

Package ‘prefmod’

June 14, 2009

Type Package

Title Utilities to fit paired comparison models for preferences

Version 0.8-16

Date 2009-06-05

Author Reinhold Hatzinger

Maintainer Reinhold Hatzinger <reinhold.hatzinger@wu.ac.at>

Depends stats, gnm, colorspace

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).

License GPL

Repository CRAN

Date/Publication 2009-06-14 19:35:43

R topics documented:

prefmod-package	2
baseball	3
cemspc	4
checkMIS	5
dat4	6
expand.mat	7
issp2000	7
llbt.design	9
llbt.fit	12
llbt.worth	13
llbtPC.fit	14

music	16
patt.design	17
patt.worth	21
pattL.fit	22
pattLrep.fit	25
pattPC.fit	27
pattR.fit	30
plotworth	32
print.pattMod	34
salad	35
tennis	36
xmpl	36

Index	38
--------------	-----------

prefmod-package *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).

Details

Package: prefmod
 Type: Package
 Version: 0.8-16
 Date: 2008-06-05
 Depends: stats
 License: GPL

Author(s)

Reinhold Hatzinger

Maintainer: Reinhold Hatzinger <reinhold.hatzinger@wu-wien.ac.at>

Examples

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

data(xmpl) # example data in package
```

```
dsgnmat <- patt.design(xmpl, nitems = 3, resptype="rating",
  blnIntcovs = TRUE, cov.sel="ALL")
print(head(dsgnmat))

## fit of Critchlov & Fligner (1991) Salad Dressings Data
data(salad)
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,family=poisson,data=sal)
```

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

```
data(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

```
data(baseball)
str(baseball)
```

cemspc

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

Description

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

Usage

```
data(cemspc)
```

Format

A data frame with 303 observations on the following 17 variables.

- V1** comparison of London to Paris
- V2** comparison of London to Milano
- V3** comparison of Paris to Milano
- V4** comparison of London to St.Gallen
- V5** comparison of Paris to St.Gallen
- V6** comparison of Milano to St.Gallen
- V7** comparison of London to Barcelona
- V8** comparison of Paris to Barcelona
- V9** comparison of Milano to Barcelona
- V10** comparison of St.Gallen to Barcelona
- V11** comparison of London to Stockholm
- V12** comparison of Paris to Stockholm
- V13** comparison of Milano to Stockholm
- V14** comparison of St.Gallen to Stockholm
- V15** comparison of Barcelona to Stockholm
- ENG** Knowledge of English : 1= good, 2= poor
- SEX** Gender : 1= female, 2= male

References

R. Dittrich, R.Hatzinger and W.Katzenbeisser: Modelling the effect of subject-specific covariates in paired comparison studies with an application to university rankings. Applied Statistics, Vol. 47 (1998), Part 4

Examples

```
data(cemspc)
par(mfrow=c(4,4))
for (i in 1:15) barplot(table(cemspc[,i]))
par(mfrow=c(1,1))
```

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, nitens, verbose = FALSE)
```

Arguments

obj	dataframe or datafile path/name (like pattPC.fit).
nitens	the number of compared objects, not the number of comparisons (like pattPC.fit).
verbose	if TRUE printed output, otherwise only invisible output to be used, e.g., in the specification of MISalpha in pattPC.fit .

Value

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

See Also

[pattPC.fit](#)

Examples

```
## no missing NAs in dataset dat4

data(dat4)
checkMIS(dat4, nitens=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 accomodated by using checkMIS

pattPC.fit(data3, nitems=3, MISalpha=checkMIS(data3,nitems=3))

```

dat4

Data (paired comparisons): dat4

Description

A fictitious data set with 100 observations on 6 paired comparisons. The responses are 1 assigned if the first object in a comparison is the preferred one and -1 if it is not preferred. For the ordering of objects and comparisons see [llbt.design](#).

Usage

```
data(dat4)
```

Format

A data frame with 100 observations on 6 comparisons:

comp1 responses to first comparison
comp2 responses to second comparison
comp3 responses to third comparison
comp4 responses to fourth comparison
comp5 responses to fifth comparison
comp6 responses to sixth comparison

Examples

```

data(dat4)
str(dat4)
# to get a general idea we use the histogram plot
par(mfrow=c(2,3))
for (i in 1:6) barplot(table(dat4[,i]))
par(mfrow=c(1,1))

```

`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

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

Value

the expanded matrix

Note

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

Examples

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

`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 Likert type response scale) where respondents from Austria and Great Britain were asked about their perception of environmental dangers.

Usage

```
data(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 Likert type scale 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/matric*, (2) *A-level/matric or higher*

Source

Dataset: International Social Survey Programme 2000: Environment II (ISSP 2000)

Identification Number: 3440

Central Archive for Empirical Social Research

http://www.gesis.org/en/data_service/issp/index.htm

Usage regulations:

http://www.gesis.org/en/data_service/order/usage_regulations.htm

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

```
data(issp2000)
str(issp2000)
```

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. Optionally, the function provides all necessary structures (commands, data/design files) to fit the loglinear paired comparisons pattern model in GLIM, which is often more efficient at fitting large loglinear models of this type.

Usage

```
llbt.design(obj, nitems = NULL, objnames = "",
            blnCasewise = FALSE, cov.sel = "",
            blnGLIMcmds = FALSE, glimCmdFile = "", outFile = "")
```

Arguments

<code>obj</code>	either a data frame, a data file name, or a control object.
<code>nitems</code>	number of items (objects).
<code>objnames</code>	an optional character vector with names for the objects. These names are the columns names in the output data frame. If <code>objnames</code> is not specified <code>o1,o2,</code> etc. will be used.
<code>blnCasewise</code>	If <code>blnCasewise = TRUE</code> 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.
<code>cov.sel</code>	a character vector with the names of the subject covariates in the data file to be included into the design matrix. (example: <code>cov.sel = c("SEX", "AGE")</code>). If all covariates are to be included the specification can be abbreviated to <code>cov.sel = "ALL"</code> . For no covariates specify <code>cov.sel = ""</code> .
<code>blnGLIMcmds</code>	TRUE, if GLIM output is wanted. If <code>blnGLIMcmds = FALSE</code> the following items can be set to any value (such as a null text string) and are ignored. Please note that if <code>blnGLIMcmds</code> is set to be TRUE there is no output in R but instead goes to the the following files.
<code>glimCmdFile</code>	name of the output file which will contain all necessary commands to fit a basic model (defining all structures and reading the necessary data).
<code>outFile</code>	name of the data/design file to be read into GLIM. It consists of the response frequencies and the covariates for the objects, the undecided comparison responses and the subject effects.

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 – ... – \emph{not at all preferred}*). In both cases an undecided category may or may not occur. If there are more than three categories 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
- **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. Prior to fit a model, the categorical subject covariates have to be declared as `factor(s)`. See example below.
- **GLIM output.** If the user specifies `blnGLIMcmds = TRUE` two files are generated one of which contains all GLIM commands to fit a basic loglinear paired comparisons pattern model. The other contains the design matrix optionally including subject covariates.

Value

The output is a dataframe. 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 `blnCasewise = 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.`) and for the items. If subject covariates are present they are in the rightmost columns.

Alternatively, the function `llbt.design` does not produce visible output in R if GLIM output is requested via `blnGLIMcmds = TRUE`. The output is then written to the corresponding files.

Input Data

Responses have to be coded as consecutive integers (e.g., (0,1), or (1,2,3,...)), where the lowest value corresponds to (highest) preference for the first object in a comparison. For paired comparison without undecided category (-1,1), or (1,0,-1) for paired comparison with an undecided category, can also be used ('-1' is the not preferred category). 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](#) or the file `xmpl.dat` in the package's `data/` directory.

(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 with slightly different specification is given in [patt.design](#).)

Author(s)

Reinhold Hatzinger

References

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

See Also

[patt.design](#), [pattPC.fit](#), [pattL.fit](#), [pattR.fit](#)

Examples

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

eng<-factor(des$ENG)
res<-gnm(y~o1+o2+o3+o4+o5+o6+eng:(o1+o2+o3+o4+o5+o6)+g1,
        eliminate=mu:eng, family=poisson, data=des)
summary(res)

## calculating and plotting worth parameters
lambda<-res$coefficients[c(2:6,23:27)]
lambda[6:10]<-lambda[1:5]+lambda[6:10]
lambmat<-matrix(lambda,ncol=2)
lambmat<-rbind(lambmat,c(0,0))
worthmat<-apply(lambmat,2,
               function(x) exp(2*x)/sum(exp(2*x)))
colnames(worthmat)<-c("English:good","English:poor")
rownames(worthmat)<-c("London", "Paris", "Milano",
                    "St.Gallen", "Barcelona","Stockholm")
plotworth(worthmat)
```

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	nmaximum 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

Reinhold Hatzinger, Brian J. Francis: Fitting Paired Comparison Models in R. (http://epub.wu-wien.ac.at/dyn/openURL?id=oai:epub.wu-wien.ac.at:epub-wu-01_709)

Examples

```
## fit basic model casewise
data(cemspc)
mfr<-llbt.design(cemspc,nitems=6,objnames=c("lo","pa","mi","sg","ba","st"),
  blnCasewise=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

data(cemspc)
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	<i>Function to calculate and print worth parameters from LLBT model results</i>
------------	---

Description

Worth parameter are calculated from the results of an LLBT model fit, i.e., from `llbtPC.fit`. The resulting estimates for all groups are based on the subject covariates as specified in the model formula (`formel`) of the LLBT model.

Usage

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

Arguments

<code>obj</code>	Object of class <code>llbtMod</code> obtained from an LLBT model fit, i.e., <code>llbtPC.fit</code> .
<code>obj.names</code>	names for the objects. If <code>NULL</code> , the names as specified in <code>obj</code> are used.
<code>outmat</code>	a matrix of estimated worth parameters (<code>outmat = "worth"</code> , the default) or LLBT model parameters (<code>outmat = "lambda"</code>).

Value

`llbt.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 `plotworth` gives a plot of the estimates.

Author(s)

Reinhold Hatzinger

See Also

`llbtPC.fit`, `plotworth`

Examples

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

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

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 = FALSE)
```

Arguments

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

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` or the file `cemspc.dat` in the package's `data/` directory.

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
data(cemspc)
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 (General social survey)

Description

The General Social Survey has been conducted by the National Opinion Research Center in 1993. The dataset focusses on 11 items with a 5-point Likert type response scale. The respondents were asked about their music preferences.

Usage

```
data(music)
```

Format

A data frame with 1500 observations on the following 11 variables. The variables are items with the question for liking bigband/ blugrass,.. very much, like it, have mixed feelings, dislike it and dislike it very much. Missing values are categorized as NA, so it is not a numeric data frame.

bigband responses to bigband item

blugrass responses to blugrass item

country responses to country item

blues responses to blues item

musicals responses to musicals item

klassik responses to klassik item

folk responses to folk item

jazz responses to jazz item

opern responses to opern item

rap responses to rap item

hvy metal responses to hvymetal item

Details

The 11 variables are items to be answered on a 5-point Likert type scale with response categories: (1) like it very much, (2) like it, (3) have mixed feelings, (4) dislike it and (5) dislike it very much.

Source

Dataset: General Social Survey 1993 [United States]
 CPANDA Identification Number: a00006
 Cultural Policy and the Arts National Data Archive
<http://www.cpanda.org/cpanda/getDDI.xq?studyID=a00006#dataDscr>

References

This is just an abstract for further information concerning the General Social Survey dataset have a look at the Cultural Policy and the Arts National Data Archive.

Examples

```
data(music)
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 (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. Optionally, the function provides all necessary structures (commands, data/design files) to fit the loglinear paired comparisons pattern model in GLIM, which is often more efficient at fitting large loglinear models of this type.

Usage

```
patt.design(obj, nitens = NULL, objnames = "", resptype = "paircomp",
            blnRevert = FALSE, cov.sel = "", blnIntcovs = FALSE,
            blnGLIMcmds = FALSE, glimCmdFile = "", outFile = "", intFile = "")
```

Arguments

<code>obj</code>	either a data frame, a data file name, or a control object.
<code>nitens</code>	number of items (objects). <code>nitens</code> is not the number of comparisons!
<code>objnames</code>	an optional character vector with names for the objects These names are the columns names in the output data frame. If <code>objnames</code> is not specified consecutive letters starting with "a" will be used.
<code>resptype</code>	one of "paircomp", "rating", or "ranking".

<code>blnRevert</code>	If the responses are such that low values correspond to high preference or agreement and high values to low preference or agreement (e.g., (1) <i>I strongly agree</i> ... (5) <i>I strongly disagree</i>) then <code>blnRevert</code> should be specified to be <code>FALSE</code> . Otherwise set <code>blnRevert = TRUE</code> .
<code>cov.sel</code>	a character vector with the names of the subject covariates in the data file to be included into the design matrix. (example: <code>cov.sel = c("SEX", "AGE")</code>). If all covariates are to be included the specification can be abbreviated to <code>cov.sel = "ALL"</code> . For no covariates specify <code>cov.sel = ""</code> .
<code>blnIntcovs</code>	generates covariates for interactions between comparisons if <code>blnIntcovs = TRUE</code> .
<code>blnGLIMcmds</code>	<code>TRUE</code> , if <code>GLIM</code> output is wanted. If <code>blnGLIMcmds = FALSE</code> the following items can be set to any value (such as a null text string) and are ignored. Please note that if <code>blnGLIMcmds</code> is set to be <code>TRUE</code> there is no output in <code>R</code> but instead goes to the the following files.
<code>glimCmdFile</code>	name of the output file which will contain all necessary commands to fit a basic model (defining all structures and reading the necessary data).
<code>outFile</code>	name of the data/design file to be read into <code>GLIM</code> . It consists of the response frequencies and the covariates for the objects, the undecided comparison responses and the subject effects.
<code>intFile</code>	name of the design file for the interaction effects. This file is only generated if <code>blnIntcovs = TRUE</code> .

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 – ... – \emph{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.
- **item 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 dependency between two comparisons that have one item in common) can be obtained by setting `blnIntcovs = TRUE`.

- **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.
- **GLIM output.** If the user specifies `blnGLIMcmds = TRUE` two files are generated one of which contains all GLIM commands to fit a basic loglinear paired comparisons pattern model. The other contains the design matrix optionally including subject covariates. If dependency covariates are requested they are written to a third file. (Please note that the corresponding part of design matrix is transposed in the interactions output file to allow GLIM for using the `$array` facility in case of a large number of parameters to be estimated)

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.

Alternatively, the function `patt.design` does not produce output in R if GLIM output is requested via `blnGLIMcmds = TRUE`. The output is then written to the corresponding files (see **The Control List** below).

Input Data

Responses have to be coded as consecutive integers (e.g., (0,1), or (1,2,3,...)). For paired comparison without undecided category (-1,1), or (1,0,-1) for paired comparison with an undecided category, can also be used ('-1' is the not preferred category).

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 categorical subject covariates. These have to be specified such that the categories are represented by consecutive integers starting with 1. Missing values are not allowed (use functions `pattPC.fit`, `pattL.fit`, or `pattR.fit` instead), and treated such that rows with NAs are removed from the resulting design structure and a message is printed. For an example see `xmpl` or the file `xmpl.dat` in the package’s `data/` directory.

(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.)

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

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

## ILLUSTRATING THE ISSP2000 EXAMPLE
## simplified version of the analysis as
## given in Dittrich et.al.(2007)

data(issp2000)

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,
           family=poisson, data=design))

# 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,
           family=poisson, data=design))

## Not run:
## example with control list
## not run because input data file does not exist

# defining the ctrl list
# would be typically read from file using source()
testex1<-list(
  resptype      = "paircomp",
  datafile      = "test/test.dat",
  nitens        = 5,
  blnRevert     = FALSE,
  blnReducecat  = TRUE,
  blnIntcovs    = FALSE,
  cov.sel       = c("SEX", "URB"),

  blnGLIMcmds   = TRUE,
  glimCmdFile   = "test/test.gli",
  outFile       = "test/test.design",
  intFile       = "" # since blnIntcovs = FALSE
)
patt.design(testex1)
## End(Not run)

```

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](#). The resulting estimates for all groups are based on the subject covariates as specified in the model formula (`formel`) of a pattern model.

Usage

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

Arguments

obj	Object of class <code>pattMod</code> 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 (<code>outmat = "worth"</code> , the default) or pattern model parameters (<code>outmat = "lambda"</code>).

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 `plotworth` gives a plot of the estimates.

Author(s)

Reinhold Hatzinger

See Also

[pattPC.fit](#), [pattR.fit](#), [pattL.fit](#), and [pattLrep.fit](#), [plotworth](#)

Examples

```
## fit only first three objects with SEX effect
data(cemspc)
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",
  undec = FALSE, ia = FALSE, Nitest = FALSE, pr.it = FALSE)
```

Arguments

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

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

`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:

	main results of the fit like estimates (<code>coefficients</code>), log likelihood of the model (<code>ll</code>), log likelihood of the saturated model (<code>fl</code>), and the call
<code>result</code>	a list of results from the fitting routine (see Value of <code>nlm</code>).
<code>envList</code>	a list with further fit details like subject covariates design structure <code>covdesmat</code> , paired comparison response pattern matrix <code>Y</code> , 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` or the file `issp2000.dat` in the package’s `data/` directory.

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 $(2 * \text{numberofcategories} - 1)^{\text{numberofitems}}$, 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
data(music)
music4<-music[,c("jazz", "blues", "folk", "rap")]
pattL.fit(music4, nitems=4)

## fit additional undecided effect
pattL.fit(music4, nitems=4, undec=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", undec = FALSE, 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 "rating" by default and is reserved for future usage. Any other specification will not change the behaviour of pattL.fit
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 ia = 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:

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

For an example see [issp2000](#) or the file `issp2000.dat` in the package’s `data/` directory.

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

$(2 * \text{numberofcategories} - 1)^{\text{numberofitems}}$). After reducing to three categories the number of patterns are 13, 75, 541 for 3 to 5 items. Generally, the number of rows in the design matrix is $(\text{numberofpatterns}^{\text{numberoftimepoints}}) * \text{numberof}(\text{combinedcovariatelevels})$. A (reasonable) maximum number of items for two time points is 5, for three timepoints 4, and for four to five timepoints 3.

Author(s)

Reinhold Hatzinger

See Also

[pattL.fit](#), [patt.design](#), [pattPC.fit](#), [pattR.fit](#)

Examples

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

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 = FALSE, ia = FALSE, NItest = FALSE,
  NI = FALSE, MISalpha = NULL, MIScommon = FALSE, 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 <code>pattPC.fit</code>
obj.names	character vector with names for objects.
undec	for paired comparisons with a undecided/neutral category, a common parameter will be estimated if <code>undec = TRUE</code> .
ia	interaction parameters between comparisons that have one object in common if <code>ia = TRUE</code> .
NItest	separate estimation of object parameters for complete and incomplete patterns if <code>NItest = TRUE</code> . Currently, <code>NItest</code> is set to <code>FALSE</code> if subject covariates are specified.
NI	if <code>TRUE</code> , fits large table (crossclassification with NA patterns), for comparison with models including <code>MISalpha</code> (and <code>MISbeta</code>).
MISalpha	if not <code>NULL</code> , specification to fit parameters for NA indicators using a logical vector, where <code>TRUE</code> means that the NA indicator parameter for the corresponding object should be estimated (see example below). Currently each comparison is reparameterized with $\alpha_i + \alpha_j$.
MIScommon	if <code>TRUE</code> , fits a common parameter for NA indicators, i.e., $\alpha = \alpha_i = \alpha_j = \dots$
MISbeta	if not <code>NULL</code> , fits parameters for MNAR model, i.e., interactions between outcome model object parameters and NA indicator parameters. Currently each comparison is reparameterized with $\beta_i + \beta_j$. The specification is the same as for <code>MISalpha</code> (see example below). Usually, the specification for <code>MISbeta</code> is the same as for <code>MISalpha</code> , but any subset is reasonable. If <code>MISalpha = NULL</code> but <code>MISbeta</code> is not, then <code>MISalpha</code> is set to <code>MISbeta</code> .
pr.it	a dot is printed at each iteration cycle if set to <code>TRUE</code>

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:

main results of the fit like estimates (`coefficients`), log likelihood of the model (`ll`), log likelihood of the saturated model (`fl`), and the call

result	a list of results from the fitting routine (see Value of <code>nlm</code>).
envList	a list with further fit details like subject covariates design structure <code>covdesmat</code> , paired comparison response pattern matrix <code>Y</code> , 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 <code>partsList</code> is again a list containing counts, missing value pattern, the CL matrix represented as a vector, and the specification of the covariates. Use <code>str</code> 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` or the file `cemspc.dat` in the package's `data/` directory.

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
data(cemspc)
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 is",lr,"on df =",df," ( p =",round(1-pchisq(lr,df),digits=4),")\n")

## generates data set with three items and some missing values
## in comparison (23), column 3, then there are no NAs for
## object 1
data(dat4)
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",
          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.

<code>resptype</code>	is "ranking" by default and is reserved for future usage. Any other specification will not change the behaviour of <code>pattR.fit</code>
<code>ia</code>	interaction parameters between comparisons that have one object in common if <code>ia = TRUE</code> .
<code>NItest</code>	separate estimation of object parameters for complete and incomplete patterns if <code>NItest = TRUE</code> . Currently, <code>NItest</code> is set to <code>FALSE</code> if subject covariates are specified.
<code>pr.it</code>	a dot is printed at each iteration cycle if set to <code>TRUE</code>

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:

	main results of the fit like estimates (<code>coefficients</code>), log likelihood of the model (<code>ll</code>), log likelihood of the saturated model (<code>fl</code>), and the call
<code>result</code>	a list of results from the fitting routine (see Value of nlm).
<code>envList</code>	a list with further fit details like subject covariates design structure <code>covdesmat</code> , paired comparison response pattern matrix <code>Y</code> , etc.
<code>partsList</code>	a list of the basic data structures for each subgroup defined by crossing all covariate levels and different missing value patterns. Each element of <code>partsList</code> is again a list containing counts, missing value pattern, the CL matrix represented as a vector, and the specification of the covariates. Use <code>str</code> 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](#) or the file `salad.dat` in the package's `data/` directory.

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

data(salad)
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,family=poisson,data=sal)
```

plotworth

Function 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. The parameter matrix is produced by `llbt.worth` or `patt.worth`.

Usage

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

Arguments

worthmat	parameter matrix as generated from <code>llbt.worth</code> or <code>patt.worth</code> . Alternatively, the contents of a user defined matrix can also be plotted (for an example see <code>llbt.design</code> , provided that the rows correspond to objects and the columns to groups).
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

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 `colspace` 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)).

Author(s)

Reinhold Hatzinger

See Also

[patt.worth](#)

Examples

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

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

```
plotworth(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")
plotworth(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

<code>x</code>	Object of class <code>pattMod</code> .
<code>...</code>	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

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

`salad`*Data (ranks): Salad Dressings (Crichtlow and Fligner)*

Description

The data frame shows the ranking of four salad dressing concerning tartness by 32 judges, whereas (1) is assigned as most tart and (4) is assigned as least tart.

Usage

```
data(salad)
```

Format

- A** salad dressing preparation A
- B** salad dressing preparation B
- C** salad dressing preparation C
- D** salad dressing preparation D

References

Douglas E. Crichtlow and Michael A. Fligner (1991), Paired Comparison, Triple Comparison, and Ranking Experiments As Generalized Linear Models, and Their Implementation on GLIM. *Psychometrika* Vol. 56, No. 3

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),
data(salad)
conc<-matrix(c(.5,0, .5,10, 1,0, 0,10), nc=2, byrow=TRUE)
sal<-patt.design(salad,nitems=4,resptype="ranking")
X<-as.matrix(sal[,2:5])
glm(y~X,family=poisson,data=sal)
```

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

```
data(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. and Dillon, W.R., Modelling within-subject dependencies in ordinal paired comparison data, *Psychometrika*, 1997, 62, p.411-434

Examples

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

xmpl

Data (Likert items): Example Data Set

Description

Data to illustrate the usage of `patt.design` for Likert items.

Usage

```
data(xmpl)
```

Format

A data frame with 100 observations on the following 5 variables.

I1 responses to first Likert item, (1) *strong agreement*, to (5) *strong disagreement*

I2 responses to second Likert item, like I1

I3 responses to third Likert item, like I1

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

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

All values are numeric.

Details

Datasets and/or dataframes used in `patt.design` are required to have 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

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

Index

*Topic **datasets**

baseball, 2
cemspc, 3
dat4, 5
issp2000, 7
music, 15
salad, 34
tennis, 35
xmpl, 35

*Topic **models**

checkMIS, 4
llbt.design, 8
llbt.fit, 11
llbt.worth, 12
llbtPC.fit, 13
patt.design, 16
patt.worth, 20
pattL.fit, 21
pattLrep.fit, 24
pattPC.fit, 26
pattR.fit, 29
plotworth, 31
print.pattMod, 33

*Topic **multivariate**

checkMIS, 4
llbtPC.fit, 13
pattL.fit, 21
pattLrep.fit, 24
pattPC.fit, 26
pattR.fit, 29

*Topic **package**

prefmod-package, 1

*Topic **regression**

llbt.design, 8
patt.design, 16

*Topic **utilities**

expand.mat, 6

baseball, 2

cemspc, 3, 14, 28

checkMIS, 4, 28

dat4, 5

expand.mat, 6

factor, 9

formula, 13, 22, 25, 27, 30

glm, 14

grDevices, 32

issp2000, 7, 23, 25

llbt.design, 5, 8, 14, 19, 32

llbt.fit, 11

llbt.worth, 12, 14, 32

llbtPC.fit, 12, 13

music, 15

nlm, 22, 25, 28, 30

patt.design, 10, 16, 23, 26, 28, 31, 35, 36

patt.worth, 20, 22, 25, 27, 30, 32

pattL.fit, 10, 14, 19, 20, 21, 21, 26, 28,
30, 31, 33

pattLrep.fit, 20, 21, 24, 33

pattPC.fit, 4, 5, 10, 19–21, 23, 26, 26, 31,
33

pattR.fit, 10, 19–21, 23, 26, 28, 29, 33

plotworth, 12, 21, 31

points, 32

prefmod (*prefmod-package*), 1

prefmod-package, 1

print, 22, 25, 27, 30

print.pattMod, 22, 25, 27, 30, 33

rainbow_hcl, 32

salad, 31, 34

str, 23, 25, 28, 30

tennis, 35

xmpl, 10, 19, 35