Package 'MRSP'

January 27, 2015

Type Package

Title Multinomial Response Models with Structured Penalties

Version 0.4.3

Date 2014-12-09

Author Wolfgang Poessnecker

Maintainer Wolfgang Poessnecker <wolfgang.poessnecker@stat.uni-muenchen.de>

Description Fits regularized multinomial response models using penalized loglikelihood methods with structured penalty terms.

License GPL (>= 2)

Depends methods, parallel, compiler, matrixcalc, Formula, R (>= 2.14.1)

Imports base

LazyData yes

NeedsCompilation yes

Repository CRAN

Collate MRSP-prespecifier.r MRSP-classes-7.r FISTA-classes-3.r FISTA-generics-2.r FISTA-27.r MRSP-methods-15.r MRSP-cfunctions-2.r MRSP-helpers-21.r MRSP-fista-7.r MRSP-family-23.r MRSP-fit-39.r MRSP-formula-2.r MRSP-3.r zzz.r

Date/Publication 2014-12-09 16:11:35

R topics documented:

coef-methods *Methods for Function* coef *in Package* **MRSP**

Description

Extract the regression coefficients from an object of class MRSP.

Usage

```
## S4 method for signature 'MRSP'
coef(object, type=c("original", "standardized", "prethreshold"), simplify=TRUE, ...)
```

Arguments

object	Object of class MRSP.
type	Character string specifying the type of coefficients to extract from object.
simplify	Logical: should the output be simplified?
...	Not used here.

Details

Depending on argument type, the regression coefficients belonging to the original covariates or those belonging to standardized covariates are returned. For type="prethreshold", the regression coefficients prior to (potential) thresholding (and a potential backtransformation from standardized to original scale) are returned. type is found by a call to match.arg.

Depending on arguments standardize and threshold in the call to MRSP that created object, the output of this function might be identical for different choices of type.

If simplify=TRUE, the coefficient object is returned via invisible and its actual content is printed in structured and concise fashion.

Value

An object of class MRSP.coef. Depending on simplify, information about the coefficients is printed.

Author(s)

Wolfgang Poessnecker

Examples

```
## see the examples in ?MRSP
```

MRSP *Multinomial Response Models with Structured Penalties*

Description

Fit models with multinomial response (including ordinal regression) with structured penalties.

Usage

```
MRSP(formula, data, class.names = NULL, model = multinomlogit(), constr = NULL,
     offset = NULL, weights = NULL, penweights = NULL, standardize = TRUE,
     nrlambda = 50, lambdamin = 0.01, lambdamax = NULL, control = NULL,
     penalty = TRUE, group.classes = TRUE, group.dummies = TRUE,
     sparse.groups = FALSE, adaptive = FALSE, threshold = FALSE, refit = FALSE,
     lambda, lambdaR = 0, lambdaF = 0, gamma = 1, psi = 1, fusion = FALSE,
     nonneg = FALSE, y = NULL, X = NULL, Z = NULL, penindex = NULL,
     grpindex = NULL, mlfit = NULL, perform.fit = TRUE, ...)
```

Arguments

`formula`	A symbolic description of the model to be fitted. The left-hand side specifies the response, which must be a categorical variable with `K` categories. The right-hand-side specifies the covariate structure. See details below.
`data`	A data frame containing the variables in `formula`. It must be in **long** format. This means that `K` rows are required for each individual observation. For details and a possibility to convert to this format, see `mlogit.data` from the **mlogit** package. The response variable must be either a 0-1-vector or a logical vector, with 1 or TRUE indicating the observed class/category/alternative.
`class.names`	An optional character vector of length `K` specifying the names of the response classes/categories.
`model`	An object of class `MRSP.model` that specifies the model to be used. Currently, `model = multinomlogit()` and `model = sequentiallogit()` are available, yielding a multinomial or sequential logit model, respectively. Cumulative logit models will be included in future versions of `MRSP`.
`constr`	The identifiability constraint to be used. The coefficients of predictors which do not vary over categories (i.e. global/individual-specific predictors) are not identifiable in (unpenalized) multinomial logit models. Either an integer in `[1,K]` or `"symmetric"` or `"none"`. See details below.
`offset`	An optional vector of offsets. Must be of appropriate length.
`weights`	An optional vector of observations weights. Must be of appropriate length.
`penweights`	An object containing weights that modify the penalty terms on different parameters. See `MRSP.fit` for details.
`standardize`	If `TRUE`, predictors are mean-centered and standardized to unit variance. The reported coefficients, by default, correspond to the original covariates.

`nrlambda, lambdamin, lambdamax`
: If a sequence of `lambda` values is not specified explicitly via argument `lambda`, `MRSP` computes a suitable grid of length `nrlambda`, ranging from `lambdamin` to `lambdamax`. If `lambdamax` is missing, `MRSP` tries to find a suitable value for it.

`control`
: An object of class `MRSP.control` that stores control information. It's slots `max.iter` and `rel.tol` specify the max number of iterations and the relative change in penalized log-likelihood values that indicates convergence. The other slots should not be changed unless by experienced users.

`penalty`
: Specifies the general type of penalty to be applied. FALSE means no penalty is used. `TRUE` means that a lasso-type penalty is applied. `"Ridge"` applies a ridge penalty. Arguments `group.classes`, `group.dummies` and `sparse.groups` have no effect unless `penalty = TRUE`.

`group.classes`
: If `TRUE`, lasso-type penalties will be grouped across all coefficients belonging to the same covariate. This corresponds to the 'CATS Lasso' penalty proposed in Tutz, Poessnecker and Uhlmann (2015).

`group.dummies`
: If entries on the rhs of `formula` refer to factor variables with more than 2 levels, several dummy variables will enter the model that are related to the same actual covariate. Setting this argument to `TRUE` will yield penalty terms that treat all corresponding coefficients as one parameter group. This corresponds to the original 'Group Lasso' of Yuan and Lin (2006).

`sparse.groups, gamma`
: If `TRUE`, parameter groups will also be penalized by an L1 penalty on top of the unsquared L2 penalty. If the L2 penalty uses a tuning parameter of value lambda, the L1 penalty will use tuning parameter lambda*gamma.

`adaptive`
: Should adaptive weights be used? Use `adaptive="ML"` to obtain the traditional adaptive weights proposed in the literature. Using `adaptive = TRUE` computes the penalized estimator with whatever penalty is specified and no adaptive weights and computes adaptive weights from the output of this penalized model. The final output is the computed with those adaptive weights. It is strongly recommended to prefer `adaptive="ML"` over `adaptive=TRUE`.

`threshold`
: If `TRUE`, the coefficients will be thresholded with an appropriate threshold value. You can also specify an explicit nonnegative value to be used as the threshold.

`refit`
: Should refitting be performed? If `TRUE`, the model is first fit traditionally, and then refitted on the active set found by this first fit. This can improve variable selection, but tends to be rather slow and memory-consuming.

`lambda`
: Tuning parameter(s) to be used. Can be a nonnegative scalar or vector. If missing, `MRSP` computes a suitable grid of lambda values, see also `nrlambda`.

`lambdaR`
: Lambda value(s) for ridge penalties. Same structure as `lambda`.

`lambdaF`
: Lambda values(s) for fusion penalties. Same structure as `lambda`. Not yet supported for end-users of `MRSP`, but included for compatibility with future releases of `MRSP`.

`psi`
: A numeric that balances the weighting of penalties on global and class-/category-/alternative-specific predictors (if present) in multinomial logit models. Penalties on global predictors are weighted with `psi`, penalties on class-specific predictors are weighted with `(2-psi)`.

`fusion` If fusion penalties are used, this specifies the type of fusion. Not yet supported for end-users of `MRSP`, but included for compatibility with future releases of `MRSP`.

`nonneg` If `TRUE`, all coefficients are restricted to be nonnegative.

`y, X, Z, penindex, grpindex, mlfit` These optional arguments allow to supply the corresponding objects directly to `MRSP.fit`. It is highly recommended not to use those arguments and to use `formula` instead for the model specification.

`perform.fit` If TRUE, the model is fitted. If FALSE, function `MRSP` prepares the call to `MRSP.fit` and returns a list from which this call can be accessed.

`...` Further arguments to be passed.

Details

For `model = multinomlogit()`, a formula of the form "`Y ~ x | z1 | z2`" yields linear predictors

$$\eta_r = \beta_{0r} + x^T\beta_r + z_{1_r}^T\gamma + z_{2_r}^T\delta_r$$

for $r = 1, \ldots, K$, which are connected to probabilities by the multinomial logit link, also known as softmax function:

$$P(Y = r|x, z_1, z_2) = \frac{\exp(\eta_r)}{\sum_{s=1}^{K}\exp(\eta_s)}$$

This means that the `x`-variables have global values that are class-/category-/alternative-unspecific. The coefficients belonging to those global variables are not identifiable in the generic form of the multinomial logit model as given above. Therefore, an identifiability constraint can be specified with argument `constr`: By setting `constr = r` (with $r \in [1, K]$), category `r` is chosen as reference category. Technically, this means setting $\beta_r = 0$. Alternatively, `constr="symmetric"` specifies a so-called symmetric side constraint, which technically means imposing that

$$\sum_{s=1}^{K}\beta_{sj} = 0$$

for all $j = 1, \ldots, p$. If `constr = "none"`, no constraint is used for penalized parameter groups and identifiability is ensured by the penalty term (see Friedman, Hastie and Tibshirani, 2010). Coefficients of an unpenalized `x`-variable are subject to a symmetric side constraint in this case.
The `z1`- and `z2`-variables are class-/category-/alternative-specific. The `z1`-variables have a global effect while the category-specific `z2`-variables are equipped with coefficients that are also category-specific. Note that no identifiability constraints are required for category-specific variables.

For `model = sequentiallogit()`, a formula of the form "`Y ~ x1 | x2`" yields linear predictors of the form

$$\eta_r = \beta_{0r} + x_1^T\alpha + x_2^T\beta_r$$

for $r = 1, \ldots, K-1$, which are connected to conditional probabilities via the logit link in the following way:

$$P(Y = r|Y \geq r, x_1, x_2) = \frac{\exp(\eta_r)}{1 + \exp(\eta_r)}$$

for $r = 1, \ldots, K-1$.
Note that in the sequential case, slot "mu" of an MRSP-class object contains the unconditional class probabilities P(Y = r). If you want to get the "discrete" hazard rates P(Y = r | Y >= r), use methods fitted or predict with argument convert2hazard=TRUE.

Value

Depending on nrlambda, either an object of class MRSP or of class MRSP.list, which are lists of length nrlambda whose elements are MRSP objects. Additionally, the attributes "topcall", "call" and "dat" store, respectively, the call to MRSP, the call to MRSP.fit which was prepared by MRSP and the data object to be supplied to MRSP.fit. Note that dat contains the *standardized* covariates if standardize = TRUE.

Author(s)

Wolfgang Poessnecker

References

Tutz, G., Poessnecker, W. and Uhlmann, L. (2015): *Variable Selection in General Multinomial Logit Models*
Computational Statistics and Data Analysis, Vol. 82, 207-222.
http://www.sciencedirect.com/science/article/pii/S0167947314002709

Friedman, J., Hastie, T. and Tibshirani, R. (2010): *Regularization Paths for Generalized Linear Models via Coordinate Descent*, http://www.stanford.edu/~hastie/Papers/glmnet.pdf
Journal of Statistical Software, Vol. 33(1), 1-22.
http://www.jstatsoft.org/v33/i01/

Yuan, M. and Lin, Y. (2006): *Model selection and estimation in regression with grouped variables*
Journal of the Royal Statistical Society Series B, Vol. 68(1), 49-67.

Examples

```
## load data
data(TravelMode, package="MRSP")
## bring the response variable to the form required by MRSP
TravelMode$choice <- ifelse(TravelMode$choice=="yes",1,0)

## construct a list of fitted models for different lambda values.
## income is a global predictor, wait is a class-specific predictor with global
## coefficients, vcost and travel are specified as class-specific predictors
## with class-specific coefficients. The fourth category ("car") is chosen as
## reference.
fit <- MRSP(choice~income|wait|vcost+travel, data=TravelMode, constr=4,
            class.names=levels(TravelMode$mode), lambdamax=150, nrlambda=10,
            group.classes=TRUE, sparse.groups=FALSE, adaptive="ML")

fit
```

```
## slots can be extracted from all elements via function 'extract':
BICs <- extract(fit, "BIC")
lambdagrid <- extract(fit, "lambda")


## to select a concrete lambda/model, one can use function 'select'. Here, we
## chose the best model according to its AIC value
bestfit <- select(fit, "AIC")
bestfit

## some methods:
summary(bestfit)
BIC(bestfit)
fitted(bestfit)[1:6,]
bestfit@coef
## get the coefficients belonging to standardized predictors:
coef(bestfit, type="stand")
residuals(bestfit)
predict(bestfit, newdata = TravelMode[1:40,c(4,5,6,8)])

## plot some coefficient paths:
par(mfrow=c(1,2))
## you can either specify the number of the variable...
plot(fit, 2, legendpars=list(x="bottomright"))
## ... or its name as a character string. lcex is the cex parameter for legends.
## set it to zero to disable legend plotting.
plot(fit,"travel", lambda = bestfit@lambda, lcex=0)
```

`MRSP-class` *Class* `"MRSP"`

Description

An object containing the output of fitting a multinomial response model with structured penalties for one concrete value of the tuning parameter (or one concrete combination of all used tuning parameters).

Objects from the Class

Objects of this class are created by function `MRSP.fit`, which will most likely be called internally by the end-user function `MRSP`.

Slots

`coef`**:** Object of class `"MRSP.coef"` containing the estimated regression coefficients. Structurally, it is simply a list of one or two matrices.

`coef.stand`**:** Same as slot `coef`, but contains the coefficients belonging to standardized covariates.

`coef.pretres`: Same as slot `coef`, but contains the coefficients belonging to standardized covariates before any (potential) thresholding took place.

`dat`: Data object in the form required by `MRSP.fit`. To save memory, this slot is usually set to `NULL`.

`x.original`: Original matrix containing the `"x"`-variables. To save memory, this slot is often times set to `NULL`.

`x.stand`: The `"x"`-matrix after standardization.

`V.original`: The original data object containing the category-specific predictors. To save memory, this slot is often times `NULL`.

`V.stand`: The `"V"`-object after standardization.

`y`: Response matrix in the form required by `MRSP.fit`. To save memory, this slot is often times set to `NULL`.

`weights`: Vector of observation weights.

`penindex`: Object specifying how each covariate is penalized. See the documentation of `MRSP.fit` for details.

`grpindex`: Object specifying which predictors form parameter groups. See the documentation of `MRSP.fit` for details.

`penweights`: Object specifying the weighting of the penalty on different parameters or parameter groups. See `MRSP.fit` for details.

`guessed.active`: Object indexing the covariates that are found to have an effect on the response. If necessary, this refers to groups of coefficients.

`guessed.active.coef`: An object with the same structure as slot `coef`, indexing the atomic coefficients that are nonzero.

`guessed.active.groupdiff`: Index of `"columnwise"` parameter groups that contain at least two different parameter values.

`guessed.active.diff`: An index of nonzero pairwise differences between parameters belonging to the same covariate.

`df`: Estimated effective degrees of freedom.

`tuning`: List of tuning parameter values as supplied to internal function `fista`.

`lambda`: The lambda value, which controls the degree of penalization of most traditional penalties.

`lambdaR`: The lambda value used for ridge penalties.

`lambdaF`: The lambda value used for fusion penalties.

`fusion`: Either `FALSE` or character string specifying the type of fusion that was used in fusion penalties. Note that those fusion penalties are not yet supported for end-users of `MRSP`.

`gamma`: A numeric that weighs lasso penalties vs CATS lasso penalties. See `MRSP` for details.

`psi`: A numeric that weighs penalties on coefficients of global vs category-specific predictors. See `MRSP` for details.

`eta`: A `nobs x K` matrix of linear predictor values.

`mu`: A `nobs x K` matrix of estimated probabilities for the response categories.

`offset`: Vector of offset values that were added to the linear predictors.

`residuals`: Object with residuals. Currently always `NULL`. Use method `residuals` instead to compute residuals.

`mlfit`: Not to be used by or of interest for end-users.

`AIC`: The AIC of the fitted model.

`BIC`: The BIC of the fitted model.

`Brier`: The Brier score of the fitted model.

`threshold`: The numeric threshold used.

`refit`: Logical indicating whether the object results from a refitting procedure.

`indg`: Not to be used by or of interest for end-users.

`indcs`: Not to be used by or of interest for end-users.

`model`: The `model`-object used. To save memory, this is often stored as an expression.

`constr`: The identifiability constraint that was used. See `MRSP` for details.

`control`: Object of class `"MRSP.control"` that contains control information.

`fn.val`: The final value of the objective function that was minimized, i.e. the negative penalized loglikelihood: `-loglik + lambda*penweights*penalty`.

`loglik`: Loglikelihood value of the fitted model.

`penalty`: Value of the penalty term for the fitted model, weighted with the corresponding tuning parameters and `penweights`.

`iter.count`: Number of iterations until convergence of the proximal gradient algorithm.

`best.iter`: Iteration number with the best value of the objective function.

`ridgestabil`: Logical indicating whether a small, untuned ridge penalty was applied to all coefficients in order to stabilize otherwise diverging estimates.

`name`: A character string specifying the name and thus type of the fitted model.

`fisher`: Fisher matrix. Currently not computed and thus always `NULL`.

`arglist`: Not to be used by or of interest for end-users.

`call`: The call to `MRSP.fit` that created this object.

Methods

AIC `signature(object = "MRSP")`: Compute the AIC of an `MRSP` object.

BIC `signature(object = "MRSP")`: Compute the BIC of an `MRSP` object.

coef `signature(object = "MRSP")`: See `coef-methods`.

fitted `signature(object = "MRSP")`: ompute fitted values, which for multinomial response correspond to class probabilities P(Y = r). If option `convert2hazard=TRUE` and a sequential model is used, "discrete" hazard rates P(Y = r | Y >= r) are returned instead.

logLik `signature(object = "MRSP")`: Returns the loglikelihood and the (estimated, effective) degrees of freedom.

nobs `signature(object = "MRSP")`: Returns the number of invidual observations.

predict `signature(object = "MRSP")`: `predict(object, newdata, type=c("response", "link"), ...)` predicts the response values (`type="response"`) or the linear predictors (`type="link"`) for the observations given in `newdata`. Additional arguments `offset` and `weights` can specify offsets and weights to be used. An argument `convert2hazard` can be supplied for sequential models, see `fitted` above.

residuals `signature(object = "MRSP")`: Depending on argument `type = c("deviance", "pearson")`, which is matched via `match.arg`, deviance or pearson residuals are returned.

show `signature(object = "MRSP")`: Print some basic infos about the `MRSP` object.

summary `signature(object = "MRSP")`: Show some slots of an `MRSP` object which are typically of interest.

Author(s)

Wolfgang Poessnecker

References

Tutz, G., Poessnecker, W., Uhlmann, L. (2015) *Variable Selection in General Multinomial Logit Models*
Computational Statistics and Data Analysis, Vol. 82, 207-222.
http://www.sciencedirect.com/science/article/pii/S0167947314002709

See Also

`MRSP.list`

Examples

```
showClass("MRSP")
## for examples, see ?MRSP
```

`MRSP.fit` *Fitting function for multinomial response models with structured penalties*

Description

This function performs the actual fitting of multinomial response models with structured penalties. Function `MRSP` is actually just a user-friendly wrapper that prepares calls to this function. `MRSP.fit` is designed mostly for internal use and therefore not user-friendly, so that it is highly recommended to use `MRSP` instead of calling `MRSP.fit` directly.

Usage

```
MRSP.fit(dat, coef.init = NULL, coef.stand.init = NULL, coef.pretres.init = NULL,
        offset = NULL, weights = NULL, grpindex = NULL, penindex = NULL, lambda,
         lambdaR = lambda, lambdaF = lambda, gamma = 1, psi = 1, indg = NULL,
         indcs = NULL, model = NULL, constr = NULL, control = NULL,
         fista.control = NULL, Proximal.control = NULL, Proximal.args = NULL,
         penweights = NULL, mlfit = NULL, adaptive = FALSE, threshold = FALSE,
         refit = FALSE, fusion = FALSE, nonneg = FALSE, ...)
```

Arguments

`dat`

A list that contains the data in the format required by `MRSP.fit`. `dat` is a list with elements 'y', 'x' and, possibly, 'V'. If `nobs` individual observations are available, `dat$y` must be a matrix of dimension `nobs x K`. For `model=multinomlogit()`, `K` is equal to the number of categories of the response variable, for `model=sequentiallogit()`, it equals the number of response categories minus 1. In other words: `K` here always refers to the column rank of the response matrix. (Note that this is for notational convenience and in contrast to the documentation of MRSP, where `K` always refers to the number of categories of the multicategorical response!) The entries of `dat$y` are either 0 or 1, with `dat$y[i,r]==1` indicating that class `r` is observed for the i-th observation. For `model=multinomlogit()`, the rowSums of `dat$y` must be all 1; for `model=sequentiallogit()`, they must be either 1 or 0; with a row of zeros indicating that the last category was observed.

`dat$x` is a matrix of dimension `nobs x p` which contains covariates whose value is constant across classes. They are called 'global predictors/covariates' in the following. In the context of discrete choice modeling, they are often referred to as 'individual-specific' predictors.

If available, covariates whose value varies from class to class can be included in an entry `dat$V`. Such variables are called 'category-specific' in the following since their value depends on the categories of the response variable. In the literature on discrete choice modeling, they are often referred to as 'alternative-specific'. These variables can either be equipped with global or with category-specific coefficients. If a total of `L` category-specific variables shall be used, `dat$V` must be a list (!) of length `K` whose elements each are matrices of dimension `nobs x L`.

`coef.init`

An optional coefficient object supplying initial coefficient values to be used. A list whose first entry is a matrix of dimension `K x p`, with row `r` containing the coefficients for class `r` and column `j` containing the coefficients for global predictor x_j. If category-specific predictors are included, the second entry of `coef.init` is a matrix of dimension `K x L` that contains the coefficients for those category-specific predictors.

`coef.stand.init`

Optional initial coefficient values for the standardized predictors. Same structure as `coef.init`.

`coef.pretres.init`
: Optional initial coefficient values, prior to potential thresholding, for the standardized predictors. Same structure as `coef.init`.

`offset`
: An optional vector or matrix of offset values to be used. Either length `nobs` or dimension `nobs x K`.

`weights`
: An optional vector of observation weights of length `nobs`.

`grpindex`
: A list of one or two integer vectors that indicate which columns of the design matrix form a group that has to be penalized jointly, e.g. the different dummies of a categorical predictor. The first element is the grouping vector for x, the optional second one for V. Those columns with the same number belong to one group. The numbers must begin with 1 and increase with every group. An example: grpindex = list(c(1,2,3,3,4,4,4)) means that variables 1 and 2 form their own, 'scalar' group; variables 3 and 4 as well as variables 5, 6 and 7 form multi-parameter-groups.

`penindex`
: A list of one or two vectors which specifies the exact penalty type to use for each covariate. The first entry specifies the penalty type for the variables in `dat$x`, the (optional) second entry those for the variables in `dat$V`. The following penalty types are available:

  1: global predictor ('x') whose coefficients shall be penalized with a group lasso penalty with grouping 'across' categories, i.e. CATS Lasso (see Tutz, Poessnecker and Uhlmann, 2015).
  10: global predictor, unpenalized.
  11: global predictor, sparse group lasso.
  12: global predictor, ordinary lasso.
  13: global predictor, ridge penalty. does not support penweights.
  2: category-specific predictor with global coefficient which is penalized with the ordinary (group-)lasso. (depending on grpindex.)
  20: category-specific, unpenalized.
  21: category-specific, ridge penalty.
  3: category-speficic predictor with category-specific coefficients that are penalized by a group lasso like in '1'.
  30: category-specific with category-specific coefs, unpenalized.
  31: category-specific with category-specific coefficients and sparse group lasso penalty.
  32: cat-cat-specific, with ordinary lasso.
  33: cat-cat, with ridge. does not support penweights.
  4: global predictor with global effect, penalized. (cf. the '2'-series).
  40: global predictor, global effect, unpenalized.
  41: global predictor, global effect, ridge penalty.

  The '4-series' only makes sense for ordinal models!

`lambda`
: Optional object specifying the lambda values to be used as tuning parameter(s) for the main variable selection penalty. Either a vector or a single numeric. If missing, a suitable grid of lambda values is computed. See arguments `nrlambda, lambdamin` and `lambdamax` in function `MRSP`.

`lambdaR`
: Lambda(s) to be used for ridge penalties. Typically, if only a ridge penalty and

no other penalty is used, one can specify the Ridge lambda via argument `lambda` instead.

`lambdaF` Lambda(s) to be used for fusion penalties. Not available yet for end-users, but included for compatibility with future releases of `MRSP`.

`gamma` See argument `gamma` in `MRSP`.

`psi` See argument `psi` in `MRSP`.

`indg` A vector of the column indices of the category-specific variables that are equipped with global coefficients.

`indcs` A vector of the column indices of the category-specific variables that are equipped with category-specific coefficients

`model` An object of class `MRSP.model` that specifies the model to be used. Currently, `model = multinomlogit()` and `model = sequentiallogit()` are available, yielding a multinomial or sequential logit model, respectively. Cumulative logit models will be included in future versions of `MRSP`.

`constr` The identifiability constraint to be used. The coefficients of predictors which do not vary over categories (i.e. global/individual-specific predictors) are not identifiable in (unpenalized) multinomial logit models. If `constr` is an integer in [1, K], the corresponding class is used as reference, which means that the coefficients of global predictors for this class are set to 0. If `constr = "symmetric"`, a symmetric side constraint is used, which means that all coefficients belonging to the same global predictor sum to zero. If `constr = "none"`, no constraint is used for penalized parameter groups and identifiability is ensured by the penalty term (see Friedman, Hastie and Tibshirani, 2010.) For ordinal regression, `constr` must take value `"none"`. If left unspecified, a symmetric side constraint is used for multinomial and no constraint for ordinal models.

`control` An object of class `MRSP.control` that stores control information. It's slots `max.iter` and `rel.tol` specify the max number of iterations and the relative change in penalized log-likelihood values that indicates convergence. The other slots should not be changed unless by experienced users.

`fista.control` An object of class `fista.control` that contains control information for the FISTA algorithm that is internally used to compute numerical estimates. Not intended for end-users!

`Proximal.control, Proximal.args`
Arguments to be passed to the proximal gradient algorithm. Not intended for end-users!

`penweights` An optional list containing weights for the various penalty terms of different coefficients or coefficient groups. Assuming that category-specific covariates are present, the first element of penweights is a list of length two, with the first element of `penweights[[1]]` being a numeric of length `p` that contains the weights for the CATS penalty on the group of coefficients of the global covariates for the response classes. The second element of `penweights[[1]]` is a numeric of length `L` with the group penalty weights for the category-specific variables. The second element of penweights is again a list of length two. The first element of `penweights[[2]]` is a `K x p` matrix with penalty weights for unstructured lasso penalties on atomic coefficients belonging to global predictors. The second element of `penweights[[2]]` is a `K x L` matrix with penalty weights for unstructured lasso penalties on category-specific covariates.

mlfit	A list that contains information about the ML or 'pseudo-ML' coefficients of the specified model. It must contain at least one entry called 'coef.stand' that has the same structure as the coefficient object (see coef.init). The value of those parameters must be known to compute the effective degrees of freedom of penalized parameter estimates with grouped penalties.
adaptive	Should adaptive weights be used? Use adaptive="ML" to obtain the traditional adaptive weights proposed in the literature. Using adaptive = TRUE computes the penalized estimator with whatever penalty is specified and no adaptive weights and computes adaptive weights from the output of this penalized model. The final output is the computed with those adaptive weights. It is strongly recommended to prefer adaptive="ML" over adaptive=TRUE.
threshold	If TRUE, the coefficients will be thresholded with an appropriate threshold value. You can also specify an explicit nonnegative value to be used as the threshold.
refit	Should refitting be performed? If TRUE, the model is first fit traditionally, and then refitted on the active set found by this first fit. This can improve variable selection, but tends to be rather slow and time-consuming.
fusion	If fusion penalties are used, this specifies the type of fusion. Not yet supported for end-users of MRSP, but included for compatibility with future releases of MRSP.
nonneg	If TRUE, all coefficients are restricted to be nonnegative.
...	Further arguments or objects to be passed to MRSP.fit.

Details

This function does the actual work of fitting multinomial response models with structured penalties. It is intended mainly for internal use. The main purpose of function MRSP is to provide a user-friendly wrapper that prepares and evaluates a call to MRSP.fit.

Value

Depending on nrlambda, either an object of class MRSP or of class MRSP.list, which are lists of length nrlambda whose elements are MRSP objects.

Author(s)

Wolfgang Poessnecker

References

Tutz, G., Poessnecker, W., Uhlmann, L. (2015) *Variable Selection in General Multinomial Logit Models*
Computational Statistics and Data Analysis, Vol. 82, 207-222.
http://www.sciencedirect.com/science/article/pii/S0167947314002709

MRSP.list-class *Class* "MRSP.list"

Description

An object containing the output of MRSP-calls with more than one tuning parameter value, i.e. model output related to a whole "coefficient path". With nrlambda denoting the number of different lambda values for which to fit a penalized multinomial response model, MRSP.list-objects are lists of length nrlambda. Each entry of this list is an object of class MRSP and contains the model output for one concrete lambda value. If an MRSP.list-object was created by function MRSP, it additionally has attributes "topcall", "call" and "dat". Those attributes store, respectively, the call to MRSP, the call to MRSP.fit which was prepared by MRSP and the data object to be supplied to MRSP.fit. Note that dat contains the *standardized* covariates if standardize = TRUE.

Objects from the Class

Objects of this class are created by function MRSP.fit, which will most likely be called internally by the end-user function MRSP.

Methods

cv signature(object = "MRSP.list"): cv(object, k, type, parallel, cores, ...) performs k-fold crossvalidation of the models in object. The output is a list whose entry mean gives the mean values of the criterion specified via argument type over the crossvalidation folds; for each model found in object. type can be "deviance" (the default), "loglik" or "brier", using the deviance, loglikelihood or the Brier score. Depending on the logical argument parallel and integer argument cores, crossvalidation is performed either serially or with cores parallel processes.

extract signature(object = "MRSP.list"): extract(object, slotname) returns the value of the slot whose name was given as a character string from all elements of object. The result is coerced to a vector or matrix if appropriate.

getCall signature(x = "MRSP.list"): Get the call that created the MRSP.list-object.

plot signature(x = "MRSP.list"): Plot coefficient paths.

select signature(object = "MRSP.list"): select(object, criterion=c("AIC","BIC","cv"), ...) selects one concrete MRSP model from object based on criterion. For criterion="cv", an additional, integer argument k, a logical argument parallel and an integer argument cores can be supplied, see cv above.

show signature(object = "MRSP.list"): Print some basic infos about the MRSP.list object.

Author(s)

Wolfgang Poessnecker

References

Tutz, G., Poessnecker, W., Uhlmann, L. (2015) *Variable Selection in General Multinomial Logit Models*
Computational Statistics and Data Analysis, Vol. 82, 207-222.
http://www.sciencedirect.com/science/article/pii/S0167947314002709

See Also

MRSP

Examples

```
showClass("MRSP.list")
```

TravelMode *Travel Mode Choice*

Description

Data on travel mode choice for travel between Sydney and Melbourne, Australia.

Usage

```
data("TravelMode")
```

Format

A data frame containing 840 observations on 4 modes for 210 individuals. This corresponds to so-called **long** format since we have 4 rows per individual in a situation with 4 classes/categories of the response variable `mode`.

individual Factor indicating individual with levels 1 to `200`.

mode Factor indicating travel mode with levels `"car"`, `"air"`, `"train"`, or `"bus"`.

choice Factor indicating choice with levels `"no"` and `"yes"`.

wait Terminal waiting time, 0 for car.

vcost Vehicle cost component.

travel Travel time in the vehicle.

gcost Generalized cost measure.

income Household income.

size Party size.

Details

Data and description are taken from R package **AER** by Christian Kleiber and Achim Zeileis.

In MRSP-terminology, variables `wait`, `vcost`, `travel` and `gcost` are category-specific since they take different values for different categories of `mode`. The variables `income` and `size` are global. In the literature on discrete choice modelling, these two variable types are called "alternative-specific" and "individual-specific", respectively.

Source

R package **AER**, which states its source as follows:

Online complements to Greene (2003).

http://pages.stern.nyu.edu/~wgreene/Text/tables/tablelist5.htm

References

Greene, W.H. (2003). *Econometric Analysis*, 5th edition. Upper Saddle River, NJ: Prentice Hall.

Index