Package 'smurf'

November 24, 2020

Type Package

Title Sparse Multi-Type Regularized Feature Modeling

Version 1.0.7

Date 2020-11-24

Description Implementation of the SMuRF algorithm of Devriendt et al. (2018) <arXiv:1810.03136> to fit generalized linear models (GLMs) with multiple types of predictors via regularized maximum likelihood.

URL https://gitlab.com/TReynkens/smurf

License GPL (>= 2)

BugReports https://gitlab.com/TReynkens/smurf/-/issues

Depends R (>= 3.1)

Imports catdata, glmnet (>= 4.0), graphics, MASS, Matrix, methods, mgcv, parallel, RColorBrewer, Rcpp (>= 0.12.12), speedglm, stats

Suggests bookdown, knitr, roxygen2 (>= 6.0.0), testthat

LinkingTo Rcpp, RcppArmadillo (>= 0.8.300.1.0)

VignetteBuilder knitr

ByteCompile yes

Encoding UTF-8

LazyData yes

NeedsCompilation yes

RoxygenNote 7.1.1

Author Tom Reynkens [aut, cre] (<https://orcid.org/0000-0002-5516-5107>), Sander Devriendt [aut], Katrien Antonio [aut]

Maintainer Tom Reynkens <tomreynkens@hotmail.com>

Repository CRAN

Date/Publication 2020-11-24 10:00:02 UTC

R topics documented:

`smurf-package` *smurf: Sparse Multi-Type Regularized Feature Modeling*

Description

Implementation of the SMuRF algorithm of Devriendt et al. (2018) <arXiv:1810.03136> to fit generalized linear models (GLMs) with multiple types of predictors via regularized maximum likelihood.

Author(s)

Maintainer: Tom Reynkens <tomreynkens@hotmail.com> (ORCID)

Authors:

- Sander Devriendt <sander.devriendt@kuleuven.be>
- Katrien Antonio

See Also

Useful links:

- https://gitlab.com/TReynkens/smurf
- Report bugs at https://gitlab.com/TReynkens/smurf/-/issues

coef.glmsmurf *Coefficients of Estimated Model*

Description

Function to extract the coefficients of the estimated model. `coefficients` is an *alias* for it.

Usage

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

## S3 method for class 'glmsmurf'
coefficients(object, ...)
```

Arguments

`object`	An object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
`...`	Additional arguments which are currently ignored.

Value

A vector containing the coefficients of the estimated model in `object`.

See Also

coef_reest, coef, summary.glmsmurf, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

coef_reest *Coefficients of Re-estimated Model*

Description

Function to extract the coefficients of the re-estimated model. `coefficients_reest` is an *alias* for it.

Usage

```
coef_reest(object, ...)

## S3 method for class 'glmsmurf'
coef_reest(object, ...)

coefficients_reest(object, ...)

## S3 method for class 'glmsmurf'
coefficients_reest(object, ...)
```

Arguments

object	An object for which the extraction of model coefficients is meaningful. E.g. an object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
...	Additional arguments which are currently ignored.

Value

A vector containing the coefficients of the re-estimated model in object, when they are available, or, otherwise, the coefficients of the estimated model in object with a warning.

See Also

coef.glmsmurf, coef, summary.glmsmurf, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

---

deviance.glmsmurf *Deviance of Estimated Model*

Description

Function to extract the deviance of the estimated model.

Usage

```
## S3 method for class 'glmsmurf'
deviance(object, ...)
```

Arguments

object	An object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
...	Additional arguments which are currently ignored.

Value

The deviance of the estimated model in object.

See Also

deviance_reest, deviance, summary.glmsmurf, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

---

deviance_reest *Deviance of Re-estimated Model*

Description

Function to extract the deviance of the re-estimated model.

Usage

```
deviance_reest(object, ...)

## S3 method for class 'glmsmurf'
deviance_reest(object, ...)
```

Arguments

object	An object for which the extraction of the deviance is meaningful. E.g. an object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
...	Additional arguments which are currently ignored.

Value

The deviance of the re-estimated model in object, when it is available or, otherwise, the deviance of the estimated model in object with a warning.

See Also

deviance.glmsmurf, deviance, summary.glmsmurf, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

`fitted.glmsmurf` *Fitted Values of Estimated Model*

Description

Function to extract the fitted values of the estimated model.

Usage

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

Arguments

`object`	An object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
`...`	Additional arguments which are currently ignored.

Value

A vector containing the fitted values of the estimated model in `object`.

See Also

fitted_reest, fitted, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

fitted_reest *Fitted Values of Re-estimated Model*

Description

Function to extract the fitted values of the re-estimated model.

Usage

```
fitted_reest(object, ...)

## S3 method for class 'glmsmurf'
fitted_reest(object, ...)
```

Arguments

`object`	An object for which the extraction of fitted values is meaningful. E.g. an object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
`...`	Additional arguments which are currently ignored.

Value

A vector containing the fitted values of the re-estimated model in `object`, when they are available or, otherwise, the fitted values of the estimated model in `object` with a warning.

See Also

fitted.glmsmurf, fitted, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

glmsmurf *Fit a Multi-Type Regularized GLM Using the SMuRF Algorithm*

Description

SMuRF algorithm to fit a generalized linear model (GLM) with multiple types of predictors via regularized maximum likelihood. `glmsmurf.fit` contains the fitting function for a given design matrix.

Usage

```
glmsmurf(
  formula,
  family,
  data,
  weights,
  start,
  offset,
  lambda,
  lambda1 = 0,
  lambda2 = 0,
  pen.weights,
  adj.matrix,
  standardize = TRUE,
  control = list(),
  x.return = FALSE,
  y.return = TRUE,
  pen.weights.return = FALSE
)

glmsmurf.fit(
  X,
  y,
  weights,
  start,
  offset,
  family,
  pen.cov,
  n.par.cov,
  group.cov,
  refcat.cov,
  lambda,
  lambda1 = 0,
  lambda2 = 0,
  pen.weights,
  adj.matrix,
  standardize = TRUE,
  control = list(),
  formula = NULL,
  data = NULL,
  x.return = FALSE,
  y.return = FALSE,
  pen.weights.return = FALSE
)
```

Arguments

formula	A formula object describing the model to be fitted. Penalties are specified using the p function. For glmsmurf.fit this is an optional argument which is only used when penalty weights are computed using a generalized additive model (GAM).
family	A family object specifying the error distribution and link function for the model.
data	A data frame containing the model response and predictors for n observations.
weights	An optional vector of prior weights to use in the likelihood. It should be a numeric vector of length n (the number of observations), or NULL. When NULL or nothing is given, equal prior weights (all ones) will be used.
start	A vector containing the starting values for the coefficients. It should either be a numeric vector of length p+1 (with p the number of parameters excluding the intercept) or NULL. In the latter case, the link function applied to the weighted average of the response vector is used as starting value for the intercept and zero for the other coefficients.
offset	A vector containing the offset for the model. It should be a vector of size n or NULL (no offset). Offset(s) specified using the formula object will be ignored!
lambda	Either the penalty parameter, a positive number; or a string describing the method and measure used to select the penalty parameter: • "is.aic" (in-sample; Akaike Information Criterion (AIC)), • "is.bic" (in-sample; Bayesian Information Criterion (BIC)), • "is.gcv" (in-sample; Generalized Cross-Validation (GCV) score), • "oos.dev" (out-of-sample; deviance), • "oos.mse" (out-of-sample; Mean Squared Error (MSE)), • "oos.dss" (out-of-sample; Dawid-Sebastiani Score (DSS)), • "cv.dev" (cross-validation (CV); deviance), • "cv.mse" (CV; MSE), • "cv.dss" (CV; DSS), • "cv1se.dev" (CV with one standard error (SE) rule; deviance), • "cv1se.mse" (CV with one SE rule; MSE), • "cv1se.dss" (CV with one SE rule; DSS). E.g. "is.aic" indicates in-sample selection of lambda with the AIC as measure. When lambda is missing or NULL, it will be selected using cross-validation with the one standard error rule and the deviance as measure ("cv1se.dev").
lambda1	The penalty parameter for the L_1-penalty in Sparse (Generalized) Fused Lasso or Sparse Graph-Guided Fused Lasso is $\lambda \times \lambda_1$. A positive numeric with default 0 (no extra L_1-penalty).
lambda2	The penalty parameter for the L_2-penalty in Group (Generalized) Fused Lasso or Group Graph-Guided Fused Lasso is $\lambda \times \lambda_2$. A positive numeric with default 0 (no extra L_2-penalty).
pen.weights	Either a string describing the method to compute the penalty weights: • "eq" (default; equal penalty weights),

- "stand" (standardization penalty weights),
- "glm" (adaptive GLM penalty weights),
- "glm.stand" (stand. ad. GLM penalty weights),
- "gam" (ad. GAM penalty weights),
- "gam.stand" (stand. ad. GAM penalty weights);

or a list with the penalty weight vector per predictor. This list should have length equal to the number of predictors and predictor names as element names.

adj.matrix — A named list containing the adjacency matrices (a.k.a. neighbor matrices) for each of the predictors with a Graph-Guided Fused Lasso penalty. The list elements should have the names of the corresponding predictors. If only one predictor has a Graph-Guided Fused Lasso penalty, it is also possible to only give the adjacency matrix itself (not in a list).

standardize — Logical indicating if predictors with a Lasso or Group Lasso penalty are standardized, default is TRUE. The returned coefficients are always on the original (i.e. non-standardized) scale.

control — A list of parameters used in the fitting process. This is passed to glmsmurf.control.

x.return — Logical indicating if the used model matrix should be returned in the output object, default is FALSE.

y.return — Logical indicating if the used response vector should be returned in the output object, default is TRUE.

pen.weights.return — Logical indicating if the list of the used penalty weight vector per predictor should be returned in the output object, default is FALSE.

X — Only for glmsmurf.fit: the design matrix including ones for the intercept. A n by (p+1) matrix which can be of numeric matrix class (matrix-class) or of class Matrix (Matrix-class) including sparse matrix class (dgCMatrix-class).

y — Only for glmsmurf.fit: the response vector, a numeric vector of size n.

pen.cov — Only for glmsmurf.fit: a list with the penalty type per predictor (covariate). A named list of strings with predictor names as element names. Possible types: "none" (no penalty, e.g. for intercept), "lasso" (Lasso), "grouplasso" (Group Lasso), "flasso" (Fused Lasso), "gflasso" (Generalized Fused Lasso), "2dflasso" (2D Fused Lasso) or "ggflasso" (Graph-Guided Fused Lasso).

n.par.cov — Only for glmsmurf.fit: a list with the number of parameters to estimate per predictor (covariate). A named list of strictly positive integers with predictor names as element names.

group.cov — Only for glmsmurf.fit: a list with the group of each predictor (covariate) which is only used for the Group Lasso penalty. A named list of positive integers with predictor names as element names where 0 means no group.

refcat.cov — Only for glmsmurf.fit: a list with the number of the reference category in the original order of the levels of each predictor (covariate). When the predictor is not a factor or no reference category is present, it is equal to 0. This number will only be taken into account for a Fused Lasso, Generalized Fused Lasso or Graph-Guided Fused Lasso penalty when a reference category is present.

Details

See the package vignette for more details and a complete description of a use case.

As a user, it is important to take the following into acocunt:

- The estimated coefficients are rounded to 7 digits.
- The cross-validation folds are not deterministic. The validation sample for selecting lambda out-of-sample is determined at random when no indices are provided in 'validation.index' in the control object argument. In these cases, the selected value of lambda is hence not deterministic. When selecting lambda in-sample, or out-of-sample when indices are provided in 'validation.index' in the control object argument, the selected value of lambda is deterministic.
- The `glmsmurf` function can handle many use cases and is preferred for general use. The `glmsmurf.fit` function requires a more thorough understanding of the package internals and should hence be used with care!

Value

An object of class '`glmsmurf`' is returned. See `glmsmurf-class` for more details about this class and its generic functions.

References

Devriendt, S., Antonio, K., Reynkens, T. and Verbelen, R. (2018). "Sparse Regression with Multi-type Regularized Feature Modeling." *arXiv:1810.03136*.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press.

See Also

`glmsmurf-class`, `glmsmurf.control`, `p`, `glm`

Examples

```
# Munich rent data from catdata package
data("rent", package = "catdata")

# The considered predictors are the same as in
# Gertheiss and Tutz (Ann. Appl. Stat., 2010).
# Response is monthly rent per square meter in Euro

# Urban district in Munich
rent$area <- as.factor(rent$area)

# Decade of construction
rent$year <- as.factor(floor(rent$year / 10) * 10)

# Number of rooms
rent$rooms <- as.factor(rent$rooms)

# Quality of the house with levels "fair", "good" and "excellent"
```

```
rent$quality <- as.factor(rent$good + 2 * rent$best)
levels(rent$quality) <- c("fair", "good", "excellent")

# Floor space divided in categories (0, 30), [30, 40), ...,  [130, 140)
sizeClasses <- c(0, seq(30, 140, 10))
rent$size <- as.factor(sizeClasses[findInterval(rent$size, sizeClasses)])

# Is warm water present?
rent$warm <- factor(rent$warm, labels = c("yes", "no"))

# Is central heating present?
rent$central <- factor(rent$central, labels = c("yes", "no"))

# Does the bathroom have tiles?
rent$tiles <- factor(rent$tiles, labels = c("yes", "no"))

# Is there special furniture in the bathroom?
rent$bathextra <- factor(rent$bathextra, labels = c("no", "yes"))

# Is the kitchen well-equipped?
rent$kitchen <- factor(rent$kitchen, labels = c("no", "yes"))



# Create formula with 'rentm' as response variable,
# 'area' with a Generalized Fused Lasso penalty,
# 'year', 'rooms', 'quality' and 'size' with Fused Lasso penalties,
# and the other predictors with Lasso penalties.
formu <- rentm ~ p(area, pen = "gflasso") +
 p(year, pen = "flasso") + p(rooms, pen = "flasso") +
 p(quality, pen = "flasso") + p(size, pen = "flasso") +
 p(warm, pen = "lasso") + p(central, pen = "lasso") +
 p(tiles, pen = "lasso") + p(bathextra, pen = "lasso") +
 p(kitchen, pen = "lasso")


# Fit a multi-type regularized GLM using the SMuRF algorithm.
# We use standardization adaptive penalty weights based on an initial GLM fit.
# The value for lambda is selected using cross-validation
# (with the deviance as loss measure and the one standard error rule), see example(plot_lambda)
munich.fit <- glmsmurf(formula = formu, family = gaussian(), data = rent,
                       pen.weights = "glm.stand", lambda = 0.008914)


####
# S3 methods for glmsmurf objects


# Model summary
summary(munich.fit)


# Get coefficients of estimated model
```

```
coef(munich.fit)
# Get coefficients of re-estimated model
coef_reest(munich.fit)


# Plot coefficients of estimated model
plot(munich.fit)
# Plot coefficients of re-estimated model
plot_reest(munich.fit)


# Get deviance of estimated model
deviance(munich.fit)
# Get deviance of re-estimated model
deviance_reest(munich.fit)


# Get fitted values of estimated model
fitted(munich.fit)
# Get fitted values of re-estimated model
fitted_reest(munich.fit)


# Get predicted values of estimated model on scale of linear predictors
predict(munich.fit, type = "link")
# Get predicted values of re-estimated model on scale of linear predictors
predict_reest(munich.fit, type = "link")


# Get deviance residuals of estimated model
residuals(munich.fit, type = "deviance")
# Get deviance residuals of re-estimated model
residuals_reest(munich.fit, type = "deviance")
```

`glmsmurf-class` *Class of Multi-Type Regularized GLMs Fitted Using the SMuRF Algorithm*

Description

The functions `glmsmurf` and `glmsmurf.fit` return objects of the S3 class '`glmsmurf`' which partially inherits from the '`glm`' and '`lm`' classes.

Value

An object of class '`glmsmurf`' is a list with at least following components:

`coefficients`	Coefficients of the estimated model.
`residuals`	Working residuals of the estimated model, see `glm`: $((y_1-\mu_1)/(d\mu/d\eta(\eta_1)), \ldots, (y_n-\mu_n)/(d\mu/d\eta(\eta_n)))$.

`fitted.values` Fitted mean values of the estimated model $(\mu_1, \ldots, \mu_n) = (g^{-1}(\eta_1), \ldots, g^{-1}(\eta_n))$ with g^{-1} the inverse link function.

`rank` Numeric rank of the estimated model, i.e. the number of unique non-zero coefficients.

`family` The used `family` object.

`linear.predictors` Linear fit of the estimated model on the link scale $(\eta_1, \ldots, \eta_n)$.

`deviance` Deviance of the estimated model: minus twice the log-likelihood, up to a constant.

`aic` Akaike Information Criterion of the estimated model: $-2 \times L + 2 \times rank$ with L the log-likelihood.

`bic` Bayesian Information Criterion of the estimated model: $-2 \times L + \ln(n^*) \times rank$ with n^* the number of observations excluding those with weight 0.

`gcv` Generalized Cross-Validation score of the estimated model: $deviance/(n^* \times (1 - rank/n^*)^2)$.

`null.deviance` Deviance of the null model, i.e. the model with only an intercept and offset.

`df.residual` Residual degrees of freedom of the estimated model, i.e. the number of observations (excluding those with weight 0) minus the rank of the estimated model.

`df.null` Residual degrees of freedom for the null model, i.e. the number of observations (excluding those with weight 0) minus the rank of the null model.

`obj.fun` Value of the objective function of the estimated model: minus the regularized scaled log-likelihood of the estimated model.

`weights` The prior weights that were initially supplied. Note that they are called `prior.weights` in the output of `glm`.

`offset` The used offset vector.

`lambda` The used penalty parameter: initially supplied by the user, or selected in-sample, out-of-sample or using cross-validation.

`lambda1` The used penalty parameter for the L_1-penalty in Sparse (Generalized) Fused Lasso or Sparse Graph-Guided Fused Lasso is $\lambda \times \lambda_1$

`lambda2` The used penalty parameter for the L_2-penalty in Group (Generalized) Fused Lasso or Group Graph-Guided Fused Lasso is $\lambda \times \lambda_2$.

`iter` The number of iterations that are performed to fit the model.

`converged` An integer code indicating whether the algorithm converged successfully:

0 Successful convergence.

1 Maximum number of iterations reached.

2 Two subsequent restarts were performed.

3 Low step size (i.e. below 1e-14).

`final.stepsize` Final step size used in the algorithm.

`n.par.cov` List with number of parameters to estimate per predictor (covariate).

`pen.cov` List with penalty type per predictor (covariate).

`group.cov` List with group of each predictor (covariate) for Group Lasso where 0 means no group.

`refcat.cov` List with number of the reference category in the original order of the levels of each predictor (covariate) where 0 indicates no reference category.

`control` The used control list, see `glmsmurf.control`.

Optionally, following elements are also included:

`X` The model matrix, only returned when the argument `x.return` in `glmsmurf` or `glmsmurf.fit` is `TRUE`.

`y` The response vector, only returned when the argument `y.return` in `glmsmurf` or `glmsmurf.fit` is `TRUE`.

`pen.weights` List with the vector of penalty weights per predictor (covariate), only returned when the argument `pen.weights.return` in `glmsmurf` or `glmsmurf.fit` is `TRUE`.

When the model is re-estimated, i.e. `reest = TRUE` in `glmsmurf.control`, the following components are also present:

`glm.reest` Output from the call to `speedglm` or `glm` to fit the re-estimated model.

`coefficients.reest`
Coefficients of the re-estimated model.

`residuals.reest`
Working residuals of the re-estimated model.

`fitted.values.reest`
Fitted mean values of the re-estimated model.

`rank.reest` Numeric rank of the re-estimated model, i.e. the number of unique non-zero re-estimated coefficients.

`linear.predictors.reest`
Linear fit of the re-estimated model on the link scale.

`deviance.reest` Deviance of the re-estimated model.

`aic.reest` AIC of the re-estimated model.

`bic.reest` BIC of the re-estimated model.

`gcv.reest` GCV score of the re-estimated model.

`df.residual.reest`
Residual degrees of freedom of the re-estimated model.

`obj.fun.reest` Value of the objective function of the re-estimated model: minus the regularized scaled log-likelihood of the re-estimated model.

`X.reest` The model matrix used in the re-estimation, only returned when the argument `x.return` in `glmsmurf` or `glmsmurf.fit` is `TRUE`.

When lambda is not given as input but selected in-sample, out-of-sample or using cross-validation, i.e. the `lambda` argument in `glmsmurf` or `glmsmurf.fit` is a string describing the selection method, the following components are also present:

`lambda.method` Method (in-sample, out-of-sample or cross-validation (possibly with the one standard error rule)) and measure (AIC, BIC, GCV score, deviance, MSE or DSS) used to select `lambda`. E.g. `"is.bic"` indicates in-sample selection of lambda with the BIC as measure.

`lambda.vector` Vector of `lambda` values that were considered in the selection process.

`lambda.measures`
List with for each of the relevant measures a matrix containing for each considered value of `lambda` (rows) the measure for the whole data (in-sample), for the validation data (out-of-sample) or per cross-validation fold (cross-validation) (columns).

`lambda.coefficients`
Matrix containing for each considered value of `lambda` (rows) the estimated (when `lambda.reest = FALSE` in `glmsmurf.control`) or re-estimated (when `lambda.reest = TRUE`) coefficients when selecting lambda in-sample or out-of-sample (or using cross-validation with one fold); and `NULL` otherwise.

When the object is output from `glmsmurf`, following elements are also included:

`call` The matched call.

`formula` The supplied formula.

`terms` The `terms` object used.

`contrasts` The contrasts used (when relevant).

`xlevels` The levels of the factors used in fitting (when relevant).

S3 generics

Following S3 generic functions are available for an object of class `"glmsmurf"`:

`coef` Extract coefficients of the estimated model.

`coef_reest` Extract coefficients of the re-estimated model, when available.

`deviance` Extract deviance of the estimated model.

`deviance_reest` Extract deviance of the re-estimated model, when available.

`family` Extract family object.

`fitted` Extract fitted values of the estimated model.

`fitted_reest` Extract fitted values of the re-estimated model, when available.

`plot` Plot coefficients of the estimated model.

`plot_reest` Plot coefficients of the re-estimated model, when available.

`plot_lambda` Plot goodness-of-fit statistics or information criteria as a function of lambda, when lambda is selected in-sample, out-of-sample or using cross-validation.

`predict` Obtain predictions using the estimated model.

`predict_reest` Obtain predictions using the re-estimated model, when available.

`residuals` Extract residuals of the estimated model.

`residuals_reest` Extract residuals of the re-estimated model, when available.

`summary` Print a summary of the estimated model, and of the re-estimated model (when available).

See Also

glmsmurf, glm, lm

Examples

```
## See example(glmsmurf) for examples
```

`glmsmurf.control` *Control Function for Fitting a Multi-Type Regularized GLM Using the SMuRF Algorithm.*

Description

Control function to handle parameters for fitting a multi-type regularized generalized linear model (GLM) using the SMuRF algorithm. The function sets defaults and performs input checks on the provided parameters.

Usage

```
glmsmurf.control(
  epsilon = 1e-08,
  maxiter = 10000,
  step = NULL,
  tau = 0.5,
  reest = TRUE,
  lambda.vector = NULL,
  lambda.min = NULL,
  lambda.max = NULL,
  lambda.length = 50L,
  lambda.reest = FALSE,
  k = 5L,
  oos.prop = 0.2,
  validation.index = NULL,
  ncores = NULL,
  po.ncores = NULL,
  print = FALSE
)
```

Arguments

`epsilon`	Numeric tolerance value for stopping criterion. A numeric strictly larger than 0, default is `1e-8`.
`maxiter`	Maximum number of iterations of the SMuRF algorithm. A numeric larger than or equal to 1, default is `10 000`.
`step`	Initial step size, a numeric strictly larger than 0 or `NULL`. When `NULL` (default), it is equal to `0.1` times the sample size.

tau	Parameter for backtracking the step size. A numeric strictly between 0 and 1, default is 0.5.
reest	A logical indicating if the obtained (reduced) model is re-estimated using speedglm or glm. Default is TRUE.
lambda.vector	Values of lambda to consider when selecting the optimal value of lambda. A vector of strictly positive numerics (which is preferably a decreasing sequence as we make use of warm starts) or NULL (default). When NULL, it is set to an exponential decreasing sequence of length lambda.length between lambda.max and lambda.min.
lambda.min	Minimum value of lambda to consider when selecting the optimal value of lambda. A strictly positive numeric or NULL (default). When NULL, it is equal to 0.0001 times lambda.max. This argument is ignored when lambda.vector is not NULL.
lambda.max	Maximum value of lambda to consider when selecting the optimal value of lambda. A strictly positive numeric larger than lambda.min or NULL (default). In the latter case, lambda.max will be determined based on the used penalty types such that it is one of the smallest values of lambda that results in an intercept-only model. This argument is ignored when lambda.vector is not NULL.
lambda.length	Number of lambda values to consider when selecting the optimal value of lambda. A strictly positive integer, default is 50. This argument is ignored when lambda.vector is not NULL.
lambda.reest	Logical indicating if the re-estimated coefficients are used when selecting lambda, default is FALSE. This argument is only used if reest is TRUE.
k	Number of folds when selecting lambda using cross-validation. A strictly positive integer, default is 5 (i.e. five-fold cross-validation). This number cannot be larger than the number of observations. Note that cross-validation with one fold (k=1) is the same as in-sample selection of lambda.
oos.prop	Proportion of the data that is used as the validation sample when selecting lambda out-of-sample. A numeric strictly between 0 and 1, default is 0.2. This argument is ignored when validation.index is not NULL.
validation.index	Vector containing the row indices of the data matrix corresponding to the observations that are used as the validation sample. This argument is only used when lambda is selected out-of-sample. Default is NULL meaning that randomly 100*oos.prop% of the data are used as validation sample.
ncores	Number of cores used when performing cross-validation. A strictly positive integer or NULL (default). When NULL, max(nc-1,1) cores are used where nc is the number of cores as determined by detectCores.
po.ncores	Number of cores used when computing the proximal operators. A strictly positive integer or NULL (default). When NULL or ncores > 1, po.ncores is set to 1.
print	A logical indicating if intermediate results need to be printed, default is FALSE.

Details

More details on the selection of lambda can be found in the package vignette.

Value

A list with elements named as the arguments.

See Also

Fitting procedures: `glmsmurf` and `glmsmurf.fit` (given design matrix). `glm.control`

Examples

```
## See example(plot_lambda) for examples
```

---

p *Define Individual Subpenalties for a Multi-Type Regularized GLM*

Description

Function used to define regularization terms in a `glmsmurf` model formula.

Usage

```
p(pred1, pred2 = NULL, pen = "lasso", refcat = NULL, group = NULL)
```

Arguments

`pred1`	Name of the predictor used in the regularization term.
`pred2`	Either `NULL` (default) meaning that only one predictor is used in the regularization term, or the name of the second predictor that is used in a 2D Fused Lasso regularization term.
`pen`	Type of penalty for this predictor, one of • `"none"` (no penalty), • `"lasso"` (Lasso), • `"grouplasso"` (Group Lasso), • `"flasso"` (Fused Lasso), • `"gflasso"` (Generalized Fused Lasso), • `"2dflasso"` (2D Fused Lasso), • `"ggflasso"` (Graph-Guided Fused Lasso). Default is `"lasso"`.
`refcat`	Reference level when `pred1` is a factor and `pen` is `"none"`, `"flasso"`, `"gflasso"`, or `"ggflasso"`; otherwise `refcat` is ignored. Default is `NULL` which means that the first level of `pred1` is used as the reference level (if `refcat` is not ignored).
`group`	Group to which the predictor belongs, only used for a Group Lasso penalty. Default is `NULL` which means that predictor does not belong to a group.

Details

Predictors with no penalty, a Lasso penalty or a Group Lasso penalty should be numeric or a factor which can be non-numeric. Predictors with a Fused Lasso, Generalized Fused Lasso, Graph-Guided Fused Lasso or 2D Fused Lasso penalty should be given as a factor which can also be non-numeric. When a predictor is given as a factor, there cannot be any unused levels.

For a predictor with a Fused Lasso penalty, the levels should be ordered from smallest to largest. The first level will be the reference level, but this can be changed using the `refcat` argument.

When `lambda * lambda1 > 0` or `lambda * lambda2 > 0` in `glmsmurf`, no reference level is used for the Fused Lasso, Generalized Fused Lasso and Graph-Guided Fused Lasso penalties, and `refcat` will hence be ignored.

If `pred2` is different from `NULL`, `pen` should be set to `"2dflasso"`, and vice versa. Note that there cannot be any unused levels in the interaction between `pred1` and `pred2`.

When adding an interaction between `pred1` and `pred2` with a 2D Fused Lasso penalty, the 1D effects should also be present in the model and the reference categories for the 1D predictors need to be the respective first levels. The reference level for the 2D predictor will then be the 2D level where it least one of the 1D components is equal to the 1D reference levels. It is also allowed to add binned factors, of predictors that are included in the model, in the interaction. They should have the original predictor name + '.binned' as predictor names. For example: the original predictors 'age' and 'power' are included in the model and the interaction of 'age.binned' and 'power.binned' can also be present in the model formula.

An overview of the different penalty types and their usage can be found in the package vignette.

See Also

`glmsmurf`

Examples

```
# Munich rent data from catdata package
data("rent", package = "catdata")

# The considered predictors are the same as in
# Gertheiss and Tutz (Ann. Appl. Stat., 2010).
# Response is monthly rent per square meter in Euro

# Urban district in Munich
rent$area <- as.factor(rent$area)

# Decade of construction
rent$year <- as.factor(floor(rent$year / 10) * 10)

# Number of rooms
rent$rooms <- as.factor(rent$rooms)

# Quality of the house with levels "fair", "good" and "excellent"
rent$quality <- as.factor(rent$good + 2 * rent$best)
levels(rent$quality) <- c("fair", "good", "excellent")
```

```
# Floor space divided in categories (0, 30), [30, 40), ...,  [130, 140)
sizeClasses <- c(0, seq(30, 140, 10))
rent$size <- as.factor(sizeClasses[findInterval(rent$size, sizeClasses)])

# Is warm water present?
rent$warm <- factor(rent$warm, labels = c("yes", "no"))

# Is central heating present?
rent$central <- factor(rent$central, labels = c("yes", "no"))

# Does the bathroom have tiles?
rent$tiles <- factor(rent$tiles, labels = c("yes", "no"))

# Is there special furniture in the bathroom?
rent$bathextra <- factor(rent$bathextra, labels = c("no", "yes"))

# Is the kitchen well-equipped?
rent$kitchen <- factor(rent$kitchen, labels = c("no", "yes"))



# Create formula with 'rentm' as response variable,
# 'area' with a Generalized Fused Lasso penalty,
# 'year', 'rooms', 'quality' and 'size' with Fused Lasso penalties
# where the reference category for 'year' is changed to 2000,
# 'warm' and 'central' are in one group for the Group Lasso penalty,
# 'tiles' and 'bathextra' are not regularized and
# 'kitchen' has a Lasso penalty
formu <- rentm ~ p(area, pen = "gflasso") +
  p(year, pen = "flasso", refcat = 2000) + p(rooms, pen = "flasso") +
  p(quality, pen = "flasso") + p(size, pen = "flasso") +
  p(warm, pen = "grouplasso", group = 1) + p(central, pen = "grouplasso", group = 1) +
  p(tiles, pen = "none") + bathextra +
  p(kitchen, pen = "lasso")


# Fit a multi-type regularized GLM using the SMuRF algorithm.
# We use standardization adaptive penalty weights based on an initial GLM fit.
munich.fit <- glmsmurf(formula = formu, family = gaussian(), data = rent,
                       pen.weights = "glm.stand", lambda = 0.1)

# Model summary
summary(munich.fit)
```

`plot.glmsmurf` *Plot Coefficients of Estimated Model*

Description

Function to plot the coefficients of the estimated model.

Usage

```
## S3 method for class 'glmsmurf'
plot(x, xlab = "Index", ylab = "Estimated coefficients", basic = FALSE, ...)
```

Arguments

x	An object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
xlab	Label for the x-axis, default is "Index".
ylab	Label for the y-axis, default is "Estimated coefficients".
basic	Logical indicating if the basic lay-out is used for the plot, default is FALSE.
...	Additional arguments for the plot function.

Details

When basic=FALSE, the improved lay-out for the plot is used. Per predictor, groups of equal coefficients are indicated in the same color (up to 8 colors), and zero coefficients are indicated by grey squares.

See Also

plot_reest, coef.glmsmurf, summary.glmsmurf, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

---

plot_lambda *Plot Goodness-of-Fit Statistics or Information Criteria*

Description

Function to plot the goodness-of-fit statistics or information criteria as a function of lambda when lambda is selected in-sample, out-of-sample or using cross-validation.

Usage

```
plot_lambda(x, ...)

## S3 method for class 'glmsmurf'
plot_lambda(
  x,
  xlab = NULL,
  ylab = NULL,
  lambda.opt = TRUE,
  cv1se = TRUE,
```

```
  log.lambda = TRUE,
  ...
)
```

Arguments

x	An object for which the extraction of goodness-of-fit statistics or information criteria is meaningful. E.g. an object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
...	Additional arguments for the plot function.
xlab	Label for the x-axis. The default value is NULL which means that substitute(log(lambda)) is used when log.lambda=TRUE and substitute(lambda) when log.lambda=FALSE.
ylab	Label for the y-axis. The default value is NULL which means that the y-axis label is determined based on method that was used to select lambda.
lambda.opt	Logical indicating if the optimal value of lambda should be indicated on the plot by a vertical dashed line. Default is TRUE.
cv1se	Logical indicating if the standard errors should be indicated on the plot when cross-validation with the one standard error rule is performed (e.g. "cv1se.dev"). Default is TRUE.
log.lambda	Logical indicating if the logarithm of lambda is plotted on the x-axis, default is TRUE.

Details

This plot can only be made when lambda is selected in-sample, out-of-sample or using cross-validation (possibly with the one standard error rule), see the lambda argument of glmsmurf.

See Also

glmsmurf, glmsmurf-class

Examples

```
# Munich rent data from catdata package
data("rent", package = "catdata")

# The considered predictors are the same as in
# Gertheiss and Tutz (Ann. Appl. Stat., 2010).
# Response is monthly rent per square meter in Euro

# Urban district in Munich
rent$area <- as.factor(rent$area)

# Decade of construction
rent$year <- as.factor(floor(rent$year / 10) * 10)

# Number of rooms
```

```
rent$rooms <- as.factor(rent$rooms)

# Quality of the house with levels "fair", "good" and "excellent"
rent$quality <- as.factor(rent$good + 2 * rent$best)
levels(rent$quality) <- c("fair", "good", "excellent")

# Floor space divided in categories (0, 30), [30, 40), ...,  [130, 140)
sizeClasses <- c(0, seq(30, 140, 10))
rent$size <- as.factor(sizeClasses[findInterval(rent$size, sizeClasses)])

# Is warm water present?
rent$warm <- factor(rent$warm, labels = c("yes", "no"))

# Is central heating present?
rent$central <- factor(rent$central, labels = c("yes", "no"))

# Does the bathroom have tiles?
rent$tiles <- factor(rent$tiles, labels = c("yes", "no"))

# Is there special furniture in the bathroom?
rent$bathextra <- factor(rent$bathextra, labels = c("no", "yes"))

# Is the kitchen well-equipped?
rent$kitchen <- factor(rent$kitchen, labels = c("no", "yes"))



# Create formula with 'rentm' as response variable,
# 'area' with a Generalized Fused Lasso penalty,
# 'year', 'rooms', 'quality' and 'size' with Fused Lasso penalties,
# and the other predictors with Lasso penalties.
formu <- rentm ~ p(area, pen = "gflasso") +
  p(year, pen = "flasso") + p(rooms, pen = "flasso") +
  p(quality, pen = "flasso") + p(size, pen = "flasso") +
  p(warm, pen = "lasso") + p(central, pen = "lasso") +
  p(tiles, pen = "lasso") + p(bathextra, pen = "lasso") +
  p(kitchen, pen = "lasso")

# Fit a multi-type regularized GLM using the SMuRF algorithm and select the optimal value of lambda
# using cross-validation (with the deviance as loss measure and the one standard error rule).
# We use standardization adaptive penalty weights based on an initial GLM fit.
# The number of values of lambda to consider in cross-validation is
# set to 25 using the control argument (default is 50).
munich.fit.cv <- glmsmurf(formula = formu, family = gaussian(), data = rent,
                          pen.weights = "glm.stand", lambda = "cv1se.dev",
                          control = list(lambda.length = 25L, ncores = 1L))


# Plot average deviance over cross-validation folds as a function of the logarithm of lambda
plot_lambda(munich.fit.cv)
# Zoomed plot
plot_lambda(munich.fit.cv, xlim = c(-8, -4), ylim = c(1550, 1750))
```

plot_reest *Plot Coefficients of Re-estimated Model*

Description

Function to plot the coefficients of the re-estimated model.

Usage

```
plot_reest(x, ...)

## S3 method for class 'glmsmurf'
plot_reest(
  x,
  xlab = "Index",
  ylab = "Re-estimated coefficients",
  basic = FALSE,
  ...
)
```

Arguments

x	An object for which the extraction of model coefficients is meaningful. E.g. an object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
...	Additional arguments for the plot function.
xlab	Label for the x-axis, default is "Index".
ylab	Label for the y-axis, default is "Re-estimated coefficients".
basic	Logical indicating if the basic lay-out is used for the plot, default is FALSE.

Details

When the re-estimated model is not included in x, the coefficients of the estimated model in x are plotted with a warning.

See plot.glmsmurf for more details.

See Also

plot.glmsmurf, coef_reest, summary.glmsmurf, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

predict.glmsmurf *Predictions Using Estimated Model*

Description

Function to obtain predictions using the estimated model.

Usage

```
## S3 method for class 'glmsmurf'
predict(
  object,
  newdata = NULL,
  newoffset = NULL,
  type = c("link", "response", "terms"),
  ...
)
```

Arguments

object	An object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
newdata	Optionally, a data frame containing the predictors used in the prediction. This can only be used when object contains a formula. When newdata is omitted, the predictions are based on the data used to fit the model in object.
newoffset	Optionally, a vector containing a new offset to be used in the prediction. When newoffset is omitted, the predictions use the offset which was used to fit the model in object.
type	Type of prediction. The default is on the scale of the linear predictors ("link"). Another option is on the scale of the response variable ("response"). For type "terms" a matrix containing the fitted values of each term in the model, on the linear predictor scale, is returned.
...	Additional arguments which are currently ignored.

Value

A vector containing the predicted values using the estimated model in object.

See Also

predict_reest, predict.glm, predict, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

predict_reest *Predictions Using Re-estimated Model*

Description

Function to obtain predictions using the re-estimated model.

Usage

```
predict_reest(object, ...)

## S3 method for class 'glmsmurf'
predict_reest(
  object,
  newdata = NULL,
  newoffset = NULL,
  type = c("link", "response", "terms"),
  ...
)
```

Arguments

object	An object for which predictions are meaningful. E.g. an object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
...	Additional arguments which are currently ignored.
newdata	Optionally, a data frame containing the predictors used in the prediction. This can only be used when object contains a formula. When newdata is omitted, the predictions are based on the data used to fit the model in object.
newoffset	Optionally, a vector containing a new offset to be used in the prediction. When newoffset is omitted, the predictions use the offset which was used to fit the model in object.
type	Type of prediction. The default is on the scale of the linear predictors ("link"). Another option is on the scale of the response variable ("response"). For type "terms" a matrix containing the fitted values of each term in the model, on the linear predictor scale, is returned.

Value

A vector containing the predicted values using the re-estimated model in object, when this is available, or, otherwise, the predicted values using the estimated model in object with a warning.

See Also

predict.glmsmurf, predict.glm, predict, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

`residuals.glmsmurf` *Residuals of Estimated Model*

Description

Function to extract the residuals of the estimated model. `resid` is an *alias* for it.

Usage

```
## S3 method for class 'glmsmurf'
residuals(
  object,
  type = c("deviance", "pearson", "working", "response", "partial"),
  ...
)

## S3 method for class 'glmsmurf'
resid(
  object,
  type = c("deviance", "pearson", "working", "response", "partial"),
  ...
)
```

Arguments

`object`	An object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
`type`	Type of residuals that should be returned. One of "deviance" (default), "pearson", "working", "response" or "partial".
`...`	Additional arguments which are currently ignored.

Details

See glm.summaries for an overview of the different types of residuals.

Value

A vector containing the residuals of the estimated model in `object`.

See Also

residuals_reest, residuals, glm.summaries, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

`residuals_reest` *Residuals of Re-estimated Model*

Description

Function to extract the residuals of the re-estimated model. `resid_reest` is an *alias* for it.

Usage

```
residuals_reest(object, ...)

## S3 method for class 'glmsmurf'
residuals_reest(
  object,
  type = c("deviance", "pearson", "working", "response", "partial"),
  ...
)

resid_reest(object, ...)

## S3 method for class 'glmsmurf'
resid_reest(
  object,
  type = c("deviance", "pearson", "working", "response", "partial"),
  ...
)
```

Arguments

`object`	An object for which the extraction of model residuals is meaningful. E.g. an object of class '`glmsmurf`', typically the result of a call to `glmsmurf` or `glmsmurf.fit`.
`...`	Additional arguments which are currently ignored.
`type`	Type of residuals that should be returned. One of `"deviance"` (default), `"pearson"`, `"working"`, `"response"` or `"partial"`.

Details

See `glm.summaries` for an overview of the different types of residuals.

Value

A vector containing the residuals of the re-estimated model in `object` when they are available, or, otherwise, the residuals of the estimated model in `object` with a warning.

See Also

residuals.glmsmurf, residuals, glm.summaries, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

---

summary.glmsmurf *Summary of a Multi-Type Regularized GLM Fitted Using the SMuRF Algorithm*

Description

Function to print a summary of a glmsmurf-object.

Usage

```
## S3 method for class 'glmsmurf'
summary(object, digits = 3L, ...)
```

Arguments

object	An object of class 'glmsmurf', typically the result of a call to glmsmurf or glmsmurf.fit.
digits	The number of significant digits used when printing, default is 3.
...	Additional arguments which are currently ignored.

See Also

summary.glm, glmsmurf, glmsmurf-class

Examples

```
## See example(glmsmurf) for examples
```

Index