Package: prefmod (via r-universe)

October 29, 2026

Type Package

Title Utilities to Fit Paired Comparison Models for Preferences

Version 0.8-37

Date 2025-08-18

Depends R (>= 3.0.0), stats, graphics, gnm (>= 1.0-0), colorspace

Imports grDevices, utils, methods

Description Generates design matrix for analysing real paired comparisons and derived paired comparison data (Likert type items/ratings or rankings) using a loglinear approach. Fits loglinear Bradley-Terry model (LLBT) exploiting an eliminate feature. Computes pattern models for paired comparisons, rankings, and ratings. Some treatment of missing values (MCAR and MNAR). Fits latent class (mixture) models for paired comparison, rating and ranking patterns using a non-parametric ML approach.

License GPL (>= 2)

LazyData yes

LazyLoad yes

ByteCompile yes

Encoding UTF-8

NeedsCompilation yes

Author Reinhold Hatzinger [aut], Marco Johannes Maier [cre]

Maintainer Marco Johannes Maier <marco_maier@posteo.de>

Repository https://cran.r-universe.dev

Date/Publication 2025-08-18 14:50:07 UTC

RemoteUrl https://github.com/cran/prefmod

RemoteRef HEAD

RemoteSha f14a5e7599a4abadf90ab75ed55ac4ab66ec8724

Contents

prefmod-package . . . 2
baseball . . . 3
carconf . . . 4
cemspc . . . 5
checkMIS . . . 7
dat4 . . . 8
euro55.2.des . . . 8
expand.mat . . . 9
immig . . . 10
issp2000 . . . 11
llbt.design . . . 12
llbt.fit . . . 16
llbt.worth . . . 17
llbtPC.fit . . . 19
music . . . 21
patt.design . . . 22
patt.worth . . . 26
pattL.fit . . . 27
pattLrep.fit . . . 29
pattnpml.fit . . . 32
pattPC.fit . . . 36
pattR.fit . . . 39
pattRrep.fit . . . 41
plot.wmat . . . 44
prefmod-defunct . . . 45
print.pattMod . . . 46
salad . . . 47
simPC . . . 48
summary.pattNPML . . . 49
tennis . . . 49
trdel . . . 50
xmpl . . . 51

Index **52**

`prefmod-package` **prefmod**: *Utilities to Fit Paired Comparison Models for Preferences*

Description

Generates design matrix for analysing real paired comparisons and derived paired comparison data (Likert-type items/ratings or rankings) using a loglinear approach. Fits loglinear Bradley-Terry model (LLBT) exploiting an eliminate feature. Computes pattern models for paired comparisons, rankings, and ratings. Some treatment of missing values (MCAR and MNAR). Fits pattern mixture models using a non-parametric ML approach.

Details

Package:	prefmod
Type:	Package
Version:	0.8-37
Date:	2025-08-18
License:	GPL (>= 2)

Author(s)

Reinhold Hatzinger, Marco J. Maier

Maintainer: Marco J. Maier (<marco_maier@posteo.de>)

References

Hatzinger, R., & Dittrich, R. (2012). **prefmod**: An R Package for Modeling Preferences Based on Paired Comparisons, Rankings, or Ratings. *Journal of Statistical Software, 48*(10), 1–31. https://www.jstatsoft.org/v48/i10/

Examples

```
# mini example with three Likert items and two subject covariates

# using example data "xmpl" in the package
dsgnmat <- patt.design(xmpl, nitems = 3, resptype = "rating",
    ia = TRUE, cov.sel = "ALL")
head(dsgnmat)

# fit of Critchlov & Fligner (1991) Salad Dressings Data
pattR.fit(salad, nitems = 4)

# alternatively use glm() with patt.design()
sal <- patt.design(salad, nitems = 4, resptype = "ranking")
glm(y ~ A+B+C+D, data = sal, family = poisson)
```

---

baseball *Data (paired comparisons): Baseball Games*

Description

The result of the 1987 season for seven baseball teams in the Eastern Division of the American League according to the (home team, away team) classification are shown.

Usage

```
baseball
```

Format

Baseball is a numeric vector with the results for the season according to the (home team, away team) classification.

Details

The results of the seven teams Milwaukee, Detroit, Toronto, New York, Boston, Cleveland and Baltimore, that play 13 games each. There is no possibility of ending in a draw.

References

Alan Agresti, *Categorical Data Analysis (Second Edition)*, 2002 pages 437 and 438

R. Dittrich, R. Hatzinger, and W. Katzenbeisser, Fitting paired comparison models in GLIM. *GLIM newsletter* 1997

Examples

```
# baseball example (Agresti, 2002, p. 437)

# pseudodata for generating a design matrix
d1 <- c(rep(0, 21), 1)
d2 <- c(1, rep(0, 20), 2)
d  <- data.frame(rbind(d1, d2))
names(d) <- c(paste0("v", 1:21), "cov")

# design matrix
des5 <- llbt.design(d, nitems = 7,
    objnames = c("MIL", "DET", "TOR", "NY", "BOS", "CLE", "BAL"),
    cat.scov = "cov")
des5$y  <- baseball
des5$mu <- gl(42, 2)
pos     <- c(rep(1:0, 21), rep(0:1, 21))

# fit model and display results
res5 <- gnm(y ~ MIL+DET+TOR+NY+BOS+CLE+BAL + pos,
    eliminate = mu, data = des5, family = poisson)
w5   <- llbt.worth(res5)
plot(w5)
```

---

carconf *Data (partial rankings): Car Configurator*

Description

Online configuration systems allow customers to actively participate in the creation of products and become increasingly important in various market sectors. Dabic and Hatzinger report a study on car configurators that aimed at investigating the effects of certain person characteristics (such as gender) on the configuration process. Subjects were asked to configure a car according to their

preferences. They could choose freely from several modules: such as exterior and interior design, technical equipment, brand, price, and producing country. The order of module choice was recorded as ranks. Since not all modules had to be chosen the response format was partial rankings.

Usage

```
carconf
```

Format

A data frame with 435 observations on the following 9 variables.

`price` rank (1 highest preference)

`exterior` rank

`brand` rank

`tech.equip` rank

`country` rank

`interior` rank

`sex` 1 female 2 male

`age` 1 17–29 years, 2 30–49 years, 3 50+ years

`segment` preferred car type: 1 premium-class, 2 medium-class, 3 low-budget

Source

Dabic, M., & Hatzinger, R. (2009). Zielgruppenadäquate Abläufe in Konfigurationssystemen – eine empirische Studie im Automobilmarkt: Das Paarvergleichs-Pattern-Modell für Partial Rankings. In R. Hatzinger, R. Dittrich, & T. Salzberger (Eds.), *Präferenzanalyse mit R: Anwendungen aus Marketing, Behavioural Finance und Human Resource Management* (pp. 119–150). Wien: Facultas.

Examples

```
head(carconf)
```

---

cemspc *Data (paired comparisons with undecided): CEMS (Community of European management schools)*

Description

A survey of 303 students was carried out to examine the students' preferences of 6 universities (London, Paris, Milano, St. Gallen, Barcelona and Stockholm) with a 17 items questionnaire. The first 15 variables indicate the subjects' preferences. For a given comparison the responses are coded as 0 if the first university was preferred, 2 if the second university was preferred and 1 if no decision was made. The variable `ENG` contains the knowledge of English and the variable `SEX` contains the gender.

Usage

```
cemspc
```

Format

A data frame with 303 rows for the subjects containing the outcome of the 15 comparisons and the two covariates

`V1` London vs. Paris

`V2` London vs. Milano

`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

`ENG` Knowledge of English: (1) good, (2) poor

`SEX` Gender: (1) female, (2) male

References

Dittrich, R., Hatzinger, R., & Katzenbeisser, W. (1998). Modelling the effect of subject-specific covariates in paired comparison studies with an application to university rankings. *Applied Statistics, 47*(4), 511–525.

Examples

```
old_par <- par(mfrow = c(4, 4))
for(i in 1:15){ barplot(table(cemspc[, i])) }
par(old_par)
```

checkMIS *Function to check/report missing values in paired comparison studies*

Description

For a given paired comparisons data set the function calculates and prints the number of missing comparisons and the number of times objects are missing. It can also be used to avoid failure of nonresponse-parameter for nonresponse models in

Usage

```
checkMIS(obj, nitems, MISmodel = "obj", obj.names = NULL, verbose = FALSE)
```

Arguments

obj	dataframe or datafile path/name (like pattPC.fit).
nitems	the number of compared objects, not the number of comparisons (like pattPC.fit).
MISmodel	specifies the nonresponse model, either obj (default) for missing mechanisms based on objects, or comp based on comparisons. This argument is only relevant for the (invisible) output of checkMIS.
obj.names	character vector with names for objects.
verbose	if TRUE printed output, otherwise only invisible output to be used, e.g., in the specification of MISalpha and MISbeta in pattPC.fit.

Value

a logical vector (returned invisibly) specifying for which object/comparison there are NA responses in the data (obj).

See Also

pattPC.fit

Examples

```
# no missing NAs in dataset dat4
checkMIS(dat4, nitems = 4, verbose = TRUE)

# generates data set with three items and some missing values in
# comparison (23), column 3, then there are no NAs for object 1
data3 <- dat4[, 1:3]
idx3  <- sample(1:100, 10)
data3[idx3, 3] <- NA
checkMIS(data3, nitems = 3, verbose = TRUE)

# estimate MCAR PC pattern model for data3 with NA indicators alpha1
# cannot be estimated being accommodated by using checkMIS
pattPC.fit(data3, nitems = 3, MISalpha = checkMIS(data3, nitems = 3))
```

dat4 — *Data (paired comparisons): dat4*

Description

A fictitious dataset with 100 observations on 6 paired comparisons. The responses get the value 1 if the first object in a comparison is preferred and -1 otherwise.

For the arrangement of objects and comparisons see `llbt.design`.

Usage

```
dat4
```

Format

A data frame with 100 observations on 6 comparisons (`comp1` to `comp6`)

Examples

```
str(dat4)

# to get a general idea we use the histogram plot
old_par <- par(mfrow = c(2, 3))
for(i in 1:6){ barplot(table(dat4[, i])) }
par(old_par)
```

euro55.2.des — *Design data frame for a paired comparison pattern model for rankings (Eurobarometer 55.2)*

Description

Eurobarometer public opinion surveys have been carried out in all member states of the European Union since 1973. Eurobarometer 55.2 was a special survey collected in 2001 and designed to elicit information on European experience and perception of science and technology. One question, of identical form in all states, asked respondents about their sources of information about science, and requested them to rank them in order of importance. The items were: Television, Radio, Press, Scientific Magazines, Internet, School and University.

Usage

```
euro55.2.des
```

Format

The data is a design data frame generated from the original data using `patt.design`. Each row represents a (derived) paired comparison response pattern crossclassified by the categorical subject covariates `SEX` and `AGE4`. The columns are:

`y` counts

`TV` design vector for Television

`RAD` design vector for Radio

`NEWSP` design vector for Newspapers and magazine

`SCIMAG` design vector for Scientific magazines

`WWW` design vector for The internet

`EDINST` design vector for School/University

`SEX` Gender: Factor with 2 levels: (1) male, (2) female

`AGE4` Age: Factor with 4 levels: (1) 15–24, (2) 25–39, (3) 40–54, (4) 55+

Details

The original data contained 16 029 cases, after removal of cases with missing values and improper rankings this design data frame is based on 12216 observations.

Source

https://data.europa.eu/data/datasets/s209_55_2_ebs154

References

Christensen, T. (2001). Eurobarometer 55.2: Europeans, science and technology. Technical report, European Opinion Research Group, Commission of the European Communities, Brussels.

Examples

```
str(euro55.2.des)
```

---

expand.mat *Utility function to expand aggregated data*

Description

The function expands aggregated data into casewise data. For instance, for a contingency table given in the form of a design matrix and corresponding counts the function sets up a matrix where each design row is repeated according to the frequencies for that row.

Usage

```
expand.mat(mat, freq)
```

Arguments

mat	a matrix (or column vector) or data frame to be expanded
freq	a vector of counts

Value

the expanded matrix.

Note

This utility allows to generate input data for the design generating and model fitting functions of the **prefmod** package from aggregated data.

Examples

```
tdat <- expand.mat(tennis[, -1], tennis[, 1])
head(tdat)
```

---

immig *Data (paired comparisons with undecided and forced* NA*s): Negative Attitudes towards Immigrants*

Description

A survey of 98 students was carried out to examine student's (negative) attitudes towards immigrants. Four statements had to be compared with regard to higher acceptance. The four statements were

- Foreigners increase crime rates (crimRate)
- Foreigners take apprenticeship training position away (position)
- Foreigners are a strain for the social welfare system (socBurd)
- Foreigners threaten our culture (culture)

The first 6 variables in the dataset indicate the preferences of the students. For a given comparison the responses are coded by 1 if the first item was preferred, -1 if the second university was preferred and 0 if no decision was made. The variable ENG characterises the knowledge of English and the variable SEX characterises the gender.

Usage

```
immig
```

Format

A data frame with 98 observations on the following 9 variables.

`V12` `crimRate` vs. `position`

`V13` `crimRate` vs. `socBurd`

`V23` `position` vs. `socBurd`

`V14` `crimRate` vs. `culture`

`V24` `position` vs. `culture`

`V34` `socBurd` vs. `culture`

`SEX` Gender: (1) male, (2) female

`AGE` Age (continuous)

`NAT` Nationality (Factor). Cannot directly be used in **prefmod**

References

Weber, D., & Hatzinger, R. (2011). A novel approach for modelling paired comparisons data with non-ignorable missing values on students' attitudes towards foreigners. *Data Analysis Bulletin, 12*, 11–22.

Examples

```
summary(immig)
```

---

issp2000 *Data (Likert items): ISSP 2000 Survey on Environmental Issues*

Description

In 2000 the International Social Survey Programme (ISSP) has addressed the topic of attitudes to environmental protection and preferred government measures for environmental protection. This dataset focusses on six items (with a 5-point rating scale (Likert type)) where respondents from Austria and Great Britain were asked about their perception of environmental dangers.

Usage

```
issp2000
```

Format

A data frame with 1595 observations on the following 11 variables. The first six variables are items to be answered on a 5-point rating scale (Likert type) with response categories: (1) *extremely dangerous for the environment* to (5) *not dangerous at all for the environment*.

`CAR` air pollution caused by cars

`IND` air pollution caused by industry

`FARM` pesticides and chemicals used in farming

`WATER` pollution of country's rivers, lakes and streams

`TEMP` a rise in the world's temperature

`GENE` modifying the genes of certain crops

`SEX` gender: (1) *male*, (2) *female*

`URB` location of residence: (1) *urban area*, (2) *suburbs of large cities, small town, county seat* (3) *rural area*

`AGE` age: (1) < *40 years*, (2) *41–59 years*, (3) *60+ years*

`CNTRY` country: (1) *Great Britain*, (2) *Austria*

`EDU` education: (1) *below A-level/matrice*, (2) *A-level/matrice or higher*

Source

ISSP Research Group (2003). *International Social Survey Programme: Environment II – ISSP 2000.* GESIS Data Archive, Cologne. ZA3440 Data file Version 1.0.0, doi:10.4232/1.3440 (Usage regulations)

References

Dittrich, R., Francis, B.J., Hatzinger R., Katzenbeisser, W. (2007). A Paired Comparison Approach for the Analysis of Sets of Likert Scale Responses. *Statistical Modelling*, Vol. 7, No. 1, 3–28.

Examples

```
str(issp2000)
```

`llbt.design` *Loglinear Bradley-Terry Model (LLBT) – Design Matrix Generation*

Description

The function `llbt.design` returns a data frame containing the design matrix for a loglinear paired comparison model. Additionally, the frequencies of the pairwise comparisons are computed and are stored in the first column of the data frame.

Usage

```
llbt.design(data, nitems = NULL, objnames = "", objcovs = NULL,
        cat.scovs = NULL, num.scovs = NULL, casewise = FALSE, ...)
```

Arguments

`data`	either a data frame or a data file name.
`nitems`	number of items (objects).
`objnames`	an optional character vector with names for the objects. These names are the columns names in the output data frame. If `objnames` is not specified `o1`, `o2`, etc. will be used.
`objcovs`	an optional data frame with object specific covariates. The rows correspond to the objects, the columns define the covariates. The column names of this data frame are later used to fit the covariates. Factors are not allowed. In that case dummy variables have to be set up manually (favourably using `model.matrix`).
`cat.scovs`	a character vector with the names of the categorical subject covariates in the data file to be included into the design matrix. (Example: `cat.scovs = c("SEX", "WORK")`.) If all covariates in the data are categorical and should be included, the specification can be abbreviated to `cat.scovs = "ALL"`. In that case, `num.scovs` must not be specified. For no categorical covariates: `cat.scovs = ""`, the default.
`num.scovs`	analogous to `cat.scovs` for numerical (continuous) subject covariates. If any numerical covariates are specified, `casewise` is set to `TRUE`.
`casewise`	If `casewise = TRUE` a separate design structure is set up for each subject in the data. This is required when fitting continuous subject covariates. However, the design can become very large in the case of many subjects and/or comparisons. See Details below.
`...`	deprecated options to allow for backwards compatibility (see Deprecated below)

Details

The function `llbt.design` allows for different scenarios mainly concerning

- **Paired comparison data.** Responses can be either simply *preferred – not preferred* or ordinal (*strongly preferred – . . . – not at all preferred*). In both cases an undecided category may or may not occur. If there are more than three categories a they are reduced to two or three response categories.
- **Item covariates.** The design matrix for the basic model has columns for the items (objects) and for each response category.
- **Object specific covariates.** For modelling certain characteristics of objects a reparameterisation can be included in the design. This is sometimes called conjoint analysis. The object specific covariates can be continuous or dummy variables. For the specification see Argument `objcovs` above.
- **Subject covariates.** For modelling different preference scales for the items according to characteristics of the respondents categorical and/or continuous subject covariates can be included in the design.
  Categorical subject covariates: The corresponding variables are defined as numerical vectors where the levels are specified with consecutive integers starting with 1. This format must be used in the input data file. These variables are `factor(s)` in the output data frame.
  Continuous subject covariates: also defined as numerical vectors in the input data frame. If present, the resulting design structure is automatically expanded, i.e., there are as many design blocks as there are subjects.

- **Object specific covariates.** The objects (items) can be reparameterised using an object specific design matrix. This allows for scenarios such as conjoint analysis or for modelling some characteristics shared by the objects. The number of such characteristics must not exceed the number of objects minus one.

Value

The output is a dataframe of class `llbtdes`. Each row represents a decision in a certain comparison. Dependent on the number of response categories, comparisons are made up of two or three rows in the design matrix. If subject covariates are specified, the design matrix is duplicated as many times as there are combinations of the levels of each categorical covariate or, if `casewise = TRUE`, as there are subjects in the data. Each individual design matrix consists of rows for all comparisons.

The first column contains the counts for the paired comparison response patterns and is labelled with `y`. The next columns are the covariates for the categories (labelled as `g0`, `g1`, etc.). In case of an odd number of categories, `g1` can be used to model an undecided effect. The subsequent columns are covariates for the items. If specified, subject covariates and then object specific covariates follow.

Input Data

Responses have to be coded as consecutive integers (e.g., (0, 1), or (1, 2, 3, ...), where the smallest value corresponds to (highest) preference for the first object in a comparison. For (ordinal) paired comparison data (`resptype = "paircomp"`) the codings $(1, -1)$, $(2, 1, -1, -2)$, $(1, 0, -1)$, $(2, 1, 0, -1, -2)$, etc. can also be used. Then negative numbers correspond to not preferred, 0 to undecided. Missing responses (for paired comparisons but not for subject covariates) are allowed under a missing at random assumption and specified via `NA`.

Input data (via the first argument `obj` in the function call) is specified either through a dataframe or a datafile in which case `obj` is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command `read.table(datafilename, header = TRUE)`.

The leftmost columns must be the responses to the paired comparisons (where the mandatory order of comparisons is (12) (13) (23) (14) (24) (34) (15) (25) etc.), optionally followed by columns for subject covariates. If categorical, these have to be specified such that the categories are represented by consecutive integers starting with 1. Missing values for subject covariates are not allowed and treated such that rows with NAs are removed from the resulting design structure and a message is printed.

For an example see `xmpl`.

Deprecated

The following options are for backwards compatibility and should no longer be used.

blnCasewise same as `casewise`.

cov.sel same as `cat.scovs`.

Options for requesting GLIM commands and data structures are no longer supported. Specifying the input to `llbt.design` via a control list is also deprecated. If you want to use these features you have to install **prefmod** <= 0.8-22.

Author(s)

Reinhold Hatzinger

References

Dittrich, R., Hatzinger, R., & Katzenbeisser, W. (1998). 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), 47*(4), 511–252. doi:10.1111/1467-9876.00125

See Also

patt.design, llbt.worth, llbtPC.fit

Examples

```
# cems universities example
des <- llbt.design(cemspc, nitems = 6, cat.scovs = "ENG")

res0 <- gnm(y ~ o1+o2+o3+o4+o5+o6 + ENG:(o1+o2+o3+o4+o5+o6),
    eliminate = mu:ENG, data = des, family = poisson)
summary(res0)

# inclusion of g1 allows for an undecided effect
res <- gnm(y ~ o1+o2+o3+o4+o5+o6 + ENG:(o1+o2+o3+o4+o5+o6) + g1,
    eliminate = mu:ENG, data = des, family = poisson)
summary(res)

# calculating and plotting worth parameters
wmat <- llbt.worth(res)
plot(wmat)

# object specific covariates
LAT  <- c(0, 1, 1, 0, 1, 0)        # latin cities
EC   <- c(1, 0, 1, 0, 0, 1)
OBJ  <- data.frame(LAT, EC)
des2 <- llbt.design(cemspc, nitems = 6, objcovs = OBJ, cat.scovs = "ENG")
res2 <- gnm(y ~ LAT + EC + LAT:ENG + g1,
    eliminate = mu:ENG, data = des2, family = poisson)

# calculating and plotting worth parameters
wmat2 <- llbt.worth(res2)
wmat2
plot(wmat2)
```

`llbt.fit` *Function to fit an LLBT*

Description

Function to fit an LLBT using an ELIMINATE feature

Usage

```
llbt.fit(y, Xmodel, q, ncat, maxiter = 100)
```

Arguments

`y`	response, usually counts
`Xmodel`	design matrix
`q`	number of parameters to eliminate (usually number of comparisons times number of subject covariate levels
`ncat`	number of response categories
`maxiter`	maximum number of iterations (default 100)

Details

Be careful when specifying the design matrix. Since there is no extrinsic aliasing the matrix must have full rank. Usually, one of the design columns for object must be left out.

Author(s)

Reinhold Hatzinger

References

Hatzinger, R., & Francis, B. (2004). *Fitting paired comparison models in R.* doi:10.57938/d24575d2-77d24a7ea63dc787cc9d1906

Examples

```
# fit basic model casewise
mfr <- llbt.design(cemspc, nitems = 6,
    objnames = c("lo", "pa", "mi", "sg", "ba", "st"),
    casewise = TRUE)
mm   <- model.matrix(~ lo+pa+mi+sg+ba + g1, data = mfr)
X    <- mm[, -1]
p    <- ncol(X)
ncat <- 3
q    <- length(levels(mfr$mu)) * length(levels(mfr$CASE))
llbt.fit(mfr$y, X, q, ncat)
```

```
# fit the (aggregated) model with one subject covariate
mfr <- llbt.design(cemspc, nitems = 6,
    objnames = c("lo", "pa", "mi", "sg", "ba", "st"),
    cov.sel = "ENG")
eng  <- mfr$ENG
eng  <- factor(eng)
mm   <- model.matrix(~ lo+pa+mi+sg+ba + g1 + (lo+pa+mi+sg+ba):eng, data = mfr)
X    <- mm[, -1]
q    <- length(levels(mfr$mu)) * length(levels(eng))
ncat <- 3
llbt.fit(mfr$y, X, q, ncat)
```

`llbt.worth` *Function to calculate and print worth parameters from LLBT model results*

Description

Worth parameters are calculated from the results of an LLBT model fit, i.e., from `llbtPC.fit` or from a gnm-fit, respectively. For the latter, the function only works if the design matrix had been generated using `llbt.design`.

Usage

```
llbt.worth(fitobj, outmat = "worth")
```

Arguments

`fitobj`	result of an LLBT model fit using either `llbtPC.fit` or `gnm` (having used a design matrix generated by `llbt.design`).
`outmat`	a matrix of estimated worth parameters (`outmat = "worth"`, the default) or LLBT model parameters (`outmat = "lambda"`).

Details

If the LLBT model includes categorical subject covariates, the function provides estimates for all groups formed by the full crossclassification. Numerical subject covariates are not implemented (yet)(see Warning below).

Value

`llbt.worth` returns a matrix of worth or model parameters. If subject covariates have been specified, each column represents a group defined by the crossclassification of the subject covariates.

In case of object-specific covariates (gnm-fit only) the rows are collapsed to the number of different combinations of object-specific covariate values and labelled accordingly. Additionally, there is an attribute `objtable` containing a summary of original objects (items) and their reparameterisation with object-specific covariates. This is a list or a matrix.

The function `plot` gives a plot of the estimates.

Warning

If the LLBT model has been fitted including numerical subject covariates, they are ignored. However, estimates for the remaining predictors are calculated for convenience. Please note, that these cannot be interpreted as standard estimates but are intercepts of the regression model where the objects (or reparameterised objects) are explained by one or more numerical subject covariates.

Note

If a position effect has been fitted (for details see Dittrich, et. al., 1998), the corresponding variable must have been named pos.

Author(s)

Reinhold Hatzinger

See Also

llbtPC.fit, llbt.design, plot

Examples

```
# fit only first three objects with SEX effect
mod <- llbtPC.fit(cemspc, nitems = 3, formel = ~SEX, elim = ~SEX, undec = TRUE)

# calculate and print worth parameters
mw <- llbt.worth(mod)
mw

# the same using llbt.design and gnm
des <- llbt.design(cemspc, nitems = 3, cat.scovs = "SEX")
m2  <- gnm(y ~ o1+o2+o3 + SEX:(o1+o2+o3) + g1, elim = SEX:mu,
  data = des, family = poisson)

# calculate and plot worth parameters
w2 <- llbt.worth(m2)
plot(w2)


# model with object specific covariates
latin <- c(0, 1, 1, 0, 1, 0)     # object-specific covariate
LAT   <- data.frame(LAT = latin) # objcovs must be data frame with named columns

onames <- c("LO", "PA", "MI", "SG", "BA", "ST")

des <- llbt.design(cemspc, nitems = 6, objnames = onames, objcovs = LAT)
m3  <- gnm(y ~ LAT + g1, eliminate = mu, data = des, family = poisson)
w3  <- llbt.worth(m3)
w3
attr(w3, "objtable")
```

`llbtPC.fit` *Function to fit a loglinear Bradley-Terry model for paired comparisons*

Description

Function to fit a loglinear Bradley-Terry for paired comparisons allowing subject covariates and undecided response categories.

Usage

```
llbtPC.fit(obj, nitems, formel = ~1, elim = ~1, resptype = "paircomp",
        obj.names = NULL, undec = TRUE)
```

Arguments

`obj`	either a dataframe or the path/name of the datafile to be read.
`nitems`	the number of compared objects, not the number of comparisons
`formel`	the formula for subject covariates to fit different preference scales for the objects (see below).
`elim`	the formula for the subject covariates that specify the table to be analysed. If omitted and `formel` is not `~1` then `elim` will be set to the highest interaction between all terms contained in `formel`. If `elim` is specified, the terms must be separated by the `*` operator.
`resptype`	is `"paircomp"` by default and is reserved for future usage. Any other specification will not change the behaviour of `llbtPC.fit`
`obj.names`	character vector with names for objects.
`undec`	for paired comparisons with a undecided/neutral category, a common parameter will be estimated if `undec = TRUE`.

Details

Models including categorical subject covariates can be fitted using the `formel` and `elim` arguments. `formel` specifies the actual model to be fitted. For instance, if specified as `formel = ~SEX` different preference scale for the objects will be estimated for males and females. For two or more covariates, the operators `+` or `*` can be used to model main or interaction effects, respectively. The operator `:` is not allowed. See also `formula`.

The specification for `elim` follows the same rules as for `formel`. However, `elim` specifies the basic contingency table to be set up but does not specify any covariates to be fitted. This is done using `formel`. If, e.g., `elim = ~SEX` but `formel = ~1`, then the table is set up as if `SEX` would be fitted but only one global preference scale is computed. This feature allows for the successive fitting of nested models to enable the use of deviance differences for model selection (see example below).

Value

`llbtPC.fit` returns an object of class `llbtMod`. This object is basically a `gnm` object with an additional element `envList`. This is a list with further details like the subject covariates design structure `covdesmat`, the model specification (`formel` and `elim`), the object names (`obj.names`), the number of items (`nobj`) and comparisons (`ncomp`), etc.

The function `llbt.worth` can be used to produce a matrix of estimated worth parameters.

Input Data

The responses have to be coded as 0/1 for paired comparisons without undecided category (0 means first object in a comparison preferred) or 0/1/2 for paired comparisons with an undecided category (where 1 is the undecided category). Optional subject covariates have to be specified such that the categories are represented by consecutive integers starting with 1. Rows with missing values for subject covariates are removed from the data and a message is printed. The leftmost columns in the data must be the responses to the paired comparisons (where the mandatory order of comparisons is (12) (13) (23) (14) (24) (34) (15) (25) etc.), optionally followed by columns for categorical subject covariates.

The data specified via `obj` are supplied using either a data frame or a datafile in which case `obj` is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command `read.table(datafilename, header = TRUE)`.

For an example see `cemspc`.

Note

The function `llbtPC.fit` is a wrapper function for `gnm` and was designed to facilitate fitting of LLBTs with subject covariates and undecided categories. More specialised setups (e.g., object-specific covariates) can be obtained using `llbt.design` and then calling `gnm` (or `glm`) directly (see Examples for `llbt.design`).

Author(s)

Reinhold Hatzinger

See Also

`llbt.design`, `pattL.fit`

Examples

```
# cems universities example
res0 <- llbtPC.fit(cemspc, nitems = 6, formel = ~1,   elim = ~ENG, undec = TRUE)
res1 <- llbtPC.fit(cemspc, nitems = 6, formel = ~ENG, elim = ~ENG, undec = TRUE)

anova(res1, res0)
llbt.worth(res1)
```

music *Data (ratings): Music (US General social survey 1993)*

Description

The dataset contains data on 18 items referring to the liking of musical styles (ratings on a 5-point Likert type response scale) and three subject covariates. The data is an excerpt from the US General Social Survey 1993. Missing values have been removed from the subject variables.

Usage

```
music
```

Format

A data frame with 1597 observations. The variables bigb through hvym are responses to items asking for liking/disliking of type of music: Do you like it very much (1), like it (2), have mixed feelings (3), dislike it (4), dislike it very much (5).

bigb like or dislike bigband music

blug like or dislike bluegrass music

coun like or dislike country western music

blue like or dislike blues or r and b music

musi like or dislike broadway musicals

clas like or dislike classical music

folk like or dislike folk music

gosp like or dislike gospel music

jazz like or dislike jazz

lati like or dislike latin music

mood like or dislike easy listening music

newa like or dislike new age music

oper like or dislike opera

rap like or dislike rap music

regg like or dislike reggae music

conr like or dislike contemporary rock music

oldi like or dislike oldies rock music

hvym like or dislike heavy metal music

age age in years

educ highest year of school completed

sex 1 male, 2 female

Source

Davis, J. A., Smith, T. W., & Marsden, P. V. (2016) *General Social Survey, 1993, 1998, 2000, 2002 with Cultural, Information Security, and Freedom Modules [United States]. Inter-university Consortium for Political and Social Research [distributor].* doi:10.3886/ICPSR35536.v2

Examples

```
summary(music)
```

`patt.design` *Paired Comparison Patterns – Design Matrix Generation*

Description

The function `patt.design` converts (i) real paired comparison responses, or (ii) a set of ratings (or Likert-type responses measured on a common scale), or (iii) full rankings into paired comparison patterns, returning a new data frame containing the design matrix for a loglinear paired comparison model. Additionally, the frequencies of these patterns are computed and are stored in the first column of the data frame.

Usage

```
patt.design(obj, nitems = NULL, objnames = "", objcovs = NULL,
            cat.scovs = NULL, num.scovs = NULL, resptype = "paircomp",
            reverse = FALSE, ia = FALSE, casewise = FALSE, ...)
```

Arguments

`obj`	either a data frame or a data file name.
`nitems`	number of items (objects). `nitems` is not the number of comparisons!
`objnames`	an optional character vector with names for the objects. These names are the columns names in the output data frame. If `objnames` is not specified `o1`, `o2`, etc. will be used.
`objcovs`	an optional data frame with object specific covariates. The rows correspond to the objects, the columns define the covariates. The column names of this data frame are later used to fit the covariates. Factors are not allowed. In that case dummy variables have to be set up manually (favourably using `model.matrix`).
`cat.scovs`	a character vector with the names of the categorical subject covariates in the data file to be included into the design matrix. (example: `cat.scovs = c("SEX", "WORK")`). If all covariates in the data are categorical and should be included, the specification can be abbreviated to `cat.scovs = "ALL"`. In that case, `num.scovs` must not be specified. For no categorical covariates: `cat.scovs = ""`, the default.
`num.scovs`	analogous to `cat.scovs` for numerical (continuous) subject covariates. If any numerical covariates are specified, `casewise` is set to `TRUE`.

resptype	one of "paircomp", "rating", or "ranking".
reverse	If the responses are such that low values correspond to high preference (or agreement or rank) and high values to low preference (or agreement or ranks) (e.g., (1) *I strongly agree* ... (5) *I strongly disagree*) then reverse should be specified to be FALSE, the default. Otherwise set reverse = TRUE. The only exception is paired comparison responses that are coded $-1/1$, $-1/0/1$, $-2/-1/0/1/2$, etc. Then negative numbers are treated as not preferred. (See Input Data below)
ia	generates covariates for interactions between comparisons if ia = TRUE.
casewise	If casewise = TRUE a separate design structure is set up for each subject in the data. This is required when fitting continuous subject covariates. However, the design can become very large in the case of many subjects and/or comparisons. See Details below.
...	deprecated options to allow for backwards compatibility (see Deprecated below).

Details

The function patt.design allows for different scenarios mainly concerning

- **responses.** Currently, three types of responses can be specified.
  - ***paired comparison data.*** Responses can be either simply *preferred – not preferred* or ordinal (*strongly preferred – ... – not at all preferred*). In both cases an undecided category may or may not occur. If there are more than three categories a they are reduced to two or three response categories. The set of paired comparison responses represents a response pattern.
  - ***ratings/Likert type responses.*** The responses to Likert type items are transformed to paired comparison responses by calculating the difference between each pair of the Likert items. This leads to an ordinal (adjacent categories) paired comparison model with $2k$-1 response categories where k is the number of the (original) Likert categories. Again, the transformed ratings are reduced to three response categories (*preferred – undecided – not preferred*).
  - ***rankings.*** Currently only full rankings are allowed, i.e., a (consecutive) integer must uniquely be assigned to each object in a list according to the (subjective) ordering. Ties are not allowed. As for ratings, the rankings are transformed to paired comparison responses by calculating the difference between each pair of the ranks. Again a category reduction (as described above) is automatically performed.
- **comparison covariates.** The design matrix for the basic model has columns for the items (objects) and (depending on the type of responses) for undecided comparisons. For ratings (Likert type) undecided comparisons occur if any subject has responded to two items in the same category. For paired comparisons it depends on the design. For rankings there are no undecided categories. If undecided categories occur there is one dummy variable for each comparison. Additionally, covariates for two way interaction between comparisons (i.e., for effects resulting from the dependence between two comparisons that have one item in common) can be obtained by setting ia = TRUE.
- **object specific covariates.** For modelling certain characteristics of objects a reparameterisation can be included in the design. This is sometimes called conjoint analysis. The object specific covariates can be continuous or dummy variables. For the specification see Argument objcovs above.

- **subject covariates.** For modelling different preference scales for the items according to characteristics of the respondents categorical subject covariates can be included in the design. The corresponding variables are defined as numerical vectors where the levels are specified with consecutive integers starting with 1. This format must be used in the input data file and is also used in all outputs.

Value

The output is a dataframe. Each row represents a unique response pattern. If subject covariates are specified, each row instead represents a particular combination of a unique covariate combination with a response pattern. All possible combinations are generated.

The first column contains the counts for the paired comparison response patterns and is labelled with `Y`. The next columns are the covariates for the items and the undecided category effects (one for each comparison). These are labelled as `u12`, `u13`, etc., where `12` denotes the comparison between items `1` and `2`. Optionally, covariates for dependencies between comparisons follow. The columns are labelled `Ia.bc` denoting the interaction of the comparisons between items `(a, b)` and `(a, c)` where the common item is `a`. If subject covariates are present they are in the rightmost columns and defined to be factors.

Input Data

Responses have to be coded as consecutive integers (e.g., (0, 1), or (1, 2, 3, ...), where the smallest value corresponds to (highest) preference for the first object in a comparison.

For (ordinal) paired comparison data (`resptype = "paircomp"`) the codings $(1, -1)$, $(2, 1, -1, -2)$, $(1, 0, -1)$, $(2, 1, 0, -1, -2)$ etc. can also be used. Then negative numbers correspond to not preferred, 0 to undecided. Missing responses are not allowed (use functions `pattPC.fit`, `pattL.fit`, or `pattR.fit` instead).

Input data (via the first argument `obj` in the function call) is specified either through a dataframe or a datafile in which case `obj` is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command `read.table(datafilename, header = TRUE)`.

The leftmost columns must be the responses to the paired comparisons, ratings (Likert items), or rankings. For paired comparisons the mandatory order is of comparisons is (12) (13) (23) (14) (24) (34) (15) (25) etc. For rankings, the lowest value means highest rank according to the underlying scale. Each column in the data file corresponds to one of the ranked objects. For example, if we have 3 objects denoted by `A`, `B`, and `C`, with corresponding columns in the data matrix, the response pattern `(3, 1, 2)` represents: object `B` ranked highest, `C` ranked second, and `A` ranked lowest. For ratings. again the lowest value means highest 'endorsement' (agreement) according to the underlying scale. All items are assumed to have the same number of response category.

The columns for responses are optionally followed by columns for subject covariates. If categorical, they have to be specified such that the categories are represented by consecutive integers starting with 1. Missing values are not allowed and treated such that rows with `NA`s are removed from the resulting design structure and a message is printed. For an example see `xmpl`.

(Besides supplying data via a dataframe or a datafile name, `obj` can also be specified as a control list with the same elements as the arguments in the function call. The data must then be specified as a path/filename using the element `datafile = "filename"`. The control list feature is deprecated. An example is given below.)

Deprecated

The following options are for backwards compatibility and should no longer be used.

blnCasewise same as `casewise`.

blnIntcovs same as `ia`.

blnRevert same as `reverse`.

cov.sel same as `cat.scovs`.

Options for requesting GLIM commands and data structures are no longer supported. Specifying the input to `llbt.design` via a control list is also deprecated. If you want to use these features you have to install **prefmod** <= 0.8-22.

Author(s)

Reinhold Hatzinger

References

Dittrich, R., Francis, B.J., Hatzinger R., Katzenbeisser, W. (2007), A Paired Comparison Approach for the Analysis of Sets of Likert Scale Responses. *Statistical Modelling*, Vol. 7, No. 1, 3–28.

See Also

`llbt.design`, `pattPC.fit`, `pattL.fit`, `pattR.fit`

Examples

```
# mini example with three Likert items and two subject covariates
dsgnmat <- patt.design(xmpl, nitems = 3, resptype = "rating",
      ia = TRUE, cov.sel = "ALL")
head(dsgnmat)


# ILLUSTRATING THE ISSP2000 EXAMPLE
# simplified version of the analysis as given in Dittrich et. al (2007).
design <- patt.design(issp2000, nitems = 6, resptype = "rating",
      cov.sel = c("SEX", "EDU"))


# - fit null multinomial model (basic model for items without subject
#     covariates) through Poisson distribution.
# - SEX:EDU parameters are nuisance parameters
# - the last item (GENE) becomes a reference item in the model and is aliased;
#     all other items are compared to this last item

# item parameters with undecided effects and no covariate effects.
summary(glm(y ~ SEX*EDU
  + CAR+IND+FARM+WATER+TEMP+GENE
  + u12+u13+u23+u14+u24+u34+u15+u25+u35+u45+u16+u26+u36+u46+u56,
  data = design, family = poisson))
```

```
# now add main effect of SEX on items
summary(glm(y ~ SEX:EDU
  + CAR+IND+FARM+WATER+TEMP+GENE
  + (CAR+IND+FARM+WATER+TEMP+GENE):SEX
  + u12+u13+u23+u14+u24+u34+u15+u25+u35+u45+u16+u26+u36+u46+u56,
  data = design, family = poisson))
```

`patt.worth` *Function to calculate and print worth parameters from pattern model results*

Description

Worth parameter are calculated from the results of a pattern model fit, i.e., from `pattPC.fit`, `pattR.fit`, `pattL.fit`, and `pattLrep.fit` or from a gnm-fit, respectively. For the latter, the function only works if the design matrix had been generated using `patt.design`.

Usage

```
patt.worth(fitobj, obj.names = NULL, outmat = "worth")
```

Arguments

`fitobj`	Object of class `pattMod` obtained from pattern model fit.
`obj.names`	names for the objects, for repeated measurement models just the names of objects for the first time point
`outmat`	a matrix of estimated worth parameters (`outmat = "worth"`, the default) or pattern model parameters (`outmat = "lambda"`).

Details

If the pattern model includes categorical subject covariates, the function provides estimates for all groups formed by the full crossclassification. Numerical subject covariates are not implemented (yet)(see Warning below).

Value

`patt.worth` returns a matrix of worth or model parameters. If subject covariates have been specified, each column represents a groups defined by the crossclassification of the subject covariates.

The function `plot` gives a plot of the estimates.

Warning

If the pattern model has been fitted including numerical subject covariates, they are ignored. However, estimates for the remaining predictors are calculated for convenience. Please note, that these cannot be interpreted as standard estimates but are intercepts of the regression model where the objects (or reparameterised objects) are explained by one or more numerical subject covariates.

Author(s)

Reinhold Hatzinger

See Also

pattPC.fit, pattR.fit, pattL.fit, pattLrep.fit, plot

Examples

```
# fit only first three objects with SEX effect
m2 <- pattPC.fit(cemspc, nitems = 3, formel = ~SEX, elim = ~SEX, undec = TRUE)

# calculate and print worth parameters
m2worth <- patt.worth(m2)
m2worth
```

---

pattL.fit *Function to fit a pattern model for ratings (Likert items)*

Description

Function to fit a pattern model for ratings/Likert items (transformed to paired comparisons) allowing for missing values using a CL approach.

Usage

```
pattL.fit(obj, nitems, formel = ~1, elim = ~1, resptype = "rating",
        obj.names = NULL, undec = TRUE, ia = FALSE, NItest = FALSE,
        pr.it = FALSE)
```

Arguments

obj	either a dataframe or the path/name of the datafile to be read.
nitems	the number of items
formel	the formula for subject covariates to fit different preference scales for the objects (see below).
elim	the formula for the subject covariates that specify the table to be analysed. If omitted and formel is not ~1 then elim will be set to the highest interaction between all terms contained in formel. If elim is specified, the terms must be separated by the * operator.
resptype	is "rating" by default and is reserved for future usage. Any other specification will not change the behaviour of pattL.fit
obj.names	character vector with names for objects.
undec	for paired comparisons with a undecided/neutral category, a common parameter will be estimated if undec = TRUE.

ia	interaction parameters between comparisons that have one object in common if ia = TRUE.
NItest	separate estimation of object parameters for complete and incomplete patterns if NItest = TRUE. Currently, NItest is set to FALSE if subject covariates are specified.
pr.it	a dot is printed at each iteration cycle if set to TRUE

Details

Models including categorical subject covariates can be fitted using the formel and elim arguments. formel specifies the actual model to be fitted. For instance, if specified as formel = ~SEX different preference scale for the objects will be estimated for males and females. For two or more covariates, the operators + or * can be used to model main or interaction effects, respectively. The operator : is not allowed. See also formula.

The specification for elim follows the same rules as for formel. However, elim specifies the basic contingency table to be set up but does not specify any covariates to be fitted. This is done using formel. If, e.g., elim = ~SEX but formel = ~1, then the table is set up as if SEX would be fitted but only one global preference scale is computed. This feature allows for the succesive fitting of nested models to enable the use of deviance differences for model selection (see example below).

Value

pattL.fit returns an object of class pattMod. The function print (i.e., print.pattMod) can be used to print the results and the function patt.worth to produce a matrix of worth parameters.

An object of class pattMod is a list containing the following components:

coefficients	estimates
ll	log-likelihood of the model
fl	log-likelihood of the saturated model
call	function call
result	a list of results from the fitting routine (see Value of nlm).
envList	a list with further fit details like subject covariates design structure covdesmat, paired comparison response pattern matrix Y, etc.
partsList	a list of the basic data structures for each subgroup defined by crossing all covariate levels and different missing value patterns. Each element of partsList is again a list containing counts, missing value pattern, the CL matrix represented as a vector, and the specification of the covariates. Use str to inspect the elements and see example below.

Input Data

The responses have to be coded as consecutive integers starting with 1 (or 0). The value of 1 (0) means highest 'endorsement' (agreement) according to the underlying scale. Missing values are coded as NA, rows with less than 2 valid responses are removed from the fit and a message is printed.

Optional subject covariates have to be specified such that the categories are represented by consecutive integers starting with 1. Rows with missing values for subject covariates are removed from

the data and a message is printed. The leftmost columns in the data must be the rankings, optionally followed by columns for categorical subject covariates.

The data specified via `obj` are supplied using either a data frame or a datafile in which case `obj` is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command `read.table(datafilename, header = TRUE)`.

For an example see `issp2000`.

Warning

The size of the table to be analysed increases dramatically with the number of items. For ratings (Likert items) the number of paired comparison response categories is always three. The number of rows of the table to set up the design matrix is initially $(2 * numberofcategories - 1)^{(numberofitems)}$, e.g., for six items with 5 response categories each this is 531441. A reasonable maximum number of items with five response categories to be analysed with pattern models is 7.

Author(s)

Reinhold Hatzinger

See Also

`patt.design`, `pattPC.fit`, `pattR.fit`

Examples

```
# fit only four items
music4 <- music[, c("jazz", "blue", "folk", "rap")]
pattL.fit(music4, nitems = 4)

# fit additional undecided effect
pattL.fit(music4, nitems = 4, undec = TRUE)

# fit dependence parameters
## Not run: pattL.fit(music4, nitems = 4, undec = TRUE, ia = TRUE)

# check for ignorable missing
pattL.fit(music4, nitems = 4, undec = TRUE, NItest = TRUE)
```

---

pattLrep.fit *Function to fit a pattern model for repeated ratings (Likert items)*

Description

Function to fit a pattern model for repeated ratings/Likert items (transformed to paired comparisons) allowing for missing values using a CL approach.

Usage

```
pattLrep.fit(obj, nitems, tpoints = 1, formel = ~1, elim = ~1,
        resptype = "ratingT",  obj.names = NULL, undec = TRUE, ia = FALSE,
        iaT = FALSE, NItest = FALSE, pr.it = FALSE)
```

Arguments

obj	either a dataframe or the path/name of the datafile to be read.
nitems	the number of items at one time point.
tpoints	the number of time points.
formel	the formula for subject covariates to fit different preference scales for the objects (see below).
elim	the formula for the subject covariates that specify the table to be analysed. If omitted and formel is not ~1 then elim will be set to the highest interaction between all terms contained in formel. If elim is specified, the terms must be separated by the * operator.
resptype	is "ratingT" by default and is reserved for future usage. Any other specification will not change the behaviour of pattLrep.fit
obj.names	character vector with names for objects.
undec	for paired comparisons with a undecided/neutral category, a common parameter will be estimated if undec = TRUE.
ia	for each time point interaction parameters between comparisons that have one object in common if ia = TRUE.
iaT	if iaT = TRUE, dependence parameters for each item between two successive time points.
NItest	separate estimation of object parameters for complete and incomplete patterns if NItest = TRUE. Currently, NItest is set to FALSE if subject covariates are specified.
pr.it	a dot is printed at each iteration cycle if set to TRUE

Details

Models including categorical subject covariates can be fitted using the formel and elim arguments. formel specifies the actual model to be fitted. For instance, if specified as formel = ~SEX different preference scale for the objects will be estimated for males and females. For two or more covariates, the operators + or * can be used to model main or interaction effects, respectively. The operator : is not allowed (redundant terms are removed automatically). See also formula.

The specification for elim follows the same rules as for formel. However, elim specifies the basic contingency table to be set up but does not specify any covariates to be fitted. This is done using formel. If, e.g., elim = ~SEX but formel = ~1, then the table is set up as if SEX would be fitted but only one global preference scale is computed. This feature allows for the successive fitting of nested models to enable the use of deviance differences for model selection (see example below).

Value

pattLrep.fit returns an object of class pattMod. The function print (i.e., print.pattMod) can be used to print the results and the function patt.worth to produce a matrix of worth parameters.

An object of class pattMod is a list containing the following components:

coefficients	estimates
ll	log-likelihood of the model
fl	log-likelihood of the saturated model
call	function call
result	a list of results from the fitting routine (see Value of nlm).
envList	a list with further fit details like subject covariates design structure covdesmat, paired comparison response pattern matrix Y, etc.
partsList	a list of the basic data structures for each subgroup defined by crossing all covariate levels and different missing value patterns. Each element of partsList is again a list containing counts, missing value pattern, the CL matrix represented as the vector s, and the specification of the covariates. Use str to inspect the elements and see example below.

Input Data

The input data must have the following order (from left to right): all items at first time point, all items at second time point (with the same order as before), etc. for the other time points, optional subject covariates. The responses have to be coded as consecutive integers starting with 1 (or 0). The value of 1 (0) means highest 'endorsement' (agreement) according to the underlying scale. Missing values are coded as NA, rows with less than 2 valid responses are removed from the fit and a message is printed.

Optional subject covariates have to be specified such that the categories are represented by consecutive integers starting with 1. Rows with missing values for subject covariates are removed from the data and a message is printed. Again, the leftmost columns in the data must be the ratings, optionally followed by columns for categorical subject covariates.

The data specified via obj are supplied using either a data frame or a datafile in which case obj is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command read.table(datafilename, header = TRUE).

Warning

The size of the table to be analysed increases dramatically with the number of items and time points. For ratings (Likert items) the number of paired comparison response categories is always three. For each time point the number of rows of the table to set up the design matrix is initially $(2 * numberofcategories - 1)^{(numberofitems)}$. After reducing to three categories the number of patterns are 13, 75, 541 for 3 to 5 items, respectively. Generally, the number of rows in the design matrix is $(numberofpatterns)^{(numberoftimepoints)}$. The number of covariate levels and the number of missing value patterns have effects only on the run time. A (reasonable) maximum number of items for two time points is 5, for three timepoints 4, and for four to five timepoints 3.

Note

The number of timepoints can also be regarded as different response dimensions.

Author(s)

Reinhold Hatzinger

See Also

pattL.fit, patt.design, pattPC.fit, pattR.fit, pattRrep.fit

Examples

```
# simulated data: 3 items, 2 timepoints
dat <- as.data.frame(matrix(sample(1:5, 300, replace = TRUE), ncol = 6))
res <- pattLrep.fit(dat, nitems = 3, tpoints = 2, iaT = TRUE)
res
patt.worth(res, obj.names = LETTERS[1:3])
```

---

pattnpml.fit *NPML estimation for paired comparison models*

Description

Fits a mixture model to overdispersed paired comparison data using non-parametric maximum likelihood (Aitkin, 1996a).

Usage

```
pattnpml.fit(formula, random = ~1, k = 1, design,
        tol = 0.5, startp = NULL, EMmaxit = 500, EMdev.change = 0.001,
        seed = NULL, pr.it = FALSE)
```

Arguments

formula	A formula defining the response (the count of the number of cases of each pattern) and the fixed effects (e.g. y ~ x).
random	A formula defining the random model. If there are three objects labelled o1, o2, o3, set random = ~o1+o2+o3 to model overdispersion. For more details, see below.
k	The number of mass points (latent classes). Up to 21 mass points are supported.
design	The design data frame for paired comparison data as generated using patt.design (mandatory, even if it is attached to the workspace!).
tol	The tol scalar (usually, $0 <$ tol ≤ 1). This scalar sets the scaling factor for the locations of the initial mass points. A larger value means that the starting point locations are more widely spread.

startp	Optional numerical vector of length k specifying the starting probabilities for the mass points to initialise the EM algorithm. The default is to take Gaussian quadrature probabilities.
EMmaxit	The maximum number of EM iterations.
EMdev.change	Stops EM algorithm when deviance change falls below this value.
seed	Seed for random weights. If NULL, the seed is set using the system time.
pr.it	A dot is printed at each iteration cycle of the EM algorithm if set to TRUE.

Details

The function pattnpml.fit is a wrapper function for alldistPC which in turn is a modified version of the function alldist from the **npmlreg** package.

The non-parametric maximum likelihood (NPML) approach was introduced in Aitkin (1996) as a tool to fit overdispersed generalised linear models. The idea is to approximate the unknown and unspecified distribution of the random effect by a discrete mixture of exponential family densities, leading to a simple expression of the marginal likelihood which can then be maximised using a standard EM algorithm.

This function extends the NPML approach to allow fitting of overdispersed paired comparison models. It assumes that overdispersion arises because of dependence in the patterns. Fitting a non-parametric random effects term is equivalent to specifying distinct latent classes of response patterns.

The number of components k of the finite mixture has to be specified beforehand.

The EM algorithm used by the function takes the Gauss-Hermite masses and mass points as starting points. The position of the starting points can be concentrated or extended by setting tol smaller or larger, respectively; the initial mass point probabilities of the starting points can also be specified through startp.

Fitting models for overdispersion can be achieved by specifying the paired comparison items as additive terms in the random part of the model formula. A separate estimate for each item and for each mass point is produced.

Fitting subject covariate models with the same effect for each mass point component is achieved by specifying as part of the formula a) a subject factor giving a different estimate for each covariate combination b) an interaction of the chosen subject covariates with the objects. For models with subject factor covariates only, the first term is simply the interaction of all of the factor covariates.

Fitting subject covariate models with a different effect for each mass point component (sometimes called random coefficient models, see Aitkin, Francis, Hinde and Darnell, 2009, pp. 497) is possible by specifying an interaction of the subject covariates with the items in the random term, and also in the formula part. Thus the setting random = ~x:(o1+o2+o3 gives a model with a set of random slopes (one set for each mass point) and a set of random intercepts, one set for each mass point.

The AIC and BIC functions from the **stats**-package can be used.

Value

The function produces an object of class pattNPML. The object contains the following 29 components:

coefficients	a named vector of coefficients (including the mass points). In case of Gaussian quadrature, the coefficient given at z corresponds to the standard deviation of the mixing distribution.
residuals	the difference between the true response and the empirical Bayes predictions.
fitted.values	the empirical Bayes predictions (Aitkin, 1996b) on the scale of the responses.
family	the 'family' object used.
linear.predictors	the extended linear predictors $\hat{\eta}_{ik}$.
disparity	the disparity (-2logL) of the fitted mixture regression model.
deviance	the deviance of the fitted mixture regression model.
null.deviance	The deviance for the null model (just containing an intercept), comparable with 'deviance.'
df.residual	the residual degrees of freedom of the fitted model (including the random part).
df.null	the residual degrees of freedom for the null model.
y	the (extended) response vector.
call	the matched call.
formula	the formula supplied.
random	the random term of the model formula.
data	the data argument.
model	the (extended) design matrix.
weights	the case weights initially supplied.
offset	the offset initially supplied.
mass.points	the fitted mass points.
masses	the mass point probabilities corresponding to the patterns.
sdev	a list of the two elements sdev$sdev and sdev$sdevk. The former is the estimated standard deviation of the Gaussian mixture components (estimated over all mixture components), and the latter gives the unequal or smooth component-specific standard deviations. All values are equal if lambda = 0.
shape	a list of the two elements shape$shape and shape$shapek, to be interpreted in analogy to sdev.
rsdev	estimated random effect standard deviation.
post.prob	a matrix of posteriori probabilities.
post.int	a vector of 'posteriori intercepts' (as in Sofroniou et al. (2006)).
ebp	the empirical Bayes Predictions on the scale of the linear predictor. For compatibility with older versions.
EMiter	gives the number of iterations of the EM algorithm.
EMconverged	logical value indicating if the EM algorithm converged.
lastglm	the fitted glm object from the last EM iteration.
Misc	contains additional information relevant for the summary and plot functions, in particular the disparity trend and the EM trajectories.

For further details see the help file for function alldist in package **npmlreg**.

Note

The mass point probabilities given in the output are the proportion of patterns estimated to contribute to each mass point. To estimate the proportion of cases contributing to each mass point the posterior probabilities need to be averaged over patterns with observed counts as weights (see example below).

Author(s)

Originally translated from the GLIM 4 functions `alldist` and `allvc` (Aitkin & Francis, 1995) to R by Ross Darnell (2002). Modified, extended, and prepared for publication by Jochen Einbeck and John Hinde (2006). Adapted for paired comparison modelling by Reinhold Hatzinger and Brian Francis (2009).

References

Aitkin, M. (1996). A general maximum likelihood analysis of overdispersion in generalized linear models. *Statistics and Computing, 6*(3), 251–262. doi:10.1007/BF00140869

Aitkin, M., Francis, B., Hinde, J., & Darnell, R. (2009). *Statistical Modelling in R*. Oxford: Oxford University Press.

Einbeck, J., & Hinde, J. (2006). A Note on NPML Estimation for Exponential Family Regression Models with Unspecified Dispersion Parameter. *Austrian Journal of Statistics, 35*(2&3), 233–243.

Sofroniou, N., Einbeck, J., & Hinde, J. (2006). Analyzing Irish suicide rates with mixture models. Proceedings of the 21st International Workshop on Statistical Modelling in Galway, Ireland, 2006.

See Also

`glm`

Examples

```
# two latent classes for paired comparison data
dfr   <- patt.design(dat4, 4)
modPC <- pattnpml.fit(y ~ 1, random = ~o1 + o2 + o3, k = 2, design = dfr)
modPC

# estimated proportion of cases in each mixture component
apply(modPC$post.prob, 2, function(x){ sum(x * dfr$y / sum(dfr$y)) })

## Not run:
# fitting a model for two latent classes and fixed categorical subject
# covariates to the Eurobarometer 55.2 data (see help("euro55.2.des"))
# on rankings of sources of information on scientific developments

model2cl <- pattnpml.fit(
  y ~ SEX:AGE4 + (SEX + AGE4):(TV + RAD + NEWSP + SCIMAG + WWW + EDINST) - 1,
  random = ~ TV + RAD + NEWSP + SCIMAG + WWW + EDINST,
  k = 2, design = euro55.2.des, pr.it = TRUE)
summary(model2cl)
BIC(model2cl)
## End(Not run)
```

pattPC.fit *Function to fit a pattern model for paired comparisons*

Description

Function to fit a pattern model for paired comparisons allowing for missing values using a CL approach.

Usage

```
pattPC.fit(obj, nitems, formel = ~1, elim = ~1, resptype = "paircomp",
    obj.names = NULL, undec = TRUE, ia = FALSE, NItest = FALSE,
    NI = FALSE, MIScommon = FALSE, MISmodel = "obj", MISalpha = NULL,
    MISbeta = NULL, pr.it = FALSE)
```

Arguments

obj	either a dataframe or the path/name of the datafile to be read.
nitems	the number of compared objects, not the number of comparisons
formel	the formula for subject covariates to fit different preference scales for the objects (see below).
elim	the formula for the subject covariates that specify the table to be analysed. If omitted and formel is not ~1 then elim will be set to the highest interaction between all terms contained in formel. If elim is specified, the terms must be separated by the * operator.
resptype	is "paircomp" by default and is reserved for future usage. Any other specification will not change the behaviour of pattPC.fit
obj.names	character vector with names for objects.
undec	for paired comparisons with a undecided/neutral category, a common parameter will be estimated if undec = TRUE.
ia	interaction parameters between comparisons that have one object in common if ia = TRUE.
NItest	separate estimation of object parameters for complete and incomplete patterns if NItest = TRUE. Currently, NItest is set to FALSE if subject covariates are specified.
NI	if TRUE, fits large table (crossclassification with NA patterns), for comparison with models including MISalpha (and MISbeta).
MIScommon	if TRUE, fits a common parameter for NA indicators, i.e., $\alpha = \alpha_i = \alpha_j = \ldots$.
MISmodel	either obj (default) or comp. Specifies the model for missing values. If obj, parameters for missing objects or if comp, parameters for missing comparisons are estimated. In both cases MISalpha (and optionally MISbeta) specify which parameters are involved.

MISalpha	if not NULL, specification to fit parameters for NA indicators using a logical vector, where TRUE means that the NA indicator parameter for the corresponding object (or comparison, dependent on MISmodel) should be estimated (see example below). For MISmodel = "obj" each comparison is reparameterised with $\alpha_i + \alpha_j$, for MISmodel = "comp" each comparison is reparameterised with α_{ij}.
MISbeta	if not NULL, fits parameters for MNAR model, i.e., interactions between outcome model parameters and NA indicator parameters. The reparameterisation is analogous to MISalpha. Usually, the specification for MISbeta is the same as for MISalpha, but any subset is reasonable. If MISalpha = NULL but MISbeta is not, then MISalpha is set to MISbeta.
pr.it	a dot is printed at each iteration cycle if set to TRUE

Details

Models including categorical subject covariates can be fitted using the formel and elim arguments. formel specifies the actual model to be fitted. For instance, if specified as formel = ~SEX different preference scale for the objects will be estimated for males and females. For two or more covariates, the operators + or * can be used to model main or interaction effects, respectively. The operator : is not allowed. See also formula.

The specification for elim follows the same rules as for formel. However, elim specifies the basic contingency table to be set up but does not specify any covariates to be fitted. This is done using formel. If, e.g., elim = ~SEX but formel = ~1, then the table is set up as if SEX would be fitted but only one global preference scale is computed. This feature allows for the successive fitting of nested models to enable the use of deviance differences for model selection (see example below).

Value

pattPC.fit returns an object of class pattMod. The function print (i.e., print.pattMod) can be used to print the results and the function patt.worth to produce a matrix of the estimated worth parameters.

An object of class pattMod is a list containing the following components:

coefficients	estimates
ll	log-likelihood of the model
fl	log-likelihood of the saturated model
call	function call
result	a list of results from the fitting routine (see Value of nlm).
envList	a list with further fit details like subject covariates design structure covdesmat, paired comparison response pattern matrix Y, etc.
partsList	a list of the basic data structures for each subgroup defined by crossing all covariate levels and different missing value patterns. Each element of partsList is again a list containing counts, missing value pattern, the CL matrix represented as a vector, and the specification of the covariates. Use str to inspect the elements and see example below.

Input Data

The responses have to be coded as 0/1 for paired comparisons without undecided category (0 means first object in a comparison preferred) or 0/1/2 for paired comparisons with an undecided category (where 1 is the undecided category). Optional subject covariates have to be specified such that the categories are represented by consecutive integers starting with 1. Rows with missing values for subject covariates are removed from the data and a message is printed. The leftmost columns in the data must be the responses to the paired comparisons (where the mandatory order of comparisons is (12) (13) (23) (14) (24) (34) (15) (25) etc.), optionally followed by columns for categorical subject covariates.

The data specified via `obj` are supplied using either a data frame or a datafile in which case `obj` is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command `read.table(datafilename, header = TRUE)`.

For an example see `cemspc`.

Warning

The size of the table to be analysed increases dramatically with the number of objects. For paired comparisons with two response categories the number of rows of the table is $2^{(\text{number of comparisons})}$, e.g., with six objects this is 32768, for three response categories this is 14348907. A reasonable maximum number of objects to be analysed with pattern models is 6 in the case of two response categories and 5 when an additional undecided/neutral category has been observed).

Author(s)

Reinhold Hatzinger

See Also

`patt.design`, `checkMIS`, `pattL.fit`, `pattR.fit`

Examples

```
# fit only first three objects with undecided parameter
pattPC.fit(cemspc, nitems = 3, undec = TRUE)

# check for ignorable missing
pattPC.fit(cemspc, nitems = 3, undec = TRUE, NItest = TRUE)

# check if SEX has an effect
m1 <- pattPC.fit(cemspc, nitems = 3, formel = ~1,   elim = ~SEX, undec = TRUE)
m2 <- pattPC.fit(cemspc, nitems = 3, formel = ~SEX, elim = ~SEX, undec = TRUE)

# calculate LR test for SEX
ll1 <- m1$result$minimum
ll2 <- m2$result$minimum
df1 <- length(m1$result$estimate)
df2 <- length(m2$result$estimate)
lr  <- 2*(ll1 - ll2)
df  <- df2 - df1
```

```
cat("LR test = ", lr, " on df = ", df, " (p = ",
    round(pchisq(lr, df, lower.tail = FALSE), digits = 5), ")\n", sep = "")


# generates data set with three items and some missing values in
# comparison (23), column 3, then there are no NAs for object 1
data3 <- dat4[, 1:3]
idx3  <- sample(1:100, 10)
data3[idx3, 3] <- NA
checkMIS(data3, nitems = 3, verbose = TRUE)

# estimate MNAR PC pattern model for data3 without alpha1 and beta1
pattPC.fit(data3, nitems = 3,
  MISalpha = c(FALSE, TRUE, TRUE),
  MISbeta  = c(FALSE, TRUE, TRUE))
```

`pattR.fit` *Function to fit a pattern model for (partial) rankings*

Description

Function to fit a pattern model for (partial) rankings (transformed to paired comparisons) allowing for missing values using a CL approach.

Usage

```
pattR.fit(obj, nitems, formel = ~1, elim = ~1, resptype = "ranking",
        obj.names = NULL, ia = FALSE, NItest = FALSE, pr.it = FALSE)
```

Arguments

`obj`	either a dataframe or the path/name of the datafile to be read.
`nitems`	the number of items
`formel`	the formula for subject covariates to fit different preference scales for the objects (see below).
`elim`	the formula for the subject covariates that specify the table to be analysed. If omitted and `formel` is not `~1` then `elim` will be set to the highest interaction between all terms contained in `formel`. If `elim` is specified, the terms must be separated by the `*` operator.
`resptype`	is `"ranking"` by default and is reserved for future usage. Any other specification will not change the behaviour of `pattR.fit`
`obj.names`	character vector with names for objects.
`ia`	interaction parameters between comparisons that have one object in common if `ia = TRUE`.
`NItest`	separate estimation of object parameters for complete and incomplete patterns if `NItest = TRUE`. Currently, `NItest` is set to `FALSE` if subject covariates are specified.
`pr.it`	a dot is printed at each iteration cycle if set to `TRUE`

Details

Models including categorical subject covariates can be fitted using the `formel` and `elim` arguments. `formel` specifies the actual model to be fitted. For instance, if specified as `formel = ~SEX` different preference scale for the objects will be estimated for males and females. For two or more covariates, the operators `+` or `*` can be used to model main or interaction effects, respectively. The operator `:` is not allowed. See also `formula`.

The specification for `elim` follows the same rules as for `formel`. However, `elim` specifies the basic contingency table to be set up but does not specify any covariates to be fitted. This is done using `formel`. If, e.g., `elim = ~SEX` but `formel = ~1`, then the table is set up as if `SEX` would be fitted but only one global preference scale is computed. This feature allows for the successive fitting of nested models to enable the use of deviance differences for model selection (see example below).

Value

`pattR.fit` returns an object of class `pattMod`. The function `print` (i.e., `print.pattMod`) can be used to print the results and the function `patt.worth` to produce a matrix of worth parameters.

An object of class `pattMod` is a list containing the following components:

`coefficients`	estimates
`ll`	log-likelihood of the model
`fl`	log-likelihood of the saturated model
`call`	function call
`result`	a list of results from the fitting routine (see Value of `nlm`).
`envList`	a list with further fit details like subject covariates design structure `covdesmat`, paired comparison response pattern matrix `Y`, etc.
`partsList`	a list of the basic data structures for each subgroup defined by crossing all covariate levels and different missing value patterns. Each element of `partsList` is again a list containing counts, missing value pattern, the CL matrix represented as a vector, and the specification of the covariates. Use `str` to inspect the elements and see example below.

Input Data

The responses have to be coded as consecutive integers starting with 1. The value of 1 means highest rank according to the underlying scale. Each column in the data file corresponds to one of the ranked objects. For example, if we have 3 objects denoted by `A`, `B`, and `C`, with corresponding columns in the data matrix, the response pattern `(3, 1, 2)` represents: object `B` ranked highest, `C` ranked second, and `A` ranked lowest. Missing values are coded as `NA`, ties are not allowed (in that case use `pattL.fit`. Rows with less than 2 ranked objects are removed from the fit and a message is printed.

Optional subject covariates have to be specified such that the categories are represented by consecutive integers starting with 1. Rows with missing values for subject covariates are removed from the data and a message is printed. The leftmost columns in the data must be the rankings, optionally followed by columns for categorical subject covariates.

The data specified via obj are supplied using either a data frame or a datafile in which case obj is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command read.table(datafilename, header = TRUE).

For an example without covariates and no missing values see salad.

Warning

The size of the table to be analysed increases dramatically with the number of items. For rankings the number of paired comparison response categories is always two. The number of rows of the table used to set up the design matrix is factorial(number of items). For instance, for nine objects this is 362880. A reasonable maximum number of items is 8.

The option NItest = TRUE has to be used with care. The meaning of missing responses is not obvious with partial rankings. Are the corresponding values really missing or just not chosen.

Author(s)

Reinhold Hatzinger

See Also

patt.design, pattL.fit, pattPC.fit

Examples

```
# fit of Critchlov & Fligner (1991) Salad Dressings Data
pattR.fit(salad, nitems = 4)

# alternatively use glm() with patt.design()
sal <- patt.design(salad, nitems = 4, resptype = "ranking")
glm(y ~ A+B+C+D, data = sal, family = poisson)
```

---

pattRrep.fit *Function to fit a pattern model for repeated rankings)*

Description

Function to fit a pattern model for repeated (partial) rankings (transformed to paired comparisons) allowing for missing values using a CL approach.

Usage

```
pattRrep.fit(obj, nitems, tpoints = 1, formel = ~1, elim = ~1,
        resptype = "rankingT", obj.names = NULL, ia = FALSE,
        iaT = FALSE, NItest = FALSE, pr.it = FALSE)
```

Arguments

obj	either a dataframe or the path/name of the datafile to be read.
nitems	the number of items at one time point.
tpoints	the number of time points (must be > 1).
formel	the formula for subject covariates to fit different preference scales for the objects (see below).
elim	the formula for the subject covariates that specify the table to be analysed. If omitted and formel is not ~1 then elim will be set to the highest interaction between all terms contained in formel. If elim is specified, the terms must be separated by the * operator.
resptype	is "rankingT" by default and is reserved for future usage. Any other specification will not change the behaviour of pattL.fit
obj.names	character vector with names for objects.
ia	FALSE by default, has no meaning for rankings. Reserved for future usage.
iaT	if iaT = TRUE, dependence parameters for each item between two successive time points.
NItest	separate estimation of object parameters for complete and incomplete patterns if NItest = TRUE. Currently, NItest is set to FALSE if subject covariates are specified.
pr.it	a dot is printed at each iteration cycle if set to TRUE

Details

Models including categorical subject covariates can be fitted using the formel and elim arguments. formel specifies the actual model to be fitted. For instance, if specified as formel = ~SEX different preference scale for the objects will be estimated for males and females. For two or more covariates, the operators + or * can be used to model main or interaction effects, respectively. The operator : is not allowed (redundant terms are removed automatically). See also formula.

The specification for elim follows the same rules as for formel. However, elim specifies the basic contingency table to be set up but does not specify any covariates to be fitted. This is done using formel. If, e.g., elim = ~SEX but formel = ~1, then the table is set up as if SEX would be fitted but only one global preference scale is computed. This feature allows for the successive fitting of nested models to enable the use of deviance differences for model selection (see example below).

Value

pattRrep.fit returns an object of class pattMod. The function print (i.e., print.pattMod) can be used to print the results and the function patt.worth to produce a matrix of worth parameters.

An object of class pattMod is a list containing the following components:

coefficients	estimates
ll	log-likelihood of the model
fl	log-likelihood of the saturated model
call	function call

`result` a list of results from the fitting routine (see Value of `nlm`).

`envList` a list with further fit details like subject covariates design structure `covdesmat`, paired comparison response pattern matrix `Y`, etc.

`partsList` a list of the basic data structures for each subgroup defined by crossing all covariate levels and different missing value patterns. Each element of `partsList` is again a list containing counts, missing value pattern, the CL matrix represented as the vector `s`, and the specification of the covariates. Use `str` to inspect the elements and see example below.

Input Data

The input data must have the following order (from left to right): all items at first time point, all items at second time point (with the same order as before), etc. for the other time points, optional subject covariates. The responses have to be coded as consecutive integers starting with 1 (or 0). The value of 1 (0) means highest 'endorsement' (agreement) according to the underlying scale. Missing values are coded as `NA`, rows with less than 1 valid response are removed from the fit and a message is printed.

Optional subject covariates have to be specified such that the categories are represented by consecutive integers starting with 1. Rows with missing values for subject covariates are removed from the data and a message is printed. Again, the leftmost columns in the data must be the rankings, optionally followed by columns for categorical subject covariates.

The data specified via `obj` are supplied using either a data frame or a datafile in which case `obj` is a path/filename. The input data file if specified must be a plain text file with variable names in the first row as readable via the command `read.table(datafilename, header = TRUE)`.

Warning

The size of the table to be analysed increases dramatically with the number of items J and time points T. For rankings the number of paired comparison response categories is always two. For each time point the number of rows of the table to set up the design matrix is initially $(J!$. Thus, the number of rows in the design matrix is $(J!)^T$. The number of combined covariate levels and the number of missing value patterns have effects only on the run time. A (reasonable) maximum number of items for two time points is 5 or 6, for three timepoints 4, and for four to seven timepoints 3.

Note

The number of timepoints can also be regarded as different response dimensions.

Author(s)

Reinhold Hatzinger

See Also

`pattL.fit`, `patt.design`, `pattPC.fit`, `pattR.fit`, `pattLrep.fit`

Examples

```
# simulated data: 3 items, 2 timepoints
dat1 <- simR(3, 100, c(.2, .7, .1))
dat2 <- simR(3, 100, c(.5, .4, .1))
dat  <- data.frame(dat1, dat2)
res  <- pattLrep.fit(dat, nitems = 3, tpoints = 2, iaT = TRUE)
res
patt.worth(res, obj.names = LETTERS[1:3])
```

`plot.wmat` *Method to plot worth or model parameters from LLBT or pattern models*

Description

A plot of the worth or model parameter matrix obtained from the fit of an LLBT or pattern model is produced. This matrix is obtained from `llbt.worth` or `patt.worth` and is an object of class `wmat`.

Usage

```
## S3 method for class 'wmat'
plot(x, main = "Preferences", ylab = "Estimate", psymb = NULL,
        pcol = NULL, ylim = range(worthmat), log = "", ...)
```

Arguments

`x`	worth or parameter matrix as generated from `llbt.worth` or `patt.worth`.
`main`	main title of the plot.
`ylab`	y-axis label
`psymb`	plotsymbols for objects, see Details below
`pcol`	colours for objects, see Details below
`ylim`	limits for y-axis
`log`	if specified as `log = "y"`, the y-axis is to be logarithmic
`...`	further graphical parameters, use e.g. `lty = "dashed"` to obtain dashed lines connecting the objects

Details

Plotsymbols can be defined as an integer vector of length equal to the number of objects, e.g., `psymb = c(15, 22, 18)`. They specify the graphical option `pch` as used in the `points` function. The default (`psymb = NULL`) uses the symbols 15 through 18 and 21 through 25. The number of symbols is determined from the number of rows in `worthmat`. A display of some plotsymbols may be obtained from the corresponding example below.

If `pcol = NULL`, the colours for objects are defined from the `rainbow_hcl` palette using the **colorspace** package. Other specifications include `"heat"`, `"terrain"` (see `rainbow_hcl`), and `"gray"`

(see grDevices). The number of different colours is automatically determined via the number of objects. Alternatively, pcol can be specified as a character vector containing user defined RGB colour values for all objects (as hexadecimal strings in the form "#rrggbb"), e.g., for blue "#0000FF"). These are usually set up using standard colour palettes (see rainbow or, e.g., the **RColorBrewer** package (see Examples below).

The old plot function, plotworth(), is defunct (see prefmod-defunct) and will generate errors. If you are still using it, please update your code!

Author(s)

Reinhold Hatzinger

See Also

patt.worth

Examples

```
# fit only first three objects with SEX effect
m2 <- pattPC.fit(cemspc, nitems = 3, formel = ~SEX, elim = ~SEX, undec = TRUE)

# calculate and plot worth parameters
m2worth <- patt.worth(m2)
plot(m2worth)
plot(m2worth, pcol = "terrain")

# display of some plotsymbols (pch)
plot(0:25, rep(1, 26), pch = 0:25, cex = 1.5)
text(0:25, rep(0.95, 26), 0:25)

# usage of the "RColorBrewer" package
## Not run:
library("RColorBrewer")
mypalette <- brewer.pal(3, "Set1")
plot(m2worth, pcol = mypalette)
## End(Not run)
```

---

prefmod-defunct *Defunct Functions in Package* **prefmod**

Description

A list of functions that are no longer part of **prefmod**.

Usage

```
plotworth(worthmat, main = "Preferences", ylab = "Estimate", psymb = NULL,
        pcol = NULL, ylim = range(worthmat), ...)
```

Arguments

worthmat	parameter matrix as generated from llbt.worth or patt.worth.
main	main title of the plot.
ylab	y-axis label
psymb	plotsymbols for objects, see Details below
pcol	colours for objects, see Details below
ylim	limits for y-axis
...	further graphical parameters, use e.g. log = "y" to obtain a logarithmic plot

Details

plotworth() was initially used to plot worth or model parameters from LLBT or pattern models (in a matrix created by llbt.worth() or patt.worth()). Now, the generic plot (i.e., plot.wmat()) has to be used.

Examples

```
## Not run:
###################
### plotworth() ###
###################

# fit only first three objects with SEX effect
m2 <- pattPC.fit(cemspc, nitems = 3, formel = ~SEX, elim = ~SEX, undec = TRUE)

# calculate and plot worth parameters
m2worth <- patt.worth(m2)
plot.wmat(m2worth)
plot.wmat(m2worth, pcol = "terrain")

# display of some plotsymbols (pch)
plot(0:25, rep(1, 26), pch = 0:25, cex = 1.5)
text(0:25, rep(0.95, 26), 0:25)

# usage of the "RColorBrewer" package
library("RColorBrewer")
mypalette <- brewer.pal(3, "Set1")
plot.wmat(m2worth, pcol = mypalette)
## End(Not run)
```

---

print.pattMod *Print methods for pattern models*

Description

Print method for objects of class pattMod.

Usage

```
## S3 method for class 'pattMod'
print(x, ...)
```

Arguments

x	Object of class pattMod.
...	Further arguments to be passed to or from other methods. They are ignored in this function.

Details

This print method generates output for fitted pattern models, i.e., for models of class pattMod. The functions pattPC.fit, pattR.fit, pattL.fit, and pattLrep.fit produce such objects.

Author(s)

Reinhold Hatzinger

Examples

```
res <- pattR.fit(salad, nitems = 4)
res
```

---

salad *Data (ranks): Salad Dressings (Critchlow and Fligner)*

Description

The dataset contains the rankings of four salad dressings concerning tartness by 32 judges, with values ranging from 1 (most tart) to 4 (least tart).

Usage

```
salad
```

Format

A data frame with 32 observations on 4 variables (A, B, C, D) each representing a different salad dressing.

References

Critchlow, D. E. & Fligner, M. A. (1991). Paired comparison, triple comparison, and ranking experiments as generalized linear models, and their implementation on GLIM. *Psychometrika 56*(3), 517–533.

Examples

```
# Example for object covariates
# fit object covariates:
# salads A - D have varying concentrations of acetic and gluconic acid.
# The four pairs of concentrations are
# A = (.5, 0), B = (.5, 10.0), C = (1.0, 0), and D = (0, 10.0),

conc <- matrix(c(.5, 0,  .5, 10,  1, 0,  0, 10), ncol = 2, byrow = TRUE)
sal  <- patt.design(salad, nitems = 4, resptype = "ranking")
X    <- as.matrix(sal[, 2:5])

glm(y ~ X, data = sal, family = poisson)
```

`simPC` *Utility function to simulate paired comparison or rankings data*

Description

The function generates a random paired comparison data matrix (two response categories, no undecided) or a rankings data matrix optionally based on user specified worth parameters.

Usage

```
simPC(nobj, nobs, worth = NULL, seed = NULL, pr = FALSE)
simR(nobj, nobs, worth = NULL, seed = NULL, pr = FALSE)
```

Arguments

`nobj`	Number of objects.
`nobs`	Number of cases.
`worth`	If `NULL`, values are drawn from uniform distribution (interval from 0 to 1), otherwise the user may specify arbitrary values. In both cases, the values are transformed into [0, 1] and scaled to sum up to 1.
`seed`	Starting value for the random number generator.
`pr`	If `TRUE`, the used worth parameters are printed.

Value

The random data matrix as a data frame.

Author(s)

Reinhold Hatzinger

Examples

```
data <- simPC(4, 10, worth = 1:4, seed = 123456)
data
```

summary.pattNPML *Summarizing finite mixture regression fits*

Description

These functions are the summary, print, and BIC methods for objects of type pattNPML.

Usage

```
## S3 method for class 'pattNPML'
summary(object, digits = max(3, getOption("digits") - 3), ...)
## S3 method for class 'pattNPML'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	a fitted object of class pattNPML.
x	a fitted object of class pattNPML.
digits	number of digits; applied on various displayed quantities.
...	further arguments, which will mostly be ignored.

Note

The summary and print methods are adapted versions from the **npmlreg** package.

See Also

pattnpml.fit

tennis *Data (paired comparisons): Preferred Interview Partner*

Description

The data describes results from a paired comparison study where 68 male and 96 female students were asked whom they would prefer to interview. The potential interview partners were Bonnie Blair, Jackie Joyner, and Jennifer Capriati.

Usage

```
tennis
```

Format

A data frame with 16 observations on the following 5 variables.

`n` counts of response pattern (`C1, C2, C3`)

`C1` Blair vs. Joyner: (1) Blair preferred, (-1) Joyner preferred

`C2` Blair vs. Capriati: (1) Blair preferred, (-1) Capriati preferred)

`C3` Joyner vs. Capriati: (1) Joyner preferred, (-1) Capriati preferred)

`SEX` a numeric vector: (1) male, (2) female

References

Böckenholt, U., & Dillon, W. R., (1997). Modeling within-subject dependencies in ordinal paired comparison data. *Psychometrika, 62*(3), 411–434.

Examples

```
tdat <- expand.mat(tennis[, -1], tennis[, 1])
head(tdat)
```

---

trdel *Data (paired comparisons): Training delivery modes*

Description

The dataset `trdel` contains data from a paired comparison study to investigate which of five training delivery modes trainees prefer (Schoell and Veith, 2011). The modes were computer-based (`CO`), TV-based (`TV`), paper-based (`PA`), audio-based (`AU`) and classroom-based (`CL`) training. Study participants were unemployed persons in the labour market training of the Austrian labour market service (AMS). To account for trainee characteristics that might affect the preference order the variables gender, age, and learning personality type were recorded. These variables were coded as `sex` (1 male, 2 female), `age` (numeric in years), `ltype` (1 accomodator, 2 diverger, 3 converger, 4 assimilator). The learning personality types were identified from a questionnaire.

Usage

```
trdel
```

Format

A data frame with 198 observations on the following 14 variables.

`V1`, `V2`, `V3`, `V4`, `V5`, `V6`, `V7`, `V8`, `V9`, `V10` paired comparisons in standard order: `CO:TV`, `CO:PA`, etc. 1 first object preferred, 2 second object preferred.

`ltype` learning types: (1) accomodator, (2) diverger, (3) converger, (4) assimilator

`age` numeric in years

`sex` (1) male, (2) female

Source

Schöll, B., Veith, S. (2011). Learning style evaluation and preferred training delivery modes in labour market training (in German). Master's thesis, Vienna University of Economics and Business.

Examples

```
head(trdel)
```

`xmpl` *Data (Likert items): Example Data Set*

Description

Data to illustrate the usage of `patt.design` for `rating scale (Likert type) items`.

Usage

```
xmpl
```

Format

A data frame with 100 observations on 5 numeric variables. The first three variables (`I1`, `I2`, `I3`) are the rating scale (Likert type) items with 5 response categories, ranging from 1 (strong agreement) to 5 (strong disagreement).

`I1` response to item 1

`I2` response to item 2

`I3` response to item 3

`SEX` (1) *male*, (2) *female*

`EDU` (1) *low education*, (2) *high education*

Details

Datasets in data files or Data frames used in `patt.design` require the following structure:

- All values must be numeric.
- The item responses must be in the leftmost columns (such as `I1` to `I3` above).
- Categorical subject covariates follow the item responses (rightmost columns) and their levels must be specified as consecutive integers. If in a used datafile or dataframe these are defined as **R** factors they will be converted to integers. This is not possible if characters are used as factor levels and, consequently, `patt.design` will produce an error.

Examples

```
des <- patt.design(xmpl, nitems = 3, resptype = "rating", cov.sel = "SEX")
head(des)
```

Index

∗ **datagen**
simPC, 48
∗ **datasets**
baseball, 3
carconf, 4
cemspc, 5
dat4, 8
euro55.2.des, 8
immig, 10
issp2000, 11
music, 21
salad, 47
tennis, 49
trdel, 50
xmpl, 51
∗ **models**
checkMIS, 7
llbt.design, 12
llbt.fit, 16
llbt.worth, 17
llbtPC.fit, 19
patt.design, 22
patt.worth, 26
pattL.fit, 27
pattLrep.fit, 29
pattnpml.fit, 32
pattPC.fit, 36
pattR.fit, 39
pattRrep.fit, 41
plot.wmat, 44
print.pattMod, 46
summary.pattNPML, 49
∗ **multivariate**
checkMIS, 7
llbtPC.fit, 19
pattL.fit, 27
pattLrep.fit, 29
pattPC.fit, 36
pattR.fit, 39
pattRrep.fit, 41
∗ **package**
prefmod-package, 2
∗ **regression**
llbt.design, 12
patt.design, 22
pattnpml.fit, 32
summary.pattNPML, 49
∗ **utilities**
expand.mat, 9

baseball, 3
BIC.pattNPML *(*summary.pattNPML*)*, 49

carconf, 4
cemspc, 5, *20*, *38*
checkMIS, 7, *38*

dat4, 8

euro55.2.des, 8
expand.mat, 9

factor, *13*
formula, *19*, *28*, *30*, *37*, *40*, *42*

glm, *20*, *35*
gnm, *17*
grDevices, *45*

immig, 10
issp2000, 11, *29*

llbt.design, *8*, 12, *17, 18*, *20*, *25*
llbt.fit, 16
llbt.worth, *15*, 17, *20*, *44*, *46*
llbtPC.fit, *15*, *17, 18*, 19

model.matrix, *13*, *22*
music, 21

nlm, *28*, *31*, *37*, *40*, *43*

patt.design, *9*, *15*, 22, *29*, *32*, *38*, *41*, *43*, *51*
patt.worth, 26, *28*, *31*, *37*, *40*, *42*, *44–46*
pattL.fit, *20*, *24–26*, 27, *27*, *32*, *38*, *40, 41*, *43*, *47*
pattLrep.fit, *26, 27*, 29, *43*, *47*
pattnpml.fit, 32, *49*
pattPC.fit, *7*, *24–27*, *29*, *32*, 36, *41*, *43*, *47*
pattR.fit, *24–27*, *29*, *32*, *38*, 39, *43*, *47*
pattRrep.fit, *32*, 41
plot, *17, 18*, *26, 27*, *46*
plot.wmat, 44, *46*
plotworth, *45, 46*
plotworth *(*prefmod-defunct*)*, 45
points, *44*
prefmod *(*prefmod-package*)*, 2
prefmod-defunct, 45
prefmod-package, 2
print, *28*, *31*, *37*, *40*, *42*
print.pattMod, *28*, *31*, *37*, *40*, *42*, 46
print.pattNPML *(*summary.pattNPML*)*, 49

rainbow_hcl, *44*

salad, *41*, 47
simPC, 48
simR *(*simPC*)*, 48
str, *28*, *31*, *37*, *40*, *43*
summary.pattNPML, 49

tennis, 49
trdel, 50

xmpl, *14*, *24*, 51