numeric, the third principal component of the throat spectrum.

frontleg.PC1 numeric, the first principal component of the front-leg spectrum.

frontleg.PC2 numeric, the second principal component of the front-leg spectrum.

frontleg.PC3 numeric, the third principal component of the front-leg spectrum.

badge.PC1 numeric, the first principal component of the ventral colour patch spectrum.

badge.PC2 numeric, the second principal component of the ventral colour patch spectrum.

badge.PC3 numeric, the third principal component of the ventral colour patch spectrum.

badge.size numeric, a measure of the area of the ventral colour patch.

testosterone numeric, a measure of blood testosterone concentration.

SVL numeric, the snout-vent length of the lizard.

head.length numeric, head length.

head.width numeric, head width.

head.height numeric, head height.

condition numeric, a measure of body condition.

repro.tactic a factor indicating reproductive tactic; levels are `resident` and `floater`.

Details

There were no duplicate contests (no pair of lizards was seen fighting more than once), and there were no tied contests (the result of each contest was clear).

The variables `head.length`, `head.width`, `head.height` and `condition` were all computed as residuals (of directly measured head length, head width, head height and body mass index, respectively) from simple least-squares regressions on `SVL`.

Values of some predictors are missing (`NA`) for some lizards, 'at random', because of instrument problems unconnected with the value of the measurement being made.

Source

The data were collected by Dr Martin Whiting, https://whitinglab.com/people/martin-whiting/, and they appear here with his kind permission.

References

Turner, H. and Firth, D. (2012) Bradley-Terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, **48**(9), 1–21.

Whiting, M. J., Stuart-Fox, D. M., O'Connor, D., Firth, D., Bennett, N. C. and Blomberg, S. P. (2006). Ultraviolet signals ultra-aggression in a lizard. *Animal Behaviour* **72**, 353–363.

See Also

BTm()

Examples

```
##
##  Fit the standard Bradley-Terry model, using the bias-reduced
##  maximum likelihood method:
##
result <- rep(1, nrow(flatlizards$contests))
BTmodel <- BTm(result, winner, loser, br = TRUE, data = flatlizards$contests)
summary(BTmodel)
##
##  That's fairly useless, though, because of the rather small
##  amount of data on each lizard.  And really the scientific
##  interest is not in the abilities of these particular 77
##  lizards, but in the relationship between ability and the
##  measured predictor variables.
##
##  So next fit (by maximum likelihood) a "structured" B-T model in
##  which abilities are determined by a linear predictor.
##
##  This reproduces results reported in Table 1 of Whiting et al. (2006):
##
Whiting.model <- BTm(result, winner, loser,
                     ~ throat.PC1[..] + throat.PC3[..] +
                         head.length[..] + SVL[..],
                     data = flatlizards)
summary(Whiting.model)
##
##  Equivalently, fit the same model using glmmPQL:
##
Whiting.model <- BTm(result, winner, loser,
                     ~ throat.PC1[..] + throat.PC3[..] +
                         head.length[..] + SVL[..] + (1|..),
                     sigma = 0, sigma.fixed = TRUE, data = flatlizards)
summary(Whiting.model)
##
##  But that analysis assumes that the linear predictor formula for
##  abilities is _perfect_, i.e., that there is no error in the linear
##  predictor.  This will always be unrealistic.
##
##  So now fit the same predictor but with a normally distributed error
##  term --- a generalized linear mixed model --- by using the BTm
```

```
##  function instead of glm.
##
Whiting.model2 <- BTm(result, winner, loser,
                      ~ throat.PC1[..] + throat.PC3[..] +
                          head.length[..] + SVL[..] + (1|..),
                      data = flatlizards, trace = TRUE)
summary(Whiting.model2)
##
##  The estimated coefficients (of throat.PC1, throat.PC3,
##  head.length and SVL are not changed substantially by
##  the recognition of an error term in the model; but the estimated
##  standard errors are larger, as expected.  The main conclusions from
##  Whiting et al. (2006) are unaffected.
##
##  With the normally distributed random error included, it is perhaps
##  at least as natural to use probit rather than logit as the link
##  function:
##
require(stats)
Whiting.model3 <- BTm(result, winner, loser,
                      ~ throat.PC1[..] + throat.PC3[..] +
                          head.length[..] + SVL[..] + (1|..),
                      family = binomial(link = "probit"),
                      data = flatlizards, trace = TRUE)
summary(Whiting.model3)
BTabilities(Whiting.model3)
##  Note the "separate" attribute here, identifying two lizards with
##  missing values of at least one predictor variable
##
##  Modulo the usual scale change between logit and probit, the results
##  are (as expected) very similar to Whiting.model2.
```

`football` *English Premier League Football Results 2008/9 to 2012/13*

Description

The win/lose/draw results for five seasons of the English Premier League football results, from 2008/9 to 2012/13

Usage

```
football
```

Format

A data frame with 1881 observations on the following 4 variables.

season a factor with levels `2008-9`, `2009-10`, `2010-11`, `2011-12`, `2012-13`

home a factor specifying the home team, with 29 levels `Ars` (Arsenal), ... , `Wol` (Wolverhampton)

away a factor specifying the away team, with the same levels as `home`.

result a numeric vector giving the result for the home team: 1 for a win, 0 for a draw, -1 for a loss.

Details

In each season, there are 20 teams, each of which plays one home game and one away game against all the other teams in the league. The results in 380 games per season.

Source

These data were downloaded from http://soccernet.espn.go.com in 2013. The site has since moved and the new site does not appear to have an equivalent source.

References

Davidson, R. R. (1970). On extending the Bradley-Terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association*, **65**, 317–328.

See Also

`GenDavidson()`

Examples

```
### example requires gnm
if (require(gnm)) {
    ### convert to trinomial counts
    football.tri <- expandCategorical(football, "result", idvar = "match")
    head(football.tri)

    ### add variable to indicate whether team playing at home
    football.tri$at.home <- !logical(nrow(football.tri))

    ### fit Davidson model for ties
    ###  - subset to first and last season for illustration
    Davidson <- gnm(count ~
        GenDavidson(result == 1, result == 0, result == -1,
                    home:season, away:season,
                    home.adv = ~1, tie.max = ~1,
                    at.home1 = at.home, at.home2 = !at.home) - 1,
        eliminate = match, family = poisson, data = football.tri,
        subset = season %in% c("2008-9", "2012-13"))

    ### see ?GenDavidson for further analysis
}
```

GenDavidson *Specify a Generalised Davidson Term in a gnm Model Formula*

Description

GenDavidson is a function of class "nonlin" to specify a generalised Davidson term in the formula argument to gnm::gnm(), providing a model for paired comparison data where ties are a possible outcome.

Usage

```
GenDavidson(
  win,
  tie,
  loss,
  player1,
  player2,
  home.adv = NULL,
  tie.max = ~1,
  tie.mode = NULL,
  tie.scale = NULL,
  at.home1 = NULL,
  at.home2 = NULL
)
```

Arguments

win	a logical vector: TRUE if player1 wins, FALSE otherwise.
tie	a logical vector: TRUE if the outcome is a tie, FALSE otherwise.
loss	a logical vector: TRUE if player1 loses, FALSE otherwise.
player1	an ID factor specifying the first player in each contest, with the same set of levels as player2.
player2	an ID factor specifying the second player in each contest, with the same set of levels as player2.
home.adv	a formula for the parameter corresponding to the home advantage effect. If NULL, no home advantage effect is estimated.
tie.max	a formula for the parameter corresponding to the maximum tie probability.
tie.mode	a formula for the parameter corresponding to the location of maximum tie probability, in terms of the probability that player1 wins, given the outcome is not a draw.
tie.scale	a formula for the parameter corresponding to the scale of dependence of the tie probability on the probability that player1 wins, given the outcome is not a draw.
at.home1	a logical vector: TRUE if player1 is at home, FALSE otherwise.
at.home2	a logical vector: TRUE if player2 is at home, FALSE otherwise.

Details

`GenDavidson` specifies a generalisation of the Davidson model (1970) for paired comparisons where a tie is a possible outcome. It is designed for modelling trinomial counts corresponding to the win/draw/loss outcome for each contest, which are assumed Poisson conditional on the total count for each match. Since this total must be one, the expected counts are equivalently the probabilities for each possible outcome, which are modelled on the log scale:

$$\log(p(i\text{beats}j)_k) = \theta_{ijk} + \log(\mu\alpha_i$$

$$\log(p(draw)_k) = \theta_{ijk} + \delta + c+$$

$$\sigma(\pi \log(\mu\alpha_i) - (1-\pi)log(\alpha_j))+$$

$$(1-\sigma)(\log(\mu\alpha_i + \alpha_j))$$

$$\log(p(j\text{beats}i)_k) = \theta_{ijk}+$$

$$log(\alpha_j)$$

Here θ_{ijk} is a structural parameter to fix the trinomial totals; μ is the home advantage parameter; α_i and α_j are the abilities of players i and j respectively; c is a function of the parameters such that expit(δ) is the maximum probability of a tie, σ scales the dependence of the probability of a tie on the relative abilities and π allows for asymmetry in this dependence.

For parameters that must be positive (α_i, σ, μ), the log is estimated, while for parameters that must be between zero and one (δ, π), the logit is estimated, as illustrated in the example.

Value

A list with the anticipated components of a "nonlin" function:

`predictors`	the formulae for the different parameters and the ID factors for player 1 and player 2.
`variables`	the outcome variables and the "at home" variables, if specified.
`common`	an index to specify that common effects are to be estimated for the players.
`term`	a function to create a deparsed mathematical expression of the term, given labels for the predictors.
`start`	a function to generate starting values for the parameters.

Author(s)

Heather Turner

References

Davidson, R. R. (1970). On extending the Bradley-Terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association*, **65**, 317–328.

See Also

`football()`, `plotProportions()`

Examples

```
### example requires gnm
if (require(gnm)) {
    ### convert to trinomial counts
    football.tri <- expandCategorical(football, "result", idvar = "match")
    head(football.tri)

    ### add variable to indicate whether team playing at home
    football.tri$at.home <- !logical(nrow(football.tri))

    ### fit shifted & scaled Davidson model
    ###  - subset to first and last season for illustration
    shifScalDav <- gnm(count ~
        GenDavidson(result == 1, result == 0, result == -1,
                    home:season, away:season, home.adv = ~1,
                    tie.max = ~1, tie.scale = ~1, tie.mode = ~1,
                    at.home1 = at.home,
                    at.home2 = !at.home) - 1,
        eliminate = match, family = poisson, data = football.tri,
        subset = season %in% c("2008-9", "2012-13"))

    ### look at coefs
    coef <- coef(shifScalDav)
    ## home advantage
    exp(coef["home.adv"])
    ## max p(tie)
    plogis(coef["tie.max"])
    ## mode p(tie)
    plogis(coef["tie.mode"])
    ## scale relative to Davidson of dependence of p(tie) on p(win|not a draw)
    exp(coef["tie.scale"])

    ### check model fit
    alpha <- names(coef[-(1:4)])
    plotProportions(result == 1, result == 0, result == -1,
                    home:season, away:season,
                    abilities = coef[alpha], home.adv = coef["home.adv"],
                    tie.max = coef["tie.max"], tie.scale = coef["tie.scale"],
                    tie.mode = coef["tie.mode"],
                    at.home1 = at.home, at.home2 = !at.home,
                    data = football.tri, subset = count == 1)
}

### analyse all five seasons
### - takes a little while to run, particularly likelihood ratio tests
## Not run:
### fit Davidson model
Dav <- gnm(count ~ GenDavidson(result == 1, result == 0, result == -1,
                               home:season, away:season, home.adv = ~1,
                               tie.max = ~1,
                               at.home1 = at.home,
                               at.home2 = !at.home) - 1,
```

```
            eliminate = match, family = poisson, data = football.tri)

### fit scaled Davidson model
scalDav <- gnm(count ~ GenDavidson(result == 1, result == 0, result == -1,
                                   home:season, away:season, home.adv = ~1,
                                   tie.max = ~1, tie.scale = ~1,
                                   at.home1 = at.home,
                                   at.home2 = !at.home) - 1,
               eliminate = match, family = poisson, data = football.tri)

### fit shifted & scaled Davidson model
shifScalDav <- gnm(count ~
    GenDavidson(result == 1, result == 0, result == -1,
                home:season, away:season, home.adv = ~1,
                tie.max = ~1, tie.scale = ~1, tie.mode = ~1,
                at.home1 = at.home,
                at.home2 = !at.home) - 1,
    eliminate = match, family = poisson, data = football.tri)

### compare models
anova(Dav, scalDav, shifScalDav, test = "Chisq")

### diagnostic plots
main <- c("Davidson", "Scaled Davidson", "Shifted & Scaled Davidson")
mod <- list(Dav, scalDav, shifScalDav)
names(mod) <- main

## use football.tri data so that at.home can be found,
## but restrict to actual match results
par(mfrow = c(2,2))
for (i in 1:3) {
    coef <- parameters(mod[[i]])
    plotProportions(result == 1, result == 0, result == -1,
                    home:season, away:season,
                    abilities = coef[alpha],
                    home.adv = coef["home.adv"],
                    tie.max = coef["tie.max"],
                    tie.scale = coef["tie.scale"],
                    tie.mode = coef["tie.mode"],
                    at.home1 = at.home,
                    at.home2 = !at.home,
                    main = main[i],
                    data = football.tri, subset = count == 1)
}

## End(Not run)
```

Description

Fits GLMMs with simple random effects structure via Breslow and Clayton's PQL algorithm. The GLMM is assumed to be of the form

$$g(\boldsymbol{\mu}) = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{e}$$

where g is the link function, $\boldsymbol{\mu}$ is the vector of means and $\boldsymbol{X}, \boldsymbol{Z}$ are design matrices for the fixed effects $\boldsymbol{\beta}$ and random effects $\boldsymbol{e}$ respectively. Furthermore the random effects are assumed to be i.i.d. $N(0, \sigma^2)$.

Usage

```
glmmPQL(
  fixed,
  random = NULL,
  family = "binomial",
  data = NULL,
  subset = NULL,
  weights = NULL,
  offset = NULL,
  na.action = NULL,
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  control = glmmPQL.control(...),
  sigma = 0.1,
  sigma.fixed = FALSE,
  model = TRUE,
  x = FALSE,
  contrasts = NULL,
  ...
)
```

Arguments

`fixed`	a formula for the fixed effects.
`random`	a design matrix for the random effects, with number of rows equal to the length of variables in `formula`.
`family`	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See `family()` for details of family functions.)
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame()` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `glmmPQL` called.
`subset`	an optional logical or numeric vector specifying a subset of observations to be used in the fitting process.

weights	an optional vector of 'prior weights' to be used in the fitting process.
offset	an optional numeric vector to be added to the linear predictor during fitting. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset().
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options(), and is na.fail() if that is unset.
start	starting values for the parameters in the linear predictor.
etastart	starting values for the linear predictor.
mustart	starting values for the vector of means.
control	a list of parameters for controlling the fitting process. See the glmmPQL.control() for details.
sigma	a starting value for the standard deviation of the random effects.
sigma.fixed	logical: whether or not the standard deviation of the random effects should be fixed at its starting value.
model	logical: whether or not the model frame should be returned.
x	logical: whether or not the design matrix for the fixed effects should be returned.
contrasts	an optional list. See the contrasts.arg argument of model.matrix().
...	arguments to be passed to glmmPQL.control().

Value

An object of class "BTglmmPQL" which inherits from "glm" and "lm":

coefficients	a named vector of coefficients, with a "random" attribute giving the estimated random effects.
residuals	the working residuals from the final iteration of the IWLS loop.
random	the design matrix for the random effects.
fitted.values	the fitted mean values, obtained by transforming the linear predictors by the inverse of the link function.
rank	the numeric rank of the fitted linear model.
family	the family object used.
linear.predictors	the linear fit on link scale.
deviance	up to a constant, minus twice the maximized log-likelihood.
aic	a version of Akaike's *An Information Criterion*, minus twice the maximized log-likelihood plus twice the number of parameters, computed by the aic component of the family.
null.deviance	the deviance for the null model, comparable with deviance.
iter	the numer of iterations of the PQL algorithm.
weights	the working weights, that is the weights in the final iteration of the IWLS loop.
prior.weights	the weights initially supplied, a vector of 1's if none were.

df.residual	the residual degrees of freedom.
df.null	the residual degrees of freedom for the null model.
y	if requested (the default) the y vector used. (It is a vector even for a binomial model.)
x	if requested, the model matrix.
model	if requested (the default), the model frame.
converged	logical. Was the PQL algorithm judged to have converged?
call	the matched call.
formula	the formula supplied.
terms	the terms object used.
data	the data argument used.
offset	the offset vector used.
control	the value of the control argument used.
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting.
na.action	(where relevant) information returned by model.frame on the special handling of NAs.
sigma	the estimated standard deviation of the random effects
sigma.fixed	logical: whether or not sigma was fixed
varFix	the variance-covariance matrix of the fixed effects
varSigma	the variance of sigma

Author(s)

Heather Turner

References

Breslow, N. E. and Clayton, D. G. (1993) Approximate inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association* **88**(421), 9–25.

Harville, D. A. (1977) Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association* **72**(358), 320–338.

See Also

predict.BTglmmPQL(),glmmPQL.control(),BTm()

Examples

```
##############################################
## Crowder seeds example from Breslow & Clayton
##############################################

summary(glmmPQL(cbind(r, n - r) ~ seed + extract,
        random = diag(nrow(seeds)),
        family = "binomial", data = seeds))

summary(glmmPQL(cbind(r, n - r) ~ seed*extract,
                random = diag(nrow(seeds)),
                family = "binomial", data = seeds))
```

`glmmPQL.control` *Control Aspects of the glmmPQL Algorithm*

Description

Set control variables for the glmmPQL algorithm.

Usage

```
glmmPQL.control(maxiter = 50, IWLSiter = 10, tol = 1e-06, trace = FALSE)
```

Arguments

`maxiter`	the maximum number of outer iterations.
`IWLSiter`	the maximum number of iterated weighted least squares iterations used to estimate the fixed effects, given the standard deviation of the random effects.
`tol`	the tolerance used to determine convergence in the IWLS iterations and over all (see details).
`trace`	logical: whether or not to print the score for the random effects variance at the end of each iteration.

Details

This function provides an interface to control the PQL algorithm used by `BTm()` for fitting Bradley Terry models with random effects.

The algorithm iterates between a series of iterated weighted least squares iterations to update the fixed effects and a single Fisher scoring iteration to update the standard deviation of the random effects.

Convergence of both the inner and outer iterations are judged by comparing the squared components of the relevant score vector with corresponding elements of the diagonal of the Fisher information matrix. If, for all components of the relevant score vector, the ratio is less than `tolerance^2`, or the corresponding diagonal element of the Fisher information matrix is less than 1e-20, iterations cease.

Value

A list with the arguments as components.

Author(s)

Heather Turner

References

Breslow, N. E. and Clayton, D. G. (1993), Approximate inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association* **88**(421), 9–25.

See Also

`glmmPQL()`, `BTm()`

Examples

```
## Variation on example(flatlizards)
result <- rep(1, nrow(flatlizards$contests))

## BTm passes arguments on to glmmPQL.control()
args(BTm)
BTmodel <- BTm(result, winner, loser, ~ throat.PC1[..] + throat.PC3[..] +
               head.length[..] + SVL[..] + (1|..),
               data = flatlizards, tol = 1e-3, trace = TRUE)
summary(BTmodel)
```

---

icehockey *College Hockey Men's Division I 2009-10 results*

Description

Game results from American College Hockey Men's Division I composite schedule 2009-2010.

Usage

```
icehockey
```

Format

A data frame with 1083 observations on the following 6 variables.

date a numeric vector

visitor a factor with 58 levels `Alaska` `Anchorage` ... `Yale`

v_goals a numeric vector

opponent a factor with 58 levels `Alaska` `Anchorage` ... `Yale`

o_goals a numeric vector

conference a factor with levels AH, CC, CH, EC, HE, NC, WC

result a numeric vector: 1 if visitor won, 0.5 for a draw and 0 if visitor lost

home.ice a logical vector: 1 if opponent on home ice, 0 if game on neutral ground

Details

The Division I ice hockey teams are arranged in six conferences: Atlantic Hockey, Central Collegiate Hockey Association, College Hockey America, ECAC Hockey, Hockey East and the Western Collegiate Hockey Association, all part of the National Collegiate Athletic Association. The composite schedule includes within conference games and between conference games.

The data set here contains only games from the regular season, the results of which determine the teams that play in the NCAA national tournament. There are six automatic bids that go to the conference tournament champions, the remaining 10 teams are selected based upon ranking under the NCAA's system of pairwise comparisons (https://www.collegehockeynews.com/info/?d=pwcrpi). Some have argued that Bradley-Terry rankings would be fairer (https://www.collegehockeynews.com/info/?d=krach).

Source

http://www.collegehockeystats.net/0910/schedules/men.

References

Schlobotnik, J. Build your own rankings: http://www.elynah.com/tbrw/2010/rankings.diy.shtml.

College Hockey News https://www.collegehockeynews.com/.

Selections for 2010 NCAA tournament: https://www.espn.com/college-sports/news/story?id=5012918.

Examples

```
### Fit the standard Bradley-Terry model
standardBT <- BTm(outcome = result,
    player1 = visitor, player2 = opponent,
    id = "team", data = icehockey)

## Bradley-Terry abilities
abilities <- exp(BTabilities(standardBT)[,1])

## Compute round-robin winning probability and KRACH ratings
## (scaled abilities such that KRACH = 100 for a team with
## round-robin winning probability of 0.5)
rankings <- function(abilities){
    probwin <- abilities/outer(abilities, abilities, "+")
    diag(probwin) <- 0
    nteams <- ncol(probwin)
    RRWP <- rowSums(probwin)/(nteams - 1)
    low <- quantile(abilities, 0.45)
```

```
    high <- quantile(abilities, 0.55)
    middling <- uniroot(function(x) {sum(x/(x+abilities)) - 0.5*nteams},
                        lower = low, upper = high)$root
    KRACH <- abilities/middling*100
    cbind(KRACH, RRWP)
}

ranks <- rankings(abilities)
## matches those produced by Joe Schlobotnik's Build Your Own Rankings
head(signif(ranks, 4)[order(ranks[,1], decreasing = TRUE),])

## At one point the NCAA rankings gave more credit for wins on
## neutral/opponent's ground. Home ice effects are easily
## incorporated into the Bradley-Terry model, comparing teams
## on a "level playing field"
levelBT <- BTm(result,
               data.frame(team = visitor, home.ice = 0),
               data.frame(team = opponent, home.ice = home.ice),
               ~ team + home.ice,
               id = "team", data = icehockey)

abilities <- exp(BTabilities(levelBT)[,1])
ranks2 <- rankings(abilities)

## Look at movement between the two rankings
change <- factor(rank(ranks2[,1]) - rank(ranks[,1]))
barplot(xtabs(~change), xlab = "Change in Rank", ylab = "No. Teams")

## Take out regional winners and look at top 10
regional <- c("RIT", "Alabama-Huntsville", "Michigan", "Cornell", "Boston College",
              "North Dakota")

ranks <- ranks[!rownames(ranks) %in% regional]
ranks2 <- ranks2[!rownames(ranks2) %in% regional]

## compare the 10 at-large selections under both rankings
## with those selected under NCAA rankings
cbind(names(sort(ranks, decr = TRUE)[1:10]),
      names(sort(ranks2, decr = TRUE)[1:10]),
      c("Miami", "Denver", "Wisconsin", "St. Cloud State",
        "Bemidji State", "Yale", "Northern Michigan", "New Hampshire",
        "Alsaka", "Vermont"))
```

---

plotProportions *Plot Proportions of Tied Matches and Non-tied Matches Won*

Description

Plot proportions of tied matches and non-tied matches won by the first player, within matches binned by the relative player ability, as expressed by the probability that the first player wins, given the match is not a tie. Add fitted lines for each set of matches, as given by the generalized Davidson model.

Usage

```
plotProportions(
  win,
  tie = NULL,
  loss,
  player1,
  player2,
  abilities = NULL,
  home.adv = NULL,
  tie.max = NULL,
  tie.scale = NULL,
  tie.mode = NULL,
  at.home1 = NULL,
  at.home2 = NULL,
  data = NULL,
  subset = NULL,
  bin.size = 20,
  xlab = "P(player1 wins | not a tie)",
  ylab = "Proportion",
  legend = NULL,
  col = 1:2,
  ...
)
```

Arguments

win	a logical vector: TRUE if player1 wins, FALSE otherwise.
tie	a logical vector: TRUE if the outcome is a tie, FALSE otherwise (NULL if there are no ties).
loss	a logical vector: TRUE if player1 loses, FALSE otherwise.
player1	an ID factor specifying the first player in each contest, with the same set of levels as player2.
player2	an ID factor specifying the second player in each contest, with the same set of levels as player2.
abilities	the fitted abilities from a generalized Davidson model (or a Bradley-Terry model).
home.adv	if applicable, the fitted home advantage parameter from a generalized Davidson model (or a Bradley-Terry model).
tie.max	the fitted parameter from a generalized Davidson model corresponding to the maximum tie probability.

tie.scale	if applicable, the fitted parameter from a generalized Davidson model corresponding to the scale of dependence of the tie probability on the probability that player1 wins, given the outcome is not a draw.
tie.mode	if applicable, the fitted parameter from a generalized Davidson model corresponding to the location of maximum tie probability, in terms of the probability that player1 wins, given the outcome is not a draw.
at.home1	a logical vector: TRUE if player1 is at home, FALSE otherwise.
at.home2	a logical vector: TRUE if player2 is at home, FALSE otherwise.
data	an optional data frame providing variables required by the model, with one observation per match.
subset	an optional logical or numeric vector specifying a subset of observations to include in the plot.
bin.size	the approximate number of matches in each bin.
xlab	the label to use for the x-axis.
ylab	the label to use for the y-axis.
legend	text to use for the legend.
col	a vector specifying colours to use for the proportion of non-tied matches won and the proportion of tied matches.
...	further arguments passed to plot.

Details

If home.adv is specified, the results are re-ordered if necessary so that the home player comes first; any matches played on neutral ground are omitted.

First the probability that the first player wins given that the match is not a tie is computed:

$$expit(home.adv + abilities[player1] - abilities[player2])$$

where home.adv and abilities are parameters from a generalized Davidson model that have been estimated on the log scale.

The matches are then binned according to this probability, grouping together matches with similar relative ability between the first player and the second player. Within each bin, the proportion of tied matches is computed and these proportions are plotted against the mid-point of the bin. Then the bins are re-computed omitting the tied games and the proportion of non-tied matches won by the first player is found and plotted against the new mid-point.

Finally curves are added for the probability of a tie and the conditional probability of win given the match is not a tie, under a generalized Davidson model with parameters as specified by tie.max, tie.scale and tie.mode.

The function can also be used to plot the proportions of wins along with the fitted probability of a win under the Bradley-Terry model.

Value

A list of data frames:

`win`	a data frame comprising `prop.win`, the proportion of non-tied matches won by the first player in each bin and `bin.win`, the mid-point of each bin.
`tie`	(when ties are present) a data frame comprising `prop.tie`, the proportion of tied matches in each bin and `bin.tie`, the mid-point of each bin.

Note

This function is designed for single match outcomes, therefore data aggregated over player pairs will need to be expanded.

Author(s)

Heather Turner

See Also

`GenDavidson()`, `BTm()`

Examples

```
#### A Bradley-Terry example using icehockey data

## Fit the standard Bradley-Terry model, ignoring home advantage
standardBT <- BTm(outcome = result,
                  player1 = visitor, player2 = opponent,
                  id = "team", data = icehockey)

## comparing teams on a "level playing field"
levelBT <- BTm(result,
               data.frame(team = visitor, home.ice = 0),
               data.frame(team = opponent, home.ice = home.ice),
               ~ team + home.ice,
               id = "team", data = icehockey)

## compare fit to observed proportion won
## exclude tied matches as not explicitly modelled here
par(mfrow = c(1, 2))
plotProportions(win = result == 1, loss = result == 0,
                player1 = visitor, player2 = opponent,
                abilities = BTabilities(standardBT)[,1],
                data = icehockey, subset = result != 0.5,
                main = "Without home advantage")

plotProportions(win = result == 1, loss = result == 0,
                player1 = visitor, player2 = opponent,
                home.adv = coef(levelBT)["home.ice"],
                at.home1 = 0, at.home2 = home.ice,
                abilities = BTabilities(levelBT)[,1],
```

```
                data = icehockey, subset = result != 0.5,
                main = "With home advantage")

#### A generalized Davidson example using football data
if (require(gnm)) {

    ## subset to first and last season for illustration
    football <- subset(football, season %in% c("2008-9", "2012-13"))

    ## convert to trinomial counts
    football.tri <- expandCategorical(football, "result", idvar = "match")

    ## add variable to indicate whether team playing at home
    football.tri$at.home <- !logical(nrow(football.tri))

    ## fit Davidson model
    Dav <- gnm(count ~ GenDavidson(result == 1, result == 0, result == -1,
                                   home:season, away:season, home.adv = ~1,
                                   tie.max = ~1,
                                   at.home1 = at.home,
                                   at.home2 = !at.home) - 1,
               eliminate = match, family = poisson, data = football.tri)

    ## fit shifted & scaled Davidson model
    shifScalDav <- gnm(count ~
        GenDavidson(result == 1, result == 0, result == -1,
                    home:season, away:season, home.adv = ~1,
                    tie.max = ~1, tie.scale = ~1, tie.mode = ~1,
                    at.home1 = at.home,
                    at.home2 = !at.home) - 1,
        eliminate = match, family = poisson, data = football.tri)

    ## diagnostic plots
    main <- c("Davidson", "Shifted & Scaled Davidson")
    mod <- list(Dav, shifScalDav)
    names(mod) <- main
    alpha <- names(coef(Dav)[-(1:2)])

    ## use football.tri data so that at.home can be found,
    ## but restrict to actual match results
    par(mfrow = c(1,2))
    for (i in 1:2) {
        coef <- parameters(mod[[i]])
        plotProportions(result == 1, result == 0, result == -1,
                        home:season, away:season,
                        abilities = coef[alpha],
                        home.adv = coef["home.adv"],
                        tie.max = coef["tie.max"],
                        tie.scale = coef["tie.scale"],
                        tie.mode = coef["tie.mode"],
                        at.home1 = at.home,
                        at.home2 = !at.home,
                        main = main[i],
```

```
                        data = football.tri, subset = count == 1)
    }
}
```

---

predict.BTglmmPQL *Predict Method for BTglmmPQL Objects*

Description

Obtain predictions and optionally standard errors of those predictions from a "BTglmmPQL" object.

Usage

```
## S3 method for class 'BTglmmPQL'
predict(
  object,
  newdata = NULL,
  newrandom = NULL,
  level = ifelse(object$sigma == 0, 0, 1),
  type = c("link", "response", "terms"),
  se.fit = FALSE,
  terms = NULL,
  na.action = na.pass,
  ...
)
```

Arguments

object	a fitted object of class "BTglmmPQL"
newdata	(optional) a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
newrandom	if newdata is provided, a corresponding design matrix for the random effects, will columns corresponding to the random effects estimated in the original model.
level	an integer vector giving the level(s) at which predictions are required. Level zero corresponds to population-level predictions (fixed effects only), whilst level one corresponds to the individual-level predictions (full model) which are NA for contests involving individuals not in the original data. By default level = 0 if the model converged to a fixed effects model, 1 otherwise.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale (fixed effects only).
se.fit	logical switch indicating if standard errors are required.

terms	with type ="terms" by default all terms are returned. A character vector specifies which terms are to be returned.
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	further arguments passed to or from other methods.

Details

If newdata is omitted the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit are treated is determined by the na.action argument of that fit. If na.action = na.omit omitted cases will not appear in the residuals, whereas if na.action = na.exclude they will appear (in predictions and standard errors), with residual value NA. See also napredict.

Standard errors for the predictions are approximated assuming the variance of the random effects is known, see Booth and Hobert (1998).

Value

If se.fit = FALSE, a vector or matrix of predictions. If se = TRUE, a list with components

fit	Predictions
se.fit	Estimated standard errors

Author(s)

Heather Turner

References

Booth, J. G. and Hobert, J. P. (1998). Standard errors of prediction in Generalized Linear Mixed Models. *Journal of the American Statistical Association* **93**(441), 262 – 272.

See Also

predict.glm(), predict.BTm()

Examples

```
seedsModel <- glmmPQL(cbind(r, n - r) ~ seed + extract,
                      random = diag(nrow(seeds)),
                      family = binomial,
                      data = seeds)

pred <- predict(seedsModel, level = 0)
predTerms <- predict(seedsModel, type = "terms")

all.equal(pred, rowSums(predTerms) + attr(predTerms, "constant"))
```

predict.BTm *Predict Method for Bradley-Terry Models*

Description

Obtain predictions and optionally standard errors of those predictions from a fitted Bradley-Terry model.

Usage

```
## S3 method for class 'BTm'
predict(
  object,
  newdata = NULL,
  level = ifelse(is.null(object$random), 0, 1),
  type = c("link", "response", "terms"),
  se.fit = FALSE,
  dispersion = NULL,
  terms = NULL,
  na.action = na.pass,
  ...
)
```

Arguments

object	a fitted object of class "BTm"
newdata	(optional) a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
level	for models with random effects: an integer vector giving the level(s) at which predictions are required. Level zero corresponds to population-level predictions (fixed effects only), whilst level one corresponds to the player-level predictions (full model) which are NA for contests involving players not in the original data. By default, level = 0 for a fixed effects model, 1 otherwise.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default Bradley-Terry model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale (fixed effects only).
se.fit	logical switch indicating if standard errors are required.
dispersion	a value for the dispersion, not used for models with random effects. If omitted, that returned by summary applied to the object is used, where applicable.
terms	with type ="terms" by default all terms are returned. A character vector specifies which terms are to be returned.

na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
...	further arguments passed to or from other methods.

Details

If newdata is omitted the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit are treated is determined by the na.action argument of that fit. If na.action = na.omit omitted cases will not appear in the residuals, whereas if na.action = na.exclude they will appear (in predictions and standard errors), with residual value NA. See also napredict.

Value

If se.fit = FALSE, a vector or matrix of predictions. If se = TRUE, a list with components

fit	Predictions
se.fit	Estimated standard errors

Author(s)

Heather Turner

See Also

predict.glm(), MASS::predict.glmmPQL()

Examples

```
## The final model in example(flatlizards)
result <- rep(1, nrow(flatlizards$contests))
Whiting.model3 <- BTm(1, winner, loser, ~ throat.PC1[..] + throat.PC3[..] +
                      head.length[..] + SVL[..] + (1|..),
                      family = binomial(link = "probit"),
                      data = flatlizards, trace = TRUE)

## `new' data for contests between four of the original lizards
## factor levels must correspond to original levels, but unused levels
## can be dropped - levels must match rows of predictors
newdata  <- list(contests = data.frame(
                winner = factor(c("lizard048", "lizard060"),
                levels = c("lizard006", "lizard011",
                           "lizard048", "lizard060")),
                loser = factor(c("lizard006", "lizard011"),
                levels = c("lizard006", "lizard011",
                           "lizard048", "lizard060"))
                ),
                predictors = flatlizards$predictors[c(3, 6, 27, 33), ])

predict(Whiting.model3, level = 1, newdata = newdata)
```

```
## same as
predict(Whiting.model3, level = 1)[1:2]

## introducing a new lizard
newpred <- rbind(flatlizards$predictors[c(3, 6, 27),
                    c("throat.PC1","throat.PC3", "SVL", "head.length")],
                 c(-5, 1.5, 1, 0.1))
rownames(newpred)[4] <- "lizard059"

newdata  <- list(contests = data.frame(
                 winner = factor(c("lizard048", "lizard059"),
                 levels = c("lizard006", "lizard011",
                            "lizard048", "lizard059")),
                 loser = factor(c("lizard006", "lizard011"),
                 levels = c("lizard006", "lizard011",
                            "lizard048", "lizard059"))
                 ),
                 predictors = newpred)

## can only predict at population level for contest with new lizard
predict(Whiting.model3, level = 0:1, se.fit = TRUE, newdata = newdata)

## predicting at specific levels of covariates

## consider a model from example(CEMS)
table6.model <-  BTm(outcome = cbind(win1.adj, win2.adj),
                     player1 = school1, player2 = school2,
                     formula = ~ .. +
                         WOR[student] * Paris[..] +
                         WOR[student] * Milano[..] +
                         WOR[student] * Barcelona[..] +
                         DEG[student] * St.Gallen[..] +
                         STUD[student] * Paris[..] +
                         STUD[student] * St.Gallen[..] +
                         ENG[student] * St.Gallen[..] +
                         FRA[student] * London[..] +
                         FRA[student] * Paris[..] +
                         SPA[student] * Barcelona[..] +
                         ITA[student] * London[..] +
                         ITA[student] * Milano[..] +
                         SEX[student] * Milano[..],
                     refcat = "Stockholm",
                     data = CEMS)

## estimate abilities for a combination not seen in the original data

## same schools
schools <- levels(CEMS$preferences$school1)
## new student data
students <- data.frame(STUD = "other", ENG = "good", FRA = "good",
                       SPA = "good", ITA = "good", WOR = "yes", DEG = "no",
                       SEX = "female", stringsAsFactors = FALSE)
## set levels to be the same as original data
```

```
for (i in seq_len(ncol(students))){
    students[,i] <- factor(students[,i], levels(CEMS$students[,i]))
}
newdata <- list(preferences =
    data.frame(student = factor(500), # new id matching with `students[1,]`
               school1 = factor("London", levels = schools),
               school2 = factor("Paris", levels = schools)),
    students = students,
    schools = CEMS$schools)

## warning can be ignored as model specification was over-parameterized
predict(table6.model, newdata = newdata)

## if treatment contrasts are use (i.e. one player is set as the reference
## category), then predicting the outcome of contests against the reference
## is equivalent to estimating abilities with specific covariate values

## add student with all values at reference levels
students <- rbind(students,
    data.frame(STUD = "other", ENG = "good", FRA = "good",
               SPA = "good", ITA = "good", WOR = "no", DEG = "no",
               SEX = "female", stringsAsFactors = FALSE))
## set levels to be the same as original data
for (i in seq_len(ncol(students))){
    students[,i] <- factor(students[,i], levels(CEMS$students[,i]))
}
newdata <- list(preferences =
    data.frame(student = factor(rep(c(500, 502), each = 6)),
               school1 = factor(schools, levels = schools),
               school2 = factor("Stockholm", levels = schools)),
    students = students,
    schools = CEMS$schools)

predict(table6.model, newdata = newdata, se.fit = TRUE)

## the second set of predictions (elements 7-12) are equivalent to the output
## of BTabilities; the first set are adjust for `WOR` being equal to "yes"
BTabilities(table6.model)
```

`qvcalc.BTabilities` *Quasi Variances for Estimated Abilities*

Description

A method for `qvcalc::qvcalc()` to compute a set of quasi variances (and corresponding quasi standard errors) for estimated abilities from a Bradley-Terry model as returned by `BTabilities()`.

Usage

```
## S3 method for class 'BTabilities'
qvcalc(object, ...)
```

Arguments

object	a "BTabilities" object as returned by BTabilities().
...	additional arguments, currently ignored.

Details

For details of the method see Firth (2000), Firth (2003) or Firth and de Menezes (2004). Quasi variances generalize and improve the accuracy of "floating absolute risk" (Easton et al., 1991). This device for economical model summary was first suggested by Ridout (1989).

Ordinarily the quasi variances are positive and so their square roots (the quasi standard errors) exist and can be used in plots, etc.

Value

A list of class "qv", with components

covmat	The full variance-covariance matrix for the estimated abilities.
qvframe	A data frame with variables estimate, SE, quasiSE and quasiVar, the last two being a quasi standard error and quasi-variance for each ability.
dispersion	NULL (dispersion is fixed to 1).
relerrs	Relative errors for approximating the standard errors of all simple contrasts.
factorname	The name of the ID factor identifying players in the BTm formula.
coef.indices	NULL (no required for this method).
modelcall	The call to BTm to fit the Bradley-Terry model from which the abilities were estimated.

Author(s)

David Firth

References

Easton, D. F, Peto, J. and Babiker, A. G. A. G. (1991) Floating absolute risk: an alternative to relative risk in survival and case-control analysis avoiding an arbitrary reference group. *Statistics in Medicine* **10**, 1025–1035.

Firth, D. (2000) Quasi-variances in Xlisp-Stat and on the web. *Journal of Statistical Software* **5(4)**, 1–13. doi:10.18637/jss.v005.i04.

Firth, D. (2003) Overcoming the reference category problem in the presentation of statistical models. *Sociological Methodology* **33**, 1–18.

Firth, D. and de Menezes, R. X. (2004) Quasi-variances. *Biometrika* **91**, 65–80.

Menezes, R. X. de (1999) More useful standard errors for group and factor effects in generalized linear models. *D.Phil. Thesis*, Department of Statistics, University of Oxford.

Ridout, M.S. (1989). Summarizing the results of fitting generalized linear models to data from designed experiments. In: *Statistical Modelling: Proceedings of GLIM89 and the 4th International Workshop on Statistical Modelling held in Trento, Italy, July 17–21, 1989* (A. Decarli et al., eds.), pp 262–269. New York: Springer.

See Also

`qvcalc::worstErrors()`, `qvcalc::plot.qv()`.

Examples

```
example(baseball)
baseball.qv <- qvcalc(BTabilities(baseballModel2))
print(baseball.qv)
plot(baseball.qv, xlab = "team",
     levelNames = c("Bal", "Bos", "Cle", "Det", "Mil", "NY", "Tor"))
```

---

residuals.BTm *Residuals from a Bradley-Terry Model*

Description

Computes residuals from a model object of class `"BTm"`. In additional to the usual options for objects inheriting from class `"glm"`, a `"grouped"` option is implemented to compute player-specific residuals suitable for diagnostic checking of a predictor involving player-level covariates.

Usage

```
## S3 method for class 'BTm'
residuals(
  object,
  type = c("deviance", "pearson", "working", "response", "partial", "grouped"),
  by = object$id,
  ...
)
```

Arguments

`object`	a model object for which `inherits(model, "BTm")` is `TRUE`.
`type`	the type of residuals which should be returned. The alternatives are: `"deviance"` (default), `"pearson"`, `"working"`, `"response"`, and `"partial"`.
`by`	the grouping factor to use when `type = "grouped"`.
`...`	arguments to pass on other methods.

Details

For type other than "grouped" see residuals.glm().

For type = "grouped" the residuals returned are weighted means of working residuals, with weights equal to the binomial denominators in the fitted model. These are suitable for diagnostic model checking, for example plotting against candidate predictors.

Value

A numeric vector of length equal to the number of players, with a "weights" attribute.

Author(s)

David Firth and Heather Turner

References

Firth, D. (2005) Bradley-Terry models in R. *Journal of Statistical Software* **12**(1), 1–12.

Turner, H. and Firth, D. (2012) Bradley-Terry models in R: The BradleyTerry2 package. *Journal of Statistical Software*, **48**(9), 1–21.

See Also

BTm(), BTabilities()

Examples

```
##
##  See ?springall
##
springall.model <- BTm(cbind(win.adj, loss.adj),
                       col, row,
                       ~ flav[..] + gel[..] +
                       flav.2[..] + gel.2[..] + flav.gel[..] + (1 | ..),
                       data = springall)
res <- residuals(springall.model, type = "grouped")
with(springall$predictors, plot(flav, res))
with(springall$predictors, plot(gel, res))
##  Weighted least-squares regression of these residuals on any variable
##  already included in the model yields slope coefficient zero:
lm(res ~ flav, weights = attr(res, "weights"),
   data = springall$predictors)
lm(res ~ gel, weights = attr(res, "weights"),
   data = springall$predictors)
```

seeds *Seed Germination Data from Crowder (1978)*

Description

Data from Crowder(1978) giving the proportion of seeds germinated for 21 plates that were arranged according to a 2x2 factorial layout by seed variety and type of root extract.

Usage

```
seeds
```

Format

A data frame with 21 observations on the following 4 variables.

r the number of germinated seeds.

n the total number of seeds.

seed the seed variety.

extract the type of root extract.

Source

Crowder, M. (1978) Beta-Binomial ANOVA for proportions. *Applied Statistics*, **27**, 34–37.

References

Breslow, N. E. and Clayton, D. G. (1993) Approximate inference in Generalized Linear Mixed Models. *Journal of the American Statistical Association*, **88**(421), 9–25.

See Also

glmmPQL()

Examples

```
summary(glmmPQL(cbind(r, n - r) ~ seed + extract,
                random = diag(nrow(seeds)),
                family = binomial,
                data = seeds))
```

`sound.fields` *Kousgaard (1984) Data on Pair Comparisons of Sound Fields*

Description

The results of a series of factorial subjective room acoustic experiments carried out at the Technical University of Denmark by A C Gade.

Usage

```
sound.fields
```

Format

A list containing two data frames, `sound.fields$comparisons`, and `sound.fields$design`.

The `sound.fields$comparisons` data frame has 84 observations on the following 8 variables:

field1 a factor with levels `c("000", "001", "010", "011", "100", "101", "110", "111")`, the first sound field in a comparison

field2 a factor with the same levels as `field1`; the second sound field in a comparison

win1 integer, the number of times that `field1` was preferred to `field2`

tie integer, the number of times that no preference was expressed when comparing `field1` and `field2`

win2 integer, the number of times that `field2` was preferred to `field1`

win1.adj numeric, equal to `win1 + tie/2`

win2.adj numeric, equal to `win2 + tie/2`

instrument a factor with 3 levels, `c("cello", "flute", "violin")`

The `sound.fields$design` data frame has 8 observations (one for each of the sound fields compared in the experiment) on the following 3 variables:

a'') a factor with levels `c("0", "1")`, the *direct sound* factor (0 for *obstructed sight line*, 1 for *free sight line*); contrasts are sum contrasts

b a factor with levels `c("0", "1")`, the *reflection* factor (0 for *-26dB*, 1 for *-20dB*); contrasts are sum contrasts

c a factor with levels `c("0", "1")`, the *reverberation* factor (0 for *-24dB*, 1 for *-20dB*); contrasts are sum contrasts

Details

The variables `win1.adj` and `win2.adj` are provided in order to allow a simple way of handling ties (in which a tie counts as half a win and half a loss), which is slightly different numerically from the Davidson (1970) method that is used by Kousgaard (1984): see the examples.

Author(s)

David Firth

Source

Kousgaard, N. (1984) Analysis of a Sound Field Experiment by a Model for Paired Comparisons with Explanatory Variables. *Scandinavian Journal of Statistics* **11**, 51–57.

References

Davidson, R. R. (1970) Extending the Bradley-Terry model to accommodate ties in paired comparison experiments. *Journal of the American Statistical Association* **65**, 317–328.

Examples

```
##
##  Fit the Bradley-Terry model to data for flutes, using the simple
##  'add 0.5' method to handle ties:
##
flutes.model <- BTm(cbind(win1.adj, win2.adj), field1, field2, ~ field,
                    id = "field",
                    subset = (instrument == "flute"),
                    data = sound.fields)
##
##  This agrees (after re-scaling) quite closely with the estimates given
##  in Table 3 of Kousgaard (1984):
##
table3.flutes <- c(-0.581, -1.039, 0.347, 0.205, 0.276, 0.347, 0.311, 0.135)
plot(c(0, coef(flutes.model)), table3.flutes)
abline(lm(table3.flutes ~ c(0, coef(flutes.model))))
##
##  Now re-parameterise that model in terms of the factorial effects, as
##  in Table 5 of Kousgaard (1984):
##
flutes.model.reparam <- update(flutes.model,
                               formula = ~ a[field] * b[field] * c[field]
       )
table5.flutes <- c(.267, .250, -.088, -.294, .062, .009, -0.070)
plot(coef(flutes.model.reparam), table5.flutes)
abline(lm(table5.flutes ~ coef(flutes.model.reparam)))
```

`springall` *Springall (1973) Data on Subjective Evaluation of Flavour Strength*

Description

Data from Section 7 of the paper by Springall (1973) on Bradley-Terry response surface modelling. An experiment to assess the effects of gel and flavour concentrations on the subjective assessment of flavour strength by pair comparisons.

Usage

```
springall
```

Format

A list containing two data frames, `springall$contests` and `springall$predictors`.

The `springall$contests` data frame has 36 observations (one for each possible pairwise comparison of the 9 treatments) on the following 7 variables:

row a factor with levels `1:9`, the row number in Springall's dataset#

col a factor with levels `1:9`, the column number in Springall's dataset

win integer, the number of wins for column treatment over row treatment

loss integer, the number of wins for row treatment over column treatment

tie integer, the number of ties between row and column treatments

win.adj numeric, equal to `win + tie/2`

loss.adj numeric, equal to `loss + tie/2`

The `predictors` data frame has 9 observations (one for each treatment) on the following 5 variables:

flav numeric, the flavour concentration

gel numeric, the gel concentration

flav.2 numeric, equal to `flav^2`

gel.2 numeric, equal to `gel^2`

flav.gel numeric, equal to `flav * gel`

Details

The variables `win.adj` and `loss.adj` are provided in order to allow a simple way of handling ties (in which a tie counts as half a win and half a loss), which is slightly different numerically from the Rao and Kupper (1967) model that Springall (1973) uses.

Author(s)

David Firth

Source

Springall, A (1973) Response surface fitting using a generalization of the Bradley-Terry paired comparison method. *Applied Statistics* **22**, 59–68.

References

Rao, P. V. and Kupper, L. L. (1967) Ties in paired-comparison experiments: a generalization of the Bradley-Terry model. *Journal of the American Statistical Association*, **63**, 194–204.

Examples

```
##
## Fit the same response-surface model as in section 7 of
## Springall (1973).
##
## Differences from Springall's fit are minor, arising from the
## different treatment of ties.
##
## Springall's model in the paper does not include the random effect.
## In this instance, however, that makes no difference: the random-effect
## variance is estimated as zero.
##
summary(springall.model <- BTm(cbind(win.adj, loss.adj), col, row,
                               ~ flav[..] + gel[..] +
                                 flav.2[..] + gel.2[..] + flav.gel[..] +
                                 (1 | ..),
                               data = springall))
```

Index