Package 'ordBTL'

February 20, 2015

Type Package

Title Modelling comparison data with ordinal response

Version 0.8

Date 2014-05-07

Author Giuseppe Casalicchio

Maintainer Giuseppe Casalicchio <giuseppe.casalicchio@campus.lmu.de>

Description This package extends the Bradley-Terry-Luce model for fitting pair comparison models with an ordinal response. It is also possible to incorporate an order effect, or, equivalently, an effect for the home advantage.

License GPL-3

Depends caret, VGAM, wikibooks, gtools

Suggests BradleyTerry2, prefmod, psychotree, psychotools, psychomix

LazyData TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2014-05-28 18:39:18

R topics documented:

BTLboost *Boosting ordinal Bradley-Terry-Luce models*

Description

Boosting procedure for ordinal BTL models

Usage

```
BTLboost(formula, data, objects=NULL, groupVars=NULL,
                selection=c("DEVIANCE","AIC","BIC"),
                mstop=500, nu=1, maxit=1, verbose=TRUE, ...)
```

Arguments

formula	a formula describing the full model.
data	a data frame containing the design matrix for the model (See also design to generate such an design matrix).
objects	(optional) a character vector specifying the objects that should always be part of the model.
groupVars	(optional) a character vector specifying the subject-specific covariates, whose subject-object interactions are considered simultaneously in each boosting step. By default (groupVars=NULL), all subject-object interactions are cosidered separately in each boosting step.
selection	a character specifying the criterion that is used in each boosting step to determine the best fitting covariate(s).
mstop	an integer giving the number of boosting iterations.
nu	a double between 0 and 1 defining the step size or shrinkage parameter.
maxit	an integer representing the maximum number of Fisher-scoring iterations (see also vglm.control).
verbose	logical indicating if output should be produced for each boosting iteration.
...	further arguments passed to ordBTL.

Value

A List of

- BEST contains estimated parameters of the last boosting iteration
- AIC a vector of AIC values for each boosting iteration
- BIC a vector of BIC values for each boosting iteration
- DEVIANCE a vector that reflects the deviance of each boosting iteration
- PATH a dataframe containing the coefficient build-up at the end of each boosting iteration
- UPDATED a vector of strings containing the selected components in each boosting iteration

Author(s)

Giuseppe Casalicchio

Examples

```
# Get the CEMS data and generate design matrix
example(wide2long, package="ordBTL", echo=FALSE)

des2 <- design(CEMSlong[-which(is.na(CEMSlong$Y)),],
               var1="object1", var2="object2",
               use.vars="ALL", reference="Stockholm")

# Formula for full model considering all subject-object interactions
form2 <- Y ~
  (GAMMA.London+GAMMA.Paris+GAMMA.Milano+GAMMA.StGallen+GAMMA.Barcelona)+
  (GAMMA.London+GAMMA.Paris+GAMMA.Milano+GAMMA.StGallen+GAMMA.Barcelona):
  (WOR+SEX+DEG+STUD+ENG+FRA+SPA+ITA)

## Not run:
# Exemplatory boosting call with mstop=5
#BoostDev <- BTLboost(form2, data=des2, groupVars=c("WOR","DEG","SEX","STUD"),
#                     selection="DEVIANCE", mstop=5)
## End(Not run)
```

`CEMSwide` *CEMS Data*

Description

The Community of European management schools (CEMS) data used in the paper by Dittrich et al. (1998, 2001), where the responses (the first 15 variables) are coded as 1 if the first university was preferred, 2 if no decision was made and 3 if the second university was preferred.

Usage

```
data(CEMSwide)
```

Format

A data frame with 303 observations on the following 23 variables:

- `V1` London vs. Paris
- `V2` London vs. Milano
- `V3` London vs. Paris
- `V3` Paris vs. Milano
- `V4` London vs. St.Gallen
- `V5` Paris vs. St.Gallen

- V6 Milano vs. St.Gallen
- V7 London vs. Barcelona
- V8 Paris vs. Barcelona
- V9 Milano vs. Barcelona
- V10 St.Gallen vs. Barcelona
- V11 London vs. Stockholm
- V12 Paris vs. Stockholm
- V13 Milano vs. Stockholm
- V14 St.Gallen vs. Stockholm
- V15 Barcelona vs. Stockholm
- STUD Main discipline of study : 1= commerce, 0= other (economics, business administration, business education)
- ENG Knowledge of English : 0= good, 1= poor
- FRA Knowledge of French : 0= good, 1= poor
- SPA Knowledge of Spanish : 0= good, 1= poor
- ITA Knowledge of Italian : 0= good, 1= poor
- WOR Full-time employment while studying: 0= no, 1= yes
- DEG Intention to take an international degree: 0= no, 1= yes
- SEX Gender : 0= female, 1= male

Source

The Royal Statistical Society Datasets Website (http://www.blackwellpublishing.com/rss/Readmefiles/dittrich.htm)

References

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*(2), pp. 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." Journal of the Royal Statistical Society: Series C (Applied Statistics), *50*(2), pp. 247-249.

design *Design Matrix*

Description

This function returns the design matrix for an ordinal Bradley-Terry-Luce model.

Usage

```
design(X, var1, var2, use.vars=NULL, reference=NULL,
              prefix="GAMMA", prefix.home="ALPHA",
              home.advantage=c("no","specific","yes"))
```

Arguments

X	a data frame in long format (see wide2long).
var1	a character of the column name from X specifying the first object to be compared (or, in sport context, the home team).
var2	a character of the column name from X specifying the second object to be compared (or, in sport context, the away team).
use.vars	a character vector with the names of additional covariates of X that will additionally be included into the design matrix. (example: use.vars = c("ENG", "SEX")) if the covariates ENG and SEX should be included. If all covariates of X should be included, you can use use.vars = "ALL". The default is use.vars = NULL for no additional covariates.
reference	a character specifying the reference object.
prefix	(optional) a character added in the names of the estimated object parameters
prefix.home	(optional) a character added in the names of the estimated home advantage parameters
home.advantage	Note that the home advantage is equivalent to an order effect home.advantage="no" uses no home advantage (order effect), home.advantage="specific" uses one home advantage (order effect) for each object and home.advantage="yes" uses one home advantage (order effect) for any object.

Value

A data frame where each row refers to a pair comparison and each column corresponds to an object.

Author(s)

Giuseppe Casalicchio

Examples

```
# load german football-league (Bundesliga) data
library("wikibooks")
data(Bundesliga)

# add new variable Y3 reflecting the response which is coded as
# 1 if the home team wins
# 2 if the game ends up with a tie
# 3 if the home team loses
diff <- Bundesliga$Tore.Heim - Bundesliga$Tore.Gast
Bundesliga$Y3 <- as.ordered(ifelse(diff >= 1, 1,
                                   ifelse(diff <= -1, 3, 2)))
buli0506 <- subset(Bundesliga, Saison=="2005/2006")
str(buli0506)

# Design matrix without home advantage
des.nohome <- design(buli0506, var1="Heim", var2="Gast",
                     home.advantage="no")
str(des.nohome)

# Design matrix with one home advantage parameter for all objects
des.onehome <- design(buli0506, var1="Heim", var2="Gast",
                      home.advantage="yes")
str(des.onehome)

# Design matrix with home advantage parameters for each object
des.teamhome <- design(buli0506, var1="Heim", var2="Gast",
                       home.advantage="specific")
str(des.teamhome)

# Design matrix with additional covariable "Spieltag"
des.covs <- design(buli0506, var1="Heim", var2="Gast",
                   use.vars=c("Spieltag"), home.advantage="no")
str(des.covs)
```

---

getRank *Ranking based on Estimates*

Description

Extracts the estimated parameters and sorts them based on their estimated values

Usage

```
getRank(ordBTL, decreasing=TRUE, prefix=NULL, reference=NULL)
```

Arguments

ordBTL	a fitted model returned by ordBTL.
decreasing	logical. Should the sort be increasing or decreasing?
prefix	(optional) a character that is included in the names of the parameters; only the parameters are returned that include this character (prefix=NULL extracts all estimated parameters).
reference	(optional) a character specifying the reference object.

Value

matrix containing the parameter estimates.

Author(s)

Giuseppe Casalicchio

See Also

ordBTL

Examples

```
# Get the CEMS data and generate design matrix
example(wide2long, package="ordBTL", echo=FALSE)
des1 <- design(CEMSlong, var1="object1", var2="object2",
               use.vars="Y", reference="Stockholm")

# Fit the adjacent categories model, which corresponds to
# the log-linear BTL model (see Agresti, 1992)
mod1 <- ordBTL(Y~., data=des1, family="acat",
               family.control=list(reverse=TRUE))

# Extract all parameter estimates
getRank(mod1)

# Extract all parameter estimates and add parameter for
# reference object (which is set to zero)
getRank(mod1, reference="GAMMA.Stockholm")

# Extract only parameter estimates that include the
# string "Intercept"
getRank(mod1, prefix="Intercept")

# Extract only parameter estimates that include the
# string "GAMMA" (which will be the object parameters)
getRank(mod1, prefix="GAMMA")
```

ordBTL *ordinal Bradley-Terry-Luce model (ordBTL)*

Description

Fits ordinal regression models to paired comparison data.

Usage

```
ordBTL(formula, data, family=c("cumulative","acat"),
              family.control = list(), restrict=NULL, ...)
```

Arguments

formula	a formula describing the model to be fitted.
data	a data frame containing the design matrix for the model (See also design to generate such an design matrix).
family	a character specifying which ordinal BTL model should be fitted. Can be either "cumulative" for the cumulative link model or "acat" for the adjacent categories model.
family.control	a list with arguments passed to the corresponding family, either cumulative for the cumulative link model or acat for the adjacent categories model.
restrict	(optional) a character vector specifying the covariates from formula that should be fitted with a symmetry constraint (can be used to fit threshold covariates).
...	further arguments for fitting function (see vglm).

Value

An object of class vglm.

Author(s)

Giuseppe Casalicchio

References

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." Journal of the Royal Statistical Society: Series C (Applied Statistics), *50*(2), pp. 247-249.

Agresti A (1992). "Analysis of ordinal paired comparison data." Applied Statistics, pp. 287-297. Table 1.

See Also

vglm, design, plotvgam

Examples

```
############################################################
##                                                        ##
## Example 1: Adjacent categories logit model for CEMS    ##
##                                                        ##
############################################################

############################################################
# Reproduce results from Table 3 of Dittrich et al. (2001)
############################################################

# Get the CEMS data and generate design matrix
example(wide2long, package="ordBTL", echo=FALSE)
des1 <- design(CEMSlong, var1="object1", var2="object2",
               use.vars="Y", reference="Stockholm")

# Fit the adjacent categories model, which corresponds to
# the log-linear BTL model (see Agresti, 1992)
mod1 <- ordBTL(Y~., data=des1, family="acat",
                family.control=list(reverse=TRUE))

# We get the same results from Table 3 of Dittrich et al (2001).
# Since Stockholm is the reference university, its estimate
# is set to zero (due to identifiability)
getRank(mod1)

############################################################
# Reproduce results from Table 6 of Dittrich et al. (2001)
############################################################

# Generate design matrix and specify model formula
des2 <- design(CEMSlong, var1="object1", var2="object2",
                use.vars="ALL", reference="Stockholm")
form2 <- Y~GAMMA.London + GAMMA.Paris + GAMMA.Milano +
  GAMMA.StGallen + GAMMA.Barcelona + WOR +
  SEX + WOR:GAMMA.Paris + WOR:GAMMA.Milano +
  WOR:GAMMA.Barcelona + DEG:GAMMA.StGallen +
  STUD:GAMMA.Paris + STUD:GAMMA.StGallen +
  ENG:GAMMA.StGallen + FRA:GAMMA.London +
  FRA:GAMMA.Paris + SPA:GAMMA.Barcelona +
  ITA:GAMMA.London + ITA:GAMMA.Milano +
  SEX:GAMMA.Milano

# Fit the adjacent categories model with symmetric
# constraint for covariable WOR and SEX
mod2 <- ordBTL(form2, data=des2, family="acat",
                family.control=list(reverse=TRUE),
                restrict=c("WOR", "SEX"))

# We get the same results from Table 6 of Dittrich et al. (2001)
getRank(mod2)
```

```
# Notice that the change in sign for (Intercept), WOR and SEX
# is because we use here a different "coding".

############################################################
##                                                        ##
## Example 2: Fitting models from Agresti (1992)          ##
##                                                        ##
############################################################

# Data from Table 1 of Agresti (1992)
data(ribbon)

# design matrix
des3 <- design(ribbon, var1="obj1", var2="obj2", use.vars="ALL")

# Note that Agresti (1992) used the constraint that the object
# parameters sum up to 1. To get the same results, we use the model
form3 <- cbind(V1,V2,V3,V4,V5,V6,V7)~I(GAMMA.1-GAMMA.5)+
  I(GAMMA.2-GAMMA.5)+I(GAMMA.3-GAMMA.5)+I(GAMMA.4-GAMMA.5)

# Fit the adjacent categories logit model
ac <- ordBTL(form3, data=des3, family="acat",
             family.control=list(reverse=TRUE))

# Fit the cumulative logit model
clm.logit <- ordBTL(form3, data=des3)

# Fit the cumulative probit model
clm.probit <- ordBTL(form3, data=des3,
                     family.control=list(link="probit"))

# Parameter estimates
coefs <- t(rbind("Adjacent categories logit"=coefficients(ac),
                 "Cumulative probit"=coefficients(clm.probit),
                 "Cumulative logit"=coefficients(clm.logit)))
coefs <- rbind(coefs, "GAMMA.5"=0-colSums(coefs[4:7,]))
coefs


############################################################
##                                                        ##
## Example 3: Fitting models for Bundesliga 2005/2006     ##
##                                                        ##
############################################################
# real ranking can be obtained from:
# http://fussballdaten.sport.de/bundesliga/2006

# load data
example(design, package="ordBTL", echo=FALSE)

# Model without home advantage
des.nohome <- design(buli0506, var1="Heim", var2="Gast",
                     use.vars="Y3", home.advantage="no",
```

```
                    reference="GAMMA.MSV.Duisburg")
mod.nohome <- ordBTL(Y3~., data=des.nohome)
# team 'abilities' (should be approximately the ranking of the final standings)
getRank(mod.nohome, prefix="GAMMA", reference="GAMMA.MSV.Duisburg")

# Model with home advantage
des.onehome <- design(buli0506, var1="Heim", var2="Gast",
                      use.vars="Y3", home.advantage="yes",
                      reference="GAMMA.MSV.Duisburg")
mod.onehome <- ordBTL(Y3~., data=des.onehome)
# team 'abilities'
getRank(mod.onehome, prefix="GAMMA", reference="GAMMA.MSV.Duisburg")
# home advantage
getRank(mod.onehome, prefix="ALPHA")

# Model with team-specific home advantage
des.teamhome <- design(buli0506, var1="Heim", var2="Gast",
                      use.vars="Y3", home.advantage="specific",
                      reference="GAMMA.MSV.Duisburg")
mod.teamhome <- ordBTL(Y3~., data=des.teamhome)
# team 'abilities' (should be approximately the ranking for the away table)
getRank(mod.teamhome, prefix="GAMMA", reference="GAMMA.MSV.Duisburg")
# team-specific home advantages
getRank(mod.teamhome, prefix="ALPHA")
```

`ribbon` *Typewriter Ribbon Data*

Description

The typwriter ribbon data used in the paper by Agresti (1992), where 5 different typewriter ribbon brands were compared pairwise by 30 secretaries.

Usage

```
data(ribbon)
```

Format

A data frame with columns:

- `V1` absolute frequency reflecting strong preference for corresponding `obj1`
- `V2` absolute frequency reflecting moderate preference for corresponding `obj1`
- `V3` absolute frequency reflecting weak preference for corresponding `obj1`
- `V4` absolute frequency reflecting no preference
- `V5` absolute frequency reflecting weak preference for corresponding `obj2`
- `V6` absolute frequency reflecting moderate preference for corresponding `obj2`
- `V7` absolute frequency reflecting strong preference for corresponding `obj2`

- `obj1` integer 1, ..., 5 referring to the 5 ribbon brands
- `obj2` integer 1, ..., 5 referring to the 5 ribbon brands

Source

Agresti A (1992). "Analysis of ordinal paired comparison data." Applied Statistics, pp. 287-297. Table 1.

References

Agresti A (1992). "Analysis of ordinal paired comparison data." Applied Statistics, pp. 287-297.

`wide2long` *Reshapes Paired Comparison Data*

Description

This function reshapes a data frame that contains pair comparison data in 'wide' format into a data frame in 'long' format (see 'Details').

Usage

```
wide2long(data, paircomp, names=NULL, ...)
```

Arguments

`data`	a data frame in 'wide' format.
`paircomp`	a character vector specifying the columns from `data` that corresponds to a pair comparison.
`names`	a character vector of the same length as `paircomp` specifying the names (separated with a space) of the two objects that are compared in a pair comparison.
`...`	arguments to be passed to `reshape`.

Details

In the 'wide' format each row reflects a certain subject/judge and a column contains for example the results of a pair comparison or is a subject-specific covariate. In the 'long' format each row represents a single pair comparison.

Value

The reshaped data frame.

Author(s)

Giuseppe Casalicchio

See Also

reshape, design

Examples

```
data(CEMSwide)

# Define variable names containing the outcome of the comparisons
columns <- colnames(CEMSwide)[grep("V",colnames(CEMSwide))]

# Define names of the objects that are compared (shoud be separated with space)
# "London Paris" means that the first object is "London" and the second "Paris".
comparison <- c("London Paris", "London Milano", "Paris Milano",
                "London St.Gallen", "Paris St.Gallen",
                "Milano St.Gallen", "London Barcelona",
                "Paris Barcelona", "Milano Barcelona",
                "St.Gallen Barcelona", "London Stockholm",
                "Paris Stockholm", "Milano Stockholm",
                "St.Gallen Stockholm", "Barcelona Stockholm")

# Transform to 'long' format where v.names="Y" sets the name of
# our response variable (see ?reshape)
CEMSlong <- wide2long(data=CEMSwide, paircomp=columns, names=comparison, v.names="Y")

head(CEMSlong)
head(CEMSwide)
```

Index