Package 'mvord'

October 3, 2018

Title Multivariate Ordinal Regression Models

Version 0.3.2

Date 2018-10-03

Author Rainer Hirk [aut, cre],
Kurt Hornik [aut],
Laura Vana [aut],
Alan Genz [ctb] (Fortran Code)

Maintainer Rainer Hirk <rhirk@wu.ac.at>

Description A flexible framework for fitting multivariate
ordinal regression models with composite likelihood methods.

License GPL-3

Depends minqa, BB, ucminf, dfoptim, R (>= 3.1.0)

Imports MASS, pbivnorm, stats, optimx, mnormt, numDeriv, Matrix

Suggests knitr

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 6.0.1.9000

Repository CRAN

Date/Publication 2018-10-03 10:00:03 UTC

R topics documented:

`mvord-package` *Multivariate Ordinal Regression Models in R.*

Description

The R package mvord implements composite likelihood estimation in the class of multivariate ordinal regression models with probit and logit link. A flexible modeling framework for multiple ordinal measurements on the same subject is set up, which takes into consideration the dependence among the multiple observations by employing different error structures. Heterogeneity in the error structure across the subjects can be accounted for by the package, which allows for covariate dependent error structures. In addition, regression coefficients and threshold parameters are varying across the multiple response dimensions in the default implementation. However, constraints can be defined by the user if a reduction of the parameter space is desired.

Details

see `mvord`

coef.mvord *Coefficients of Multivariate Ordinal Regression Models.*

Description

coef is a generic function which extracts regression coefficients from objects of class 'mvord'.

Usage

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

Arguments

object	an object of class 'mvord'.
...	further arguments passed to or from other methods.

constraints *Constraints on the Regression Coefficients of Multivariate Ordinal Regression Models.*

Description

An extractor function for the constraint matrices corresponding to the regression coefficients from objects of class 'mvord'.

Usage

```
constraints(object)

## S3 method for class 'mvord'
constraints(object)
```

Arguments

object	an object of class 'mvord'.

data_cr *Simulated credit ratings*

Description

A data set containing simulated credit ratings and simulated perfomance measures from four raters.

Usage

```
data(data_cr)
```

Format

A data frame with 690 rows and 11 columns

Details

- `rater1` credit ratings assigned by rater 1
- `rater2` credit ratings assigned by rater 2
- `rater3` credit ratings assigned by rater 3
- `rater4` credit ratings assigned by rater 4
- `firm_id` firm index
- `rater_id` rater index covered from the free operating cash-flow of a company
- `LR` liquidity ratio, relating the cash held by a company to the current liabilities
- `LEV` leverage ratio relating debt to earnings before interest and taxes
- `PR` profitability ratio of retained earnoings to assets
- `RSIZE` log of relative size of the company in the market
- `BETA` a measure of systematic risk

data_cr_panel *Simulated panel of credit ratings*

Description

A data set containing simulated credit ratings assigned by one rater and simulated perfomance measures for firms in different years.

- `rating` credit ratings
- `firm_id` firm index
- `year` year index
- `LR` liquidity ratio, relating the cash held by a company to the current liabilities

- LEV leverage ratio relating debt to earnings before interest and taxes
- PR profitability ratio of retained earnings to assets
- RSIZE log of relative size of the company in the market
- BETA a measure of systematic risk
- BSEC business sector of a firm (factor with 8 levels)

Usage

```
data(data_cr_panel)
```

Format

A data frame with 11320 rows and 9 variables

`data_mvord` *Simulated credit ratings*

Description

A simulated data set where three different raters (`rater1, rater2` and `rater3`) assign ordinal ratings on different firms. `rater3` uses a different rating scale compared to `rater1` and `rater2`, i.e., the number of threshold categories is different. For each firm we simulate five different covariates `X1, ..., X5` from a standard normal distribution. Additionally, each firm is randomly assigned to a business sector (sector `X`, `Y` or `Z`), captured by the covariate `X6`. Furthermore, we simulate multivariate normally distributed errors. For a given set of parameters we obtain the three rating variables for each firm by slotting the latent scores according to the corresponding threshold parameters. The IDs for each subject i of the $n = 1000$ firms are stored in the column `firm_id`. The IDs of the raters are stored in the column `rater_id`. The ordinal ratings are provided in the column `rating` and all the covariates in the remaining columns. Overall, the data set has 3000 rows, for each of the $n = 1000$ firms it has three rating observations.

Usage

```
data(data_mvord)
```

Format

A data frame with 3000 rows and 9 variables

Details

- `firm_id` firm index
- `rater_id` rater index
- `rating` ordinal credit ratings
- `X1` covariate X1

- X2 covariate X2
- X3 covariate X3
- X4 covariate X4
- X5 covariate X5
- X6 covariate X6 (factor)

---

data_mvord2 *Simulated credit ratings*

Description

A simulated data set where three different raters (rater1, rater2 and rater3) assign ordinal ratings on different firms. rater3 uses a different rating scale compared to rater1 and rater2. The IDs for each subject i of the $n = 1000$ firms are stored in the column firm_id.

Usage

```
data(data_mvord2)
```

Format

A data frame with 1000 rows and 10 variables

Details

- firm_id firm index
- rater1 ordinal rating outcome of rater 1
- rater2 ordinal rating outcome of rater 2
- rater3 ordinal rating outcome of rater 3
- X1 covariate X1
- X2 covariate X2
- X3 covariate X3
- X4 covariate X4
- X5 covariate X5
- X6 covariate X6 (factor)

data_mvord_panel *Simulated panel of credit ratings*

Description

A simulated data set where one rater assigns ratings over the years 2001 to 2010 for a set of firms. The IDs for each subject i of the $n = 1000$ firms are stored in the column `firm_id`. The year of the rating observation is stored in the column `year`. The ordinal ratings are provided in the column `rating` and all the covariates in the remaining columns.

Usage

```
data(data_mvord_panel)
```

Format

A data frame with 10000 rows and 9 variables

Details

- `firm_id` firm index
- `year` year index (2001 - 2010)
- `rating` ordinal credit ratings
- `X1` covariate X1
- `X2` covariate X2
- `X3` covariate X3
- `X4` covariate X4
- `X5` covariate X5
- `X6` covariate X6 (factor)

data_mvord_toy *Data set toy example*

Description

A data set containing two simulated ordinal responses with three categories, two quantitative covariates `X1` and `X2` and two categorical covariates `f1` and `f2`.

Usage

```
data(data_mvord_toy)
```

Format

A data frame with 100 rows and 6 variables

Details

- `Y1` ordinal outcome `Y1` (three categories)
- `Y2` ordinal outcome `Y2` (three categories)
- `X1` covariate `X1`
- `X2` covariate `X2`
- `f1` categorical covariate `f1`
- `f2` categorical covariate `f2`

`error_struct` *Error Structures in mvord*

Description

Different `error.structures` are available in **mvord**:

- general correlation structure (default) `cor_general(~ 1)`,
- general covariance structure `cov_general(~ 1)`,
- factor dependent correlation structure `cor_general(~ f)`,
- factor dependent covariance structure `cov_general(~ f)`,
- equicorrelation structure `cor_equi(~ 1)`,
- covariate dependent equicorrelation structure `cor_equi(~ S)`,
- AR(1) correlation structure `cor_ar1(~ 1)`, or
- covariate dependent AR(1) correlation structure `cor_ar1(~ S)`.

For more details see vignette.

Usage

```
cov_general(formula = ~1)

cor_general(formula = ~1)

cor_ar1(formula = ~1)

cor_equi(formula = ~1)

cor_ident(formula = ~1)
```

Arguments

`formula`	`formula` object

error_structure *Extracts Error Structure of Multivariate Ordinal Regression Models.*

Description

A generic function which extracts for each subject the estimated error structure parameters from objects of class 'mvord'.

Usage

```
error_structure(object, type, ...)

## S3 method for class 'mvord'
error_structure(object, type = NULL, ...)
```

Arguments

object	an object of class 'mvord'.
type	choose type "sigmas", "alpha", "corr", or "z".
...	further arguments passed to or from other methods.

Details

- sigmas extracts the correlation/covariance matrices corresponding to each subject. Applicable in line with cor_general, cov_general, cor_equi, cor_ar1.
- alpha extracts the parameters of the covariate dependent error structure. Applicable in line with cor_equi, cor_ar1.
- corr extracts the subject-specific correlation parameters. Applicable in line with cor_equi, cor_ar1.
- z extracts the subject-specific Fisher-z score. Applicable in line with cor_equi, cor_ar1.

fitted.mvord *Fitted Probabilities of Multivariate Ordinal Regression Models.*

Description

A generic function which extracts fitted probabilities for the observed categories from objects of class 'mvord'.

Usage

```
## S3 method for class 'mvord'
fitted(object, ...)
```

Arguments

object	an object of class 'mvord'.
...	further arguments passed to or from other methods.

---

joint_probabilities *Extracts fitted Probabilities for Multivariate Ordinal Regression Models.*

Description

Extracts fitted probabilities for given response categories from a fitted model of class 'mvord'.

Usage

```
joint_probabilities(object, response.cat, newdata = NULL, type = "prob",
  subjectID = NULL, newoffset = NULL, ...)
```

Arguments

object	an object of class 'mvord'.
response.cat	vector or matrix with response categories (for each subject one row of length equal to the number of multiple measurements).
newdata	(optional) data frame of new covariates and new responses. The names of the variables should correspond to the names of the variables used to fit the model. By default the data on which the model was estimated is considered.
type	"prob" for joint probabilities and "cum.prob" for joint cumulative probabilities.
subjectID	(optional) vector specifying for which subjectIDs the predictions or fitted values should be computed.
newoffset	(optional) list of length equal to the number of outcomes, each element containing a vector of offsets to be considered.
...	further arguments passed to or from other methods.

Details

The current implementation supports only in-sample predictions. The row names of the output correspond to the subjectIDs.

See Also

predict.mvord, marginal_predict

logLik.mvord *Pairwise Log-Likelihood of Multivariate Ordinal Regression Models.*

Description

logLik is a generic function which extracts the pairwise log-likelihood from objects of class 'mvord'.

Usage

```
## S3 method for class 'mvord'
logLik(object, ...)
```

Arguments

object	an object of class 'mvord'.
...	further arguments passed to or from other methods.

marginal_predict *Marginal Predictions for Multivariate Ordinal Regression Models.*

Description

Obtains marginal predictions/fitted measures for objects of class 'mvord'.

Usage

```
marginal_predict(object, newdata = NULL, type = "prob", subjectID = NULL,
  newoffset = NULL, ...)
```

Arguments

object	an object of class 'mvord'.
newdata	(optional) data frame of new covariates and new responses. The names of the variables should correspond to the names of the variables used to fit the model. By default the data on which the model was estimated is considered.
type	types "prob", "class", "linpred", "pred", "cum.prob" are available.
subjectID	(optional) vector specifying for which subjectIDs the predictions or fitted values should be computed.
newoffset	(optional) list of length equal to the number of outcomes, each element containing a vector of offsets to be considered.
...	further arguments passed to or from other methods.

Details

The following types can be chosen in marginal_predict:

type	description
"prob"	(default) fitted marginal probabilities for the observed response categories.
"class"	fitted marginal classes of the observed responses.
"linpred"	linear predictor
"cum.prob"	fitted marginal cumulative probabilities for the observed response categories.
"all.prob"	fitted marginal probabilities for all ordered classes of each response.

The current implementation supports only in-sample predictions. The row names of the output correspond to the subjectIDs.

See Also

predict.mvord, joint_probabilities

`model.matrix.mvord` *model.matrix of Multivariate Ordinal Regression Models.*

Description

`model.matrix` is a generic function which extracts the model matrix from objects of class `'mvord'`.

Usage

```
## S3 method for class 'mvord'
model.matrix(object, ...)
```

Arguments

`object`	an object of class `'mvord'`.
`...`	further arguments passed to or from other methods.

`mvlinks` *Multivariate link functions in mvord*

Description

Different `link` functions are available in **mvord**:

Usage

```
mvprobit()

mvlogit(df = 8L)
```

Arguments

df	integer specifying the degrees of freedom of the t copula

Details

We allow for two different link functions, the multivariate probit link and the multivariate logit link. For the multivariate probit link a multivariate normal distribution for the errors is applied. The normal bivariate probabilities which enter the pairwise log-likelihood are computed with the package **pbivnorm**.

For the multivariate logit link a t copula based multivariate distribution with logistic margins is used. The `mvlogit()` function has an optional integer valued argument `df` which specifies the degrees of freedom to be used for the t copula. The default value of the degrees of freedom parameter is 8. We restrict the degrees of freedom to be integer valued because the most efficient routines for computing bivariate t probabilities do not support non-integer degrees of freedom. For further details see vignette.

Value

The functions `mvlogit()` and `mvprobit()` returns an object of `class 'mvlink'`. An object of `class 'mvlink'` is a list containing the following components:

- `name`
  name of the multivariate link function
- `df`
  degrees of freedom of the t copula; returned only for `mvlogit()`
- `F_uni`
  a function corresponding to the univariate margins of the multivariate distribution F of the subject errors; the function returns $Pr(X \leq x) = F_1(x)$
- `F_biv`
  a function corresponding to the bivariate distribution of the multivariate distribution F of the subject errors $Pr(X \leq x, Y \leq y|r) = F_2(x, y, r)$;
- `F_biv_rect`
  the function computes the rectangle probabilities from based on `F_biv`; the function has the matrices `U` (upper bounds) and `L` (lower bounds) as well as vector `r` containing the correlation coefficients corresponding to the bivariate distribution as arguments; the matrices `U` and `L` both have two columns, first corresponding to the bounds of x, second to the bounds of y; the number of rows corresponds to the number of observations; the rectangle probabilities are defined as $Pr(L[,1] \leq X \leq U[,1], L[,2] \leq Y \leq U[,2]|r) = F_2(U[,1], U[,2], r) - F_2(U[,1], L[,2], r) - F_2(L[,1], U[,2], r) + F_2(L[,1], L[,2], r)$
- `F_multi`
  the function computes the multivariate probabilities for distribution function F; the function has the matrices `U` (upper bounds) and `L` (lower bounds) as well as the list `list_R` containing for each observation the correlation matrix; F is needed for the computation of the fitted/predicted joint probabilities. If NULL only marginal probabilities can be computed.

- deriv.fun
  (needed for computation of analytic standard errors) a list containing the following gradient functions:
  - dF1dx derivative $dF_1(x)/dx$ function,
  - dF2dx derivative $dF_2(x, y, r)/dx$ function,
  - dF2dr derivative $dF_2(x, y, r)/dr$ function.

  If deriv.fun = NULL numeric standard errors will be computed.

---

mvord *Multivariate Ordinal Regression Models.*

Description

Multivariate ordinal regression models in the R package mvord can be fitted using the function mvord(). Two different data structures can be passed on to mvord() through the use of two different multiple measurement objects MMO and MMO2 in the left-hand side of the model formula. MMO uses a long data format, which has the advantage that it allows for varying covariates across multiple measurements. This flexibility requires the specification a subject index as well as a multiple measurement index. In contrast to MMO, the function MMO2 has a simplified data structure, but is only applicable in settings where the covariates do not vary between the multiple measurements. In this case, the multiple ordinal observations as well as the covariates are stored in different columns of a data.frame. We refer to this data structure as wide data format.

Usage

```
mvord(formula, data, error.structure = cor_general(~1), link = mvprobit(),
  response.levels = NULL, coef.constraints = NULL, coef.values = NULL,
  threshold.constraints = NULL, threshold.values = NULL,
  weights.name = NULL, offset = NULL, PL.lag = NULL, contrasts = NULL,
  control = mvord.control())
```

Arguments

formula	an object of class formula of the form y ~ X1 + ... + Xp.
data	data.frame containing a subject index, an index for the multiple measurements, an ordinal response y and covariates X1, ..., Xp.
error.structure	different error.structures: general correlation structure (default) cor_general(~1), general covariance structure cov_general(~ 1), factor dependent correlation structure cov_general(~ f), factor dependent covariance structure cov_general(~ f), a constant cor_equi(~ 1) or a covariate dependent equicorrelation structure cor_equi(~ S), AR(1) correlation structure cor_ar1(~ 1) or a covariate dependent AR(1) correlation structure cor_ar1(~ S). See error_struct or 'Details'.

`link` specifies the link function by `mvprobit()` (multivariate normally distributed errors - default) or `mvlogit(df = 8)` (multivariate logistically distributed errors), where `df` specifies the degrees of freedom of the t copula.

`response.levels` (optional) `list` of length equal to the number of multiple measurements to specify the category labels in case of varying categories across multiple measurements

`coef.constraints` (optional) `vector` or `matrix` of constraints on the regression coefficients. See 'Details'.

`coef.values` (optional) `matrix` setting fixed values on the regression coefficients. See 'Details'.

`threshold.constraints` (optional) `vector` of constraints on the threshold parameters. See 'Details'.

`threshold.values` (optional) `list` of (optional) fixed values for the threshold parameters. See 'Details'.

`weights.name` (optional) character string with the column name of subject-specific weights in `data` which need to be constant across multiple measurements. Negative weights are not allowed.

`offset` (optional) this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.

`PL.lag` (optional) specifies the time lag of the pairs in the pairwise likelihood approach to be optimized (can be used with `cor_ar1`).

`contrasts` (optional) an optional list. See the `contrasts.arg` of `model.matrix.default`.

`control` (optional) a list of parameters for controlling the fitting process. See `mvord.control` for details.

Details

Implementation `MMO`:

- `data`: In `MMO` we use a long format for the input of data, where each row contains a subject index (`i`), a multiple measurement index (`j`), an ordinal observation (Y) and all the covariates (X1 to Xp). This long format data structure is internally transformed to a matrix of responses which contains NA in the case of missing entries and a list of covariate matrices. This is performed by the multiple measurement object `MMO(Y, i, j)` specifying the column names of the subject index and the multiple measurement index in data. The column containing the ordinal observations can contain integer or character values or can be of class (ordered) 'factor'. When using the long data structure, this column is basically a concatenated vector of each of the multiple ordinal responses. Internally, this vector is then split according to the measurement index. Then the ordinal variable corresponding to each measurement index is transformed into an ordered factor. For an integer or a character vector the natural ordering is used (ascending, or alphabetical). If for character vectors the alphabetical order does not correspond to the ordering of

the categories, the optional argument response.levels allows to specify the levels for each response explicitly. This is performed by a list of length q, where each element contains the names of the levels of the ordered categories in ascending (or if desired descending) order. If all the multiple measurements use the same number of classes and same labelling of the classes, the column Y can be stored as an ordered 'factor' (as it is often the case in longitudinal studies). The order of the multiple measurements is needed when specifying constraints on the thresh- old or regression parameters. This order is based on the type of the multiple measurement index column in data. For 'integer', 'character' or 'factor' the natural ordering is used (ascending, or alphabetical). If a different order of the multiple responses is desired, the multiple measurement index column should be an ordered factor with a corresponding ordering of the levels.
If the categories differ across multiple measurements (either the number of categories or the category labels) one needs to specify the `response.levels` explicitly. This is performed by a list of length J (number of multiple measurements), where each element contains the names of the levels of the ordered categories in ascending or descending order.

```
response.levels = list(c("G","F","E", "D", "C", "B", "A"),
                       c("G","F","E", "D", "C", "B", "A"),
                       c("O","N","M","L", "K", "J", "I", "H"))
```

- `formula` The ordinal responses (e.g., `rating`) are passed by a `formula` object. Intercepts can be included or excluded in the model depending on the model paramterization:
  - Model without intercept: If the intercept should be removed the `formula` for a given response (`rating`) and covariates (`X1` to `Xp`) has the following form:
    `formula = MMO(rating, firm_id, rater_id) ~ 0 + X1 + ... + Xp`.
  - Model with intercept: If one wants to include an intercept in the model, there are two equivalent possibilities to set the model `formula`. Either one includes the intercept explicitly by:
    `formula = MMO(rating, firm_id, rater_id) ~ 1 + X1 + ... + Xp`,
    or by
    `formula = MMO(rating, firm_id, rater_id) ~ X1 + ... + Xp`.

Implementation `MMO2`**:**

- `data`:The data structure applied by `MMO2` is slightly simplified, where the multiple ordinal observations as well as the covariates are stored as columns in a `data.frame`. Each subject i corresponds to one row of the data frame, where all outcomes (with missing observations set to NA) and all the covariates are stored in different columns. Ideally each outcome column is of type ordered factor. For column types like 'integer', 'character' or 'factor' a warning is given and the natural ordering is used (ascending, or alphabetical).
- `formula` The ordinal responses (e.g., `rating`) are passed by a `formula` object. Intercepts can be included or excluded in the model depending on the model parameterization:
  `formula = MMO2(rater1, rater2, rater3) ~ X1 + ... + Xp`.

`error.structure` We allow for different error structures depending on the model parameterization:

- Correlation:
  - `cor_general` The most common parameterization is the general correlation matrix.
    `error.structure = cor_general(~ 1)`
    This parameterization can be extended by allowing a factor dependent correlation structure, where the correlation of each subject i depends on a given subject-specific

factor `f`. This factor `f` is not allowed to vary across multiple measurements j for the same subject i and due to numerical constraints only up to maximum 30 levels are allowed.
`error.structure = cor_general(~ f)`

- `cor_equi` A covariate dependent equicorrelation structure, where the correlations are equal across all J dimensions and depend on subject-specific covariates `S1, ..., Sm`. It has to be noted that these covariates `S1, ..., Sm` are not allowed to vary across multiple measurements j for the same subject i.
`error.structure = cor_equi(~ S1 + ... + Sm)`
- `cor_ar1` In order to account for some heterogeneity the $AR(1)$ error structure is allowed to depend on covariates `X1, ..., Xp` that are constant over time for each subject i.
`error.structure = cor_ar1(~ S1 + ... + Sm)`

- Covariance:
  - `cov_general`
  In case of a full variance-covariance parameterization the standard parameterization with a full variance-covariance is obtained by:
  `error.structure = cov_general(~ 1)`
  This parameterization can be extended to the factor dependent covariance structure, where the covariance of each subject depends on a given factor `f`:
  `error.structure = cov_general(~ f)`

`coef.constraints` The package supports constraints on the regression coefficients. Firstly, the user can specify whether the regression coefficients should be equal across some or all response dimensions. Secondly, the values of some of the regression coefficients can be fixed.

As there is no unanimous way to specify such constraints, we offer two options. The first option is similar to the specification of constraints on the thresholds. The constraints can be specified in this case as a vector or matrix of integers, where coefficients getting same integer value are set equal. Values of the regression coefficients can be fixed through a matrix. Alternatively constraints on the regression coefficients can be specified by using the design employed by the **VGAM** package. The constraints in this setting are set through a named list, where each element of the list contains a matrix full-column rank. If the values of some regression coefficients should be fixed, offsets can be used. This design has the advantage that it supports constraints on outcome-specific as well as category-specific regression coefficients. While the first option has the advantage of requiring a more concise input, it does not support category-specific coefficients. The second option offers a more flexible design in this respect. For further information on the second option we refer to the vignette and to the documentation of `vglm`.

Using the first option, constraints can be specified by a vector or a matrix `coef.constraints`. First, a simple and less flexible way by specifying a vector `coef.constraints` of dimension J. This vector is allocated in the following way: The first element of the vector `coef.constraints` gets a value of 1. If the coefficients of the multiple measurement $j = 2$ should be equal to the coefficients of the first dimension ($j = 1$) again a value of 1 is set. If the coefficients should be different to the coefficients of the first dimension a value of 2 is set. In analogy, if the coefficients of dimensions two and three should be the same one sets both values to 2 and if they should be different, a value of 3 is set. Constraints on the regression coefficients of the remaining multiple measurements are set analogously.

```
coef.constraints <- c(1,1,2,3)
```

This vector `coef.constraints` sets the coefficients of the first two raters equal

$$\beta_{1\cdot} = \beta_{2\cdot}$$

A more flexible way to specify constraints on the regression coefficients is a matrix with J rows and p columns, where each column specifies constraints on one of the p coefficients in the same way as above. In addition, a value of `NA` excludes a corresponding coefficient (meaning it should be fixed to zero).

```
coef.constraints <- cbind(c(1,2,3,4), c(1,1,1,2), c(NA,NA,NA,1),
                          c(1,1,1,NA), c(1,2,3,4), c(1,2,3,4))
```

This matrix `coef.constraints` gives the following constraints:

- $\beta_{12} = \beta_{22} = \beta_{32}$
- $\beta_{13} = 0$
- $\beta_{23} = 0$
- $\beta_{33} = 0$
- $\beta_{44} = 0$
- $\beta_{14} = \beta_{24} = \beta_{34}$

`coef.values` In addition, specific values on regression coefficients can be set in the matrix `coef.values`. Parameters are removed if the value is set to zero (default for `NA`'s in `coef.constraints`) or to some fixed value. If constraints on parameters are set, these dimensions need to have the same value in `coef.values`. Again each column corresponds to one regression coefficient.

Together with the `coef.constraints` from above we impose:

```
coef.constraints <- cbind(c(1,2,2), c(1,1,2), c(NA,1,2),
                          c(NA,NA,NA), c(1,1,2))

coef.values <- cbind(c(NA,NA,NA), c(NA,NA,NA), c(0,NA,NA),
                     c(1,1,1), c(NA,NA,NA))
```

Interaction terms: When constraints on the regression coefficient should be specified in models with interaction terms, the `coef.constraints` matrix has to be expanded manually. In case of interaction terms (specified either by `X1 + X2 + X1:X2` or equivalently by `X1*X2`), one additional column at the end of `coef.constraints` for the interaction term has to be specified for numerical variables. For interaction terms including factor variables suitably more columns have to be added to the `coef.constraints` matrix.

`threshold.constraints` Similarly, constraints on the threshold parameters can be imposed by a vector of positive integers, where dimensions with equal threshold parameters get the same integer. When restricting the thresholds of two outcome dimensions to be the same, one has to be careful that the number of categories in the two outcome dimensions must be the same. In our example with $J = 4$ different outcomes we impose:

`threshold.constraints <- c(1,1,2)`

gives the following restrictions:

- $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$
- $\boldsymbol{\theta}_3$ arbitrary.

`threshold.values` In addition, threshold parameter values can be specified by `threshold.values` in accordance with identifiability constraints. For this purpose we use a `list` with J elements, where each element specifies the constraints of the particular dimension by a vector of length of the number of threshold parameters (number of categories - 1). A number specifies a threshold parameter to a specific value and `NA` leaves the parameter flexible. For `data_mvord` we have

```
threshold.constraints <- NULL

threshold.values <- list(c(-4,NA,NA,NA,NA,4.5),
                         c(-4,NA,NA,NA,NA,4.5),
                         c(-5,NA,NA,NA,NA,NA,4.5))
```

Value

The function `mvord` returns an object of `class` `"mvord"`.

The functions `summary` and `print` are used to display the results. The function `coef` extracts the regression coefficients, a function `thresholds` the threshold coefficients and the function `error_structure` returns the estimated parameters of the corresponding error structure.

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

- `beta`
  a named `matrix` of regression coefficients
- `theta`
  a named `list` of threshold parameters
- `error.struct`
  an object of class `error_struct` containing the parameters of the error structure
- `sebeta`
  a named `matrix` of the standard errors of the regression coefficients
- `setheta`
  a named `list` of the standard errors of the threshold parameters
- `seerror.struct`
  a `vector` of standard errors for the parameters of the error structure
- `rho`
  a `list` of all objects that are used in `mvord()`

See Also

`print.mvord`, `summary.mvord`, `coef.mvord`, `thresholds.mvord`, `error_structure.mvord`, `mvord.control`, `data_cr_panel`,`data_cr`, `data_mvord_panel`,`data_mvord`, `data_mvord2`

Examples

```
library(mvord)

#toy example
data(data_mvord_toy)
```

```
#wide data format with MMO2
res <- mvord(formula = MMO2(Y1, Y2) ~ 0 + X1 + X2,
             data = data_mvord_toy)
print(res)
summary(res)
thresholds(res)
coefficients(res)
head(error_structure(res))

# convert data_mvord_toy into long format
df <- cbind.data.frame("i" = rep(1:100,2), "j" = rep(1:2,each = 100),
                       "Y" = c(data_mvord_toy$Y1,data_mvord_toy$Y2),
                       "X1" = rep(data_mvord_toy$X1,2),
                       "X2" = rep(data_mvord_toy$X2,2))

#for long format data, use MMO instead of MMO2
res <- mvord(formula = MMO(Y, i, j) ~ 0 + X1 + X2, #or formula = MMO(Y) ~ 0 + X1 + X2
               data = df)
print(res)
summary(res)
thresholds(res)
coefficients(res)
head(error_structure(res))

res2 <- mvord(formula = MMO(Y) ~ 0 + X1 + X2,
               data = df,
               control = mvord.control(solver = "BFGS"),
               threshold.constraints = c(1,1),
               coef.constraints = c(1,1))
print(res2)
summary(res2)
thresholds(res2)
coefficients(res2)
head(error_structure(res2))

## examples
#load data
data(data_mvord)
head(data_mvord)

#-------------
# cor_general
#-------------
# approx 2 min
res_cor <- mvord(formula = MMO(rating) ~ 0 + X1 + X2 + X3 + X4 + X5,
                 data = data_mvord,
                 coef.constraints = cbind(c(1,2,2),
                                          c(1,1,2),
                                          c(NA,1,2),
                                          c(NA,NA,NA),
                                          c(1,1,2)),
                 coef.values = cbind(c(NA,NA,NA),
```

```
                                    c(NA,NA,NA),
                                    c(0,NA,NA),
                                    c(1,1,1),
                                    c(NA,NA,NA)),
                  threshold.constraints = c(1,1,2),
                  control = mvord.control(solver = "newuoa"))
print(res_cor)
summary(res_cor)
thresholds(res_cor)
coefficients(res_cor)
head(error_structure(res_cor))

#-------------
# cov_general
#-------------
#approx 4 min
res_cov <- mvord(formula = MMO(rating) ~ 1 + X1 + X2 + X3 + X4 + X5,
                  data = data_mvord,
                  error.structure = cov_general(~1),
                  threshold.values = list(c(-4,NA,NA,NA,NA,4.5),
                                          c(-4,NA,NA,NA,NA,4),
                                          c(-5,NA,NA,NA,NA,NA,4.5))
) #does not converge with BFGS

print(res_cov)
summary(res_cov)
thresholds(res_cov)
coefficients(res_cov)
head(error_structure(res_cov))


#-------------
# cor_ar1
#-------------
#approx 4min
data(data_mvord_panel)
head(data_mvord_panel)

#select subset of data
subset_dat <- data_mvord_panel$year %in% c("year3", "year4", "year5", "year6", "year7")
data_mvord_panel <- data_mvord_panel[subset_dat,]

mult.obs <- 5
res_AR1 <- mvord(formula = MMO(rating) ~ 0 + X1 + X2 + X3 + X4 + X5,
                  data = data_mvord_panel,
                  error.structure = cor_ar1(~1),
                  threshold.constraints = c(1,1,1,2,2),
                  coef.constraints = c(1,1,1,2,2),
                  control = mvord.control(solver = "BFGS"))
print(res_AR1)
summary(res_AR1)
thresholds(res_AR1)
coefficients(res_AR1)
```

```
head(error_structure(res_AR1))
head(error_structure(res_AR1, type = "corr"))

data(data_mvord2)
# approx 2 min
res_cor <- mvord(formula = MMO2(rater1, rater2, rater3) ~ 0 + X1 + X2 + X3 + X4 + X5,
                 data = data_mvord2,
                 coef.constraints = cbind(c(1,2,2),
                                          c(1,1,2),
                                          c(NA,1,2),
                                          c(NA,NA,NA),
                                          c(1,1,2)),
                 coef.values = cbind(c(NA,NA,NA),
                                     c(NA,NA,NA),
                                     c(0,NA,NA),
                                     c(1,1,1),
                                     c(NA,NA,NA)),
                 threshold.constraints = c(1,1,2),
                 control = mvord.control(solver = "newuoa"))
print(res_cor)
summary(res_cor)
thresholds(res_cor)
coefficients(res_cor)
head(error_structure(res_cor))
```

`mvord.control` *Control functions for mvord()*

Description

Control arguments are set for `mvord()`.

Usage

```
mvord.control(se = TRUE, start.values = NULL, solver = "newuoa",
  solver.optimx.control = list(maxit = 2e+05, trace = 0, kkt = FALSE))
```

Arguments

`se`	logical, if `TRUE` standard errors are computed.
`start.values`	vector of (optional) starting values.
`solver`	character string containing the name of the applicable solver of `optimx` (default is `"newuoa"`) or wrapper function for user defined solver.
`solver.optimx.control`	a list of control arguments to be passed to `optimx`. See `optimx`.

See Also

mvord

`names_constraints` *Names of regression coefficient constraints in mvord*

Description

An extractor function for the names of the regression coefficient constraints based on the model `formula` and `data`.

Usage

```
names_constraints(formula, data, contrasts = NULL)
```

Arguments

`formula`	model formula
`data`	a given data set.
`contrasts`	an optional list. See the `contrasts.arg` of `model.matrix.default`.

`nobs.mvord` *nobs of Multivariate Ordinal Regression Models.*

Description

`nobs` is a generic function which extracts the number of observations from objects of class `'mvord'`.

Usage

```
## S3 method for class 'mvord'
nobs(object, ...)
```

Arguments

`object`	an object of class `'mvord'`.
`...`	further arguments passed to or from other methods.

polycor *Computes polychoric correlations*

Description

This function computes polychoric correleations among two or more variables.

Usage

```
polycor(x, y = NULL)
```

Arguments

x	either a vector or a matrix of ordinal responses
y	an (optional) ordinal vector (only applciable if x is a vector)

predict.mvord *Predict method for Multivariate Ordinal Regression Models.*

Description

Obtains predicted or fitted values for objects of class 'mvord'.

Usage

```
## S3 method for class 'mvord'
predict(object, newdata = NULL, type = "prob",
  subjectID = NULL, newoffset = NULL, ...)
```

Arguments

object	an object of class 'mvord'.
newdata	(optional) data frame of new covariates and new responses.
type	types "class", "prob" and "cum.prob" are available.
subjectID	(optional) vector specifying for which subjectIDs the predictions or fitted values should be computed.
newoffset	(optional) list of length equal to the number of outcomes, each element containing a vector of offsets to be considered.
...	further arguments passed to or from other methods.

Details

type	description
"class"	combination of response categories with the highest probability.
"prob"	(default) fitted joint probability for the observed response categories or the categories provided in the response column(s) in newdata. If response column(s) in newdata contain only NAs, this will return a vector of ones.
"cum.prob"	fitted joint cumulative probability for the observed response categories or the categories provided in the response column(s) in newdata. If response column(s) in newdata contain only NAs, this will return a vector of ones.

The (row) names of the output correspond to the subjectIDs.

See Also

marginal_predict, joint_probabilities

---

print.error_struct *Print Method for class error_struc.*

Description

Prints error structure of class error_struct.

Usage

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

Arguments

x	object of class error_struct
...	further arguments passed to or from other methods.

---

print.mvord *Print Method for Multivariate Ordinal Regression Models.*

Description

Prints thresholds, regression coefficients and parameters of the error structure of class 'mvord'.

Usage

```
## S3 method for class 'mvord'
print(x, call = TRUE, ...)
```

Arguments

x	object of class 'mvord'
call	displays function call if TRUE
...	further arguments passed to or from other methods.

---

pseudo_R_squared *Pseudo R^2 for objects of class 'mvord'*

Description

This function computes Mc Fadden's Pseudo R^2 for objects of class 'mvord'.

Usage

```
pseudo_R_squared(object, adjusted = FALSE)
```

Arguments

object	an object of class 'mvord'.
adjusted	if TRUE, then adjusted Mc Fadden's Pseudo R^2 is computed.

See Also

mvord

---

summary.mvord *Summary method for Multivariate Ordinal Regression Models.*

Description

Summary of thresholds, regression coefficients and parameters of the error structure of class 'mvord'.

Usage

```
## S3 method for class 'mvord'
summary(object, short = TRUE, call = TRUE, ...)
```

Arguments

object	object of class 'mvord'
short	if TRUE short summary, otherwise extended summary
call	displays function call if TRUE
...	further arguments passed to or from other methods.

terms.mvord *terms of Multivariate Ordinal Regression Models.*

Description

terms is a generic function which can be used to extract terms from objects of class 'mvord'.

Usage

```
## S3 method for class 'mvord'
terms(x, ...)
```

Arguments

x	an object of class 'mvord'.
...	further arguments passed to or from other methods.

thresholds *Thresholds of Multivariate Ordinal Regression Models.*

Description

thresholds is a generic function which extracts threshold coefficients from objects of class 'mvord'.

Usage

```
thresholds(object, ...)

## S3 method for class 'mvord'
thresholds(object, ...)
```

Arguments

object	an object of class 'mvord'.
...	further arguments passed to or from other methods.

vcov.mvord *vcov of Multivariate Ordinal Regression Models.*

Description

vcov is a generic function which extracts the Godambe information matrix from objects of class 'mvord'.

Usage

```
## S3 method for class 'mvord'
vcov(object, ...)
```

Arguments

object	an object of class 'mvord'.
...	further arguments passed to or from other methods.

Index