# Package 'lassogrp'

October 28, 2009

**Type** Package

**Title** Lasso Regression including group lasso and adaptive lasso

**Version** 1.0

**Date** 2009-10-15

**Author** Lukas Meier and Werner A. Stahel, ETH Zurich

**Maintainer** Werner A. Stahel, ETH Zurich,<stahel@stat.math.ethz.ch>

**Description** Lasso Regression including group lasso and adaptive lasso

**Depends** methods

**License** GPL

**LazyLoad** yes

## R topics documented:

`lassogrp-package` *Lasso Regression including group lasso and adaptive lasso*

## Description

lassogrp implements the Lasso regression method for several regression types (ordinary, logistic, Poisson and user-provided models). It offers the group lasso which is suitable when factors appear in the model. It also provides a user-friendly way to perform the adaptive lasso, which consists of two calls to lasso, the second one using a weighted L1 penalization term, where the weights depend on the results of the first call.

## Details

| | |
|---|---|
| Package: | lassogrp |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2009-10-15 |
| License: | GPL |
| LazyLoad: | yes |

This package replaces the grplasso package and extends its functionality.

The package includes methods for printing and plotting as well as a cross validation function and functions to extract specified fits from the lasso results.

It is in development.

## Author(s)

Werner A. Stahel, ETH Zurich, <stahel@stat.math.ethz.ch>, based on code of Lukas Meier

## References

Lukas Meier, Sara van de Geer and Peter B\"uhlmann (2008), *The Group Lasso for Logistic Regression*, Journal of the Royal Statistical Society, 70 (1), 53 - 71, see also http://stat.ethz.ch/~meier/logistic-grouplasso.php

## See Also

```
library(lars)
```

## Examples

```
data(splice)
fit.splice <- lasso(y ~ ., data = splice, model = 'binomial')
```

`cv.lasso` *Cross Validation of a Lasso Fit*

## Description

Calculation of cross validation for a `lassogrp` object.

## Usage

```
cv.lasso(object, blocks = 10, trace = FALSE,
  control = lassoControl(trace = 0), env = globalenv(),
  plot.it = NULL, se = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `object` | a `lassogrp` object |
| `blocks` | number of blocks to be used in block crossvalidation, or name of variable defining the blocks |
| `trace` | logical: Should completion of each cross validation cycle be indicated? |
| `control` | see `?lassoControl` |
| `env` | environment for getting blocks |
| `plot.it` | logical: should results be plotted? |
| `se` | logical: should cross validation standard errors be plotted? |
| `...` | further arguments passed to `plot` |

## Details

If the blocks are determined by a factor which appears in the model, ...

## Value

`rmse`

`rmse.see`

`rmse.blocks`

`fitted`

`lambda`
`blocksinmodel`

## Author(s)

Werner Stahel, `<stahel@stat.math.ethz.ch>`

## See Also

`cv.lars` in package `lars`

### Examples

```
data(asphalt)
rr <- lasso(log10(RUT)~log10(VISC)+ASPH+BASE+FINES+VOIDS+RUN, data=asphalt)
cv.lasso(rr)
```

---

## descr *Define and obtain the descr or tit attribute*

---

### Description

The attributes `descr` and `tit` describe an object, typically a data frame or a model. `tit` should be a short description (title), `descr` should contain all documentation useful to identify the origin and the changes made to the object.
The `descr` and `tit` functions set them and extract these attributes.

### Usage

```
descr(x)
tit(x)
descr(x) <- value
tit(x) <- value
```

### Arguments

| | |
|---|---|
| `x` | object to which the `descr` or `tit` attribute should be attached or from which it is obtained |
| `value` | character vector (`descr`) or string (`tit`) to be stored |

### Details

Plotting and printing functions may search for the `tit` attribute or even for the `descr` attribute, depending on `c.env$docout`.

`descr(x) <- text` will append the existing `descr(x)` text to the new one unless the first element of text equals `"^"`, whereas `tit(x) <- string` replaces `tit(x)`.

### Value

`descr` and `tit` return the respective attributes of object `x`

### Author(s)

Werner A. Stahel, ETH Zurich

### Examples

```
data(asphalt)
tit(asphalt)
descr(asphalt)
descr(asphalt) <- "I will use this dataset in class soon."
descr(asphalt)
```

`extract.lassogrp` *Extract Regression Results from a Lasso Fit*

## Description

Extract several fits from a lasso fit table.

## Usage

```
extract.lassogrp(object, i = NULL, lambda = NULL, fitfun = "lm", ...)

object[i]
```

## Arguments

| | |
|---|---|
| `object` | an object with class `lassogrp` |
| `i` | a single index for `extract.lassogrp`, or a vector of indices for the simple subset operator '`[i]`' |
| `lambda` | alternatively to specifying `i`, a lambda value may be specified. |
| `fitfun` | Fitting function that determines the structure of the return value. Note that the coefficients and ... will be taken from `object`. |
| `...` | additional arguments passed to ... |

## Details

`extract.lassogrp` generates an object of a regression class like `lm` or `regr`. This is useful for applying the respective plot and print methods to the lasso fit.

## Value

`extract.lassogrp`: object of class 'lassofit' inheriting from the class specified by `fitfun`.

`x[i]`: an object of class 'lassogrp' containing only the specified fits, i.e. all the information corresponding to these fits.

## Author(s)

Werner Stahel, <stahel@stat.math.ethz.ch>

## Examples

```
data(asphalt)
rr <- lasso(log10(RUT)~log10(VISC)+ASPH+BASE+FINES+VOIDS+RUN,
  data=asphalt)
rr[c(1,10,15)]
extract.lassogrp(rr, lambda=2.5)
```

`lambdamax` *Lasso: get the Maximal lambda*

## Description

Calculates the maximal value of the weight lambda of the L1 penalty term in a Lasso regression. For values >= this value, the null model will be obtained as the result of the penalized regression.

## Usage

```
lambdamax(x, ...)

## S3 method for class 'formula'
## S3 method for class 'formula':
lambdamax(formula, nonpen = ~1, data,
  weights, subset, na.action,
  coef.init, penscale = sqrt, model = LogReg(), standardize = TRUE,
  contrasts = NULL, nlminb.opt = list(), ...)

## S3 default method
## Default S3 method:
lambdamax(x, y, index, weights = NULL,
  offset = rep(0, length(y)), coef.init = rep(0, ncol(x)),
  penscale = sqrt, model = LogReg(), standardize = TRUE,
  nlminb.opt = list(), ...)
```

## Arguments

| | |
|---|---|
| `x` | design matrix (including intercept) |
| `y` | response vector |
| `formula` | `formula` of the penalized variables. The response has to be on the left hand side of '~'. |
| `nonpen` | `formula` of the nonpenalized variables. This will be added to the `formula` argument above and doesn't need to have the response on the left hand side. |
| `data` | `data.frame` containing the variables in the model. |
| `index` | vector which defines the grouping of the variables. Components sharing the same number build a group. Non-penalized coefficients are marked with `NA`. |
| `weights` | vector of observation weights. |
| `subset` | an optional vector specifying a subset of observations to be used in the fitting process. |
| `na.action` | a function which indicates what should happen when the data contain 'NA's. |
| `offset` | vector of offset values. |
| `coef.init` | initial parameter vector. Penalized groups are discarded. |
| `penscale` | rescaling function to adjust the value of the penalty parameter to the degrees of freedom of the parameter group. See the reference below. |
| `model` | an object of class `lassoModel` implementing the negative log-likelihood, gradient, hessian etc. See `lassoModel` for more details. |

| | |
|---|---|
| standardize | logical. If true, the design matrix will be blockwise orthonormalized, such that for each block $X^T X = n1$ (*after* possible centering). |
| contrasts | an (optional) list with the contrasts for the factors in the model. |
| nlminb.opt | arguments to be supplied to nlminb. |
| ... | additional arguments to be passed to the functions defined in model. |

## Details

Uses nlminb to optimize the non-penalized parameters.

## Value

Numerical value of the maximal lambda

## Author(s)

Lukas Meier, Seminar f. Statistik, ETH Zurich

## Examples

```
data(splice)
lambdamax(y ~ ., data = splice, model = LogReg(), center = TRUE,
          standardize = TRUE)
```

---

# lasso *Fit a Model with Lasso*

---

## Description

Functions that fit a model by (Group-) lasso, including adaptive version

## Usage

```
lasso(x, ...)

## S3 method for class 'formula':
## S3 method for class 'formula':
lasso(x, data, subset, weights, na.action,
  model = "gaussian", offset, nonpen = ~1, lambda = NULL, lstep = 21,
  adaptive = FALSE, cv.function = cv.lasso,
  contrasts = NULL, save.x = TRUE, control = lassoControl(), ...)

## S3 method for class 'lassogrp':
## S3 method for class 'lassogrp':
lasso(x, lambda = NULL, lstep = 21, cv = NULL,
  adaptcoef = NULL, adaptlambda = NULL, ...)

## Default S3 method:
## Default S3 method:
lasso(x, y, index, subset, model = "gaussian",
  lambda = NULL, lstep = 21, adaptive = FALSE, cv.function = cv.lasso,
  save.x = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | for `formula` method: model formula for the penalized terms<br>for `default` method: design matrix, including column for intercept<br>for `lassogrp` method: a 'lassogrp' object i.e., the result of a first call to `lasso` |
| data | `data.frame` containing the variables in the model. |
| y | response variable (for `default` method) |
| index | a vector indicating which carriers should be penalized by the L1 term. This is usually obtained from calling `lassogrpModelmatrix`. See Details. (For `default` method) |
| subset, weights, na.action | as in other model fitting functions |
| model | type of model to be fitted: Either one of<br>**"gaussian"** ordinary linear model,<br>**"binomial"** logistic regression,<br>**"poisson"** Poisson regression, or<br>a model generated by `lassoModel` |
| offset | vector of offset values; needs to have the same length as the response vector. May be useful in logistic and Poisson regression. |
| nonpen | formula defining the model terms that must not be included in the lasso penalty term |
| lambda | vector of scaling factors of the lasso penalty term |
| lstep | number of lambda values to be chosen if `lambda` is not provided |
| cv.function | function used for cross validation |
| cv | results of cross validation, if available. |
| adaptive | logical: should the adaptive lasso be used? If TRUE, the lasso will be called twice, first in the regular mode, then adaptive to the results of the first call. (For the `formula` method) |
| adaptcoef | inverse weights for the coefficients, used for the adaptive lasso. By default, they are obtained as the coefficients of the result of an earlier call (`object$coefficients` (for `lassogrp` method), or such a first call is invoked (for `formula` method). |
| adaptlambda | lambda value used to extract the coefficients for `adaptcoef`. Either a lambda value or a negative integer, in which case the coefficients corresponding to the `abs(adaptlambda)` th lambda of `object`. If `adaptlambda` is null, it will be selected by cross validation. |
| contrasts | an optional list. See the 'contrasts.arg' of 'model.matrix.default'. |
| save.x | logical: should the model matrix be stored in the return value? |
| control | list of items to control the algorithm, see `lassoControl` |
| ... | further arguments, passed to `lassogrp`:<br>**penscale** rescaling function to adjust the value of the penalty parameter to the degrees of freedom of the parameter group. See the reference below.<br>**center** logical. If true, the columns of the design matrix will be centered (except a possible intercept column).<br>**standardize** logical. If true, the design matrix will be blockwise orthonormalized such that for each block $X^T X = n1$ (*after* possible centering). |

## Details

The `index` defines the groups of carriers and whether they are included in the L1 penalization term. There is an element in `index` for each carrier (column of the model.matrix). Carriers `j` with positive `index[j]` are included in the penalization. Elements sharing the same `index[j]` value are a group in the sense of the group lasso, that is, their coefficients will be included in the L1 term as `sqrt(sum(coef^2))`.

When using `grplasso.formula`, the grouping of the variables is derived from the type of the variables: The dummy variables of a factor will be automatically treated as a group.

The lasso optimization process starts using the largest value of `lambda` as penalty parameter $\lambda$. Once fitted, the next largest element of `lambda` is used as penalty parameter with starting values defined as the (fitted) coefficient vector based on the previous component of `lambda`.

## Value

A 'lassogrp' object is returned, for which `coef`, `print`, `plot` and `predict` methods exist.

| | |
|---|---|
| `coefficients` | coefficients with respect to the *original* input variables (even if `standardize = TRUE` is used for fitting). |
| `norms.pen` | single terms of the L1 penalty term |
| `nloglik` | log likelihood |
| `fn.val` | |
| `fitted` | fitted values (response type) |
| `linear.predictors` | linear predictors |
| `lambda` | vector of lambda values where coefficients were calculated. |
| `index` | grouping index vector. |
| `...` | and further components, see `names(...)` |

## Author(s)

Lukas Meier and Werner Stahel, <stahel@stat.math.ethz.ch>

## References

Lukas Meier, Sara van de Geer and Peter B\"uhlmann (2008), *The Group Lasso for Logistic Regression*, Journal of the Royal Statistical Society, 70 (1), 53 - 71, see also http://stat.ethz.ch/~meier/logistic-grouplasso.php

## Examples

```
## Lasso for asphalt example
data(asphalt)
rr <- lasso(log10(RUT)~log10(VISC)+ASPH+BASE+FINES+VOIDS+RUN,
  data=asphalt)
rr
names(rr)

## Use the Logistic Group Lasso on the splice data set
data(splice)

## Define a list with the contrasts of the factors
```

```
contr <- rep(list("contr.sum"), ncol(splice) - 1)
names(contr) <- names(splice)[-1]

## Fit a logistic model
fit.splice <-
  lasso(y ~ ., data = splice, model = 'binomial', lambda = 20,
  contrasts = contr)
fit.splice

## Perform the Logistic Group Lasso on a random dataset
set.seed(79)
n <- 50  ## observations
p <- 4   ## variables

## First variable (intercept) not penalized, two groups having 2 degrees
## of freedom each

index <- c(0, 2, 2, 3, 3)

## Create a random design matrix, including the intercept (first column)
x <- cbind(1, matrix(rnorm(p * n), nrow = n))
colnames(x) <- c("Intercept", paste("X", 1:4, sep = ""))

truec <- c(0, 2.1, -1.8, 0, 0)
prob <- 1 / (1 + exp(-x %*% truec))
mean(pmin(prob, 1 - prob)) ## Bayes risk
y <- rbinom(n, size = 1, prob = prob) ## binary response vector

## Use a multiplicative grid for the penalty parameter lambda, starting
## at the maximal lambda value
lambda <- lambdamax(x, y = y, index = index, penscale = sqrt,
                    model = LogReg()) * 0.5^(0:5)

## Fit the solution path on the lambda grid
fit <- lasso(x, y = y, index = index, lambda = lambda, model = 'binomial',
                control = lassoControl(update.hess = "lambda", trace = 0))

## Plot coefficient paths
plot(fit)
```

---

`lassoControl` *Options for the Group Lasso Algorithm*

---

## Description

Definition of options such as bounds on the Hessian, convergence criteria and output management for the Group Lasso algorithm.

## Usage

```
lassoControl(save.x = FALSE, save.y = TRUE,
             update.hess = c("lambda", "always"), update.every = 3,
             inner.loops = 10, line.search = TRUE, max.iter = 500,
             tol = 5 * 10^-8, lower = 10^-2, upper = Inf, beta = 0.5,
             sigma = 0.1, trace = 0)
```

## Arguments

| | |
|---|---|
| `save.x` | a logical indicating whether the design matrix should be saved. |
| `save.y` | a logical indicating whether the response should be saved. |
| `update.hess` | should the hessian be updated in each iteration ("always")? update.hess = "lambda" will update the Hessian once for each component of the penalty parameter "lambda" based on the parameter estimates corresponding to the previous value of the penalty parameter. |
| `update.every` | Only used if update.hess = "lambda". E.g. set to 3 if you want to update the Hessian only every third grid point. |
| `inner.loops` | How many loops should be done (at maximum) when solving only the active set (without considering the remaining predictors). Useful if the number of predictors is large. Set to 0 if no inner loops should be performed. |
| `line.search` | Should line searches be performed? |
| `max.iter` | Maximal number of loops through all groups |
| `tol` | convergence tolerance; the smaller the more precise, see details below. |
| `lower` | lower bound for the diagonal approximation of the corresponding block submatrix of the Hessian of the negative log-likelihood function. |
| `upper` | upper bound for the diagonal approximation of the corresponding block submatrix of the Hessian of the negative log-likelihood function. |
| `beta` | scaling factor $\beta < 1$ of the Armijo line search. |
| `sigma` | $0 < \sigma < 1$ used in the Armijo line search. |
| `trace` | integer. `0` omits any output, `1` prints the current lambda value, `2` prints the improvement in the objective function after each sweep through all the parameter groups and additional information. |

## Details

For the convergence criteria see chapter 8.2.3.2 of Gill et al. (1981).

## Value

An object of class `lassoControl`.

## References

Philip E. Gill, Walter Murray and Margaret H. Wright (1981) *Practical Optimization*, Academic Press.

Dimitri P. Bertsekas (2003) *Nonlinear Programming*, Athena Scientific.

`lassogrpModelmatrix` *Model Matrix for Lasso*

## Description

Generates the model matrix and adds information about variables to be penalized or non-penalized by the L1 term in the lasso fitting. The user will rarely call this function.

## Usage

```
lassogrpModelmatrix(m, formula, nonpen = ~1, data, weights, subset,
  na.action, contrasts, env)
```

## Arguments

| | |
|---|---|
| `m` | call to the lasso function |
| `formula` | model formula |
| `nonpen` | formula defining the non-penalized variables |
| `data` | data.frame in which the variables are found |
| `weights` | as in lm |
| `subset` | as in lm |
| `na.action` | as in lm |
| `contrasts` | as in lm |
| `env` | environment in which the formula is evaluated |

## Details

This function generates the model.matrix and the 'index' vector which stores the information about penalization

## Value

| | |
|---|---|
| `x` | model.matrix |
| `y` | response |
| `w` | weights |
| `off` | offset |
| `mf` | model.frame |
| `index` | vector of length `ncol(x)` determining groups for penalization. Negative values correspond to non-penalized variables. |

## Author(s)

Lukas Meier, modified by Werner Stahel<stahel@stat.math.ethz.ch>

## Examples

```
## not to be called by the user. Is called from function lasso.
```

## lassoModel *Lasso Models*

### Description

Generates models to be used for the (group) lasso algorithm.

### Usage

```
lassoModel(invlink, link, nloglik, ngradient, nhessian, check,
           name = "user-specified", comment = "user-specified")
LogReg()
LinReg()
PoissReg()
```

### Arguments

| | |
|---|---|
| `invlink` | a function with arguments `eta` implementing the inverse link function. |
| `link` | a function with arguments `mu` implementing the link function. |
| `nloglik` | a function with arguments `y`, `mu` and `weights` implementing the *negative* log-likelihood function. |
| `ngradient` | a function with arguments `x`, `y`, `mu` and `weights` implementing the *negative* gradient of the log-likelihood function. |
| `nhessian` | a function with arguments `x`, `mu` and `weights` implementing the *negative* hessian of the log-likelihood function. |
| `check` | a function with argument `y` to check whether the response has the correct format. |
| `name` | a character name |
| `comment` | a character comment |

### Value

An object of class `lassoModel`.

### Examples

```
LogReg()
```

## plot.lassogrp *Plotting Method for 'lassogrp' objects*

### Description

Plots either the traces of the coefficients, the penalty of the coefficient groups, or the penalty terms of the whole model obtained by lasso regression as functions of the penalty weight lambda

## Usage

```
## S3 method for class 'lassogrp':
plot(x, type = c("norms", "coefficients", "criteria"),
  col = NULL, lty = NULL, mar = NULL, main = NULL,
  cv = NULL, se = TRUE, ylim = NULL, legend = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | object of class 'lassogrp' containing the results of fitting a model by lasso. |
| type | character string defining the type of results to be plotted:<br>**'norms'** For simple (non-grouped) regressors, show the absolute value of the estimated coefficient. For grouped variables, show the (L2) norms of the coefficient vector. In both cases, this is the contribution to the L1 penalty term of the (grouped) lasso.<br>**'coefficients'** the estimated coefficients<br>**'criteria'** the (average) log-likelihood and the L1 penalty term |
| col, lty | colors and line types for the curves. If 'type' is 'criteria', they are used as follows<br>**[1** ] for the log-likelihood,<br>**[2** ] for the cross-validated root mse (!!!)<br>**[3** ] for the cross-validated standard error band of the root mse (!!!)<br>**[4** ] for the L1 penalty term. |
| mar | plot margins |
| main | main title of the plot |
| cv | results of cross validation (see cv.lasso) to be plotted, or logical: if TRUE, cv.lasso will be called and cross validation results will be plotted |
| se | logical: Should the cross validation standard error band be plotted? |
| ylim | limits of the vertical plotting axis. Defaults to the range of the values to be plotted except for the values for lambda==0 (if these are contained in 'x') |
| legend | logical: should a legend be displayed? |
| ... | further arguments passed to 'matplot' |

## Value

none

## Note

For model other than ordinary regression, the cross validation results are not yet treated in a sensible way

## Author(s)

Werner A. Stahel, ETH Zurich, <stahel@stat.math.ethz.ch>

## Examples

```
data(asphalt)
rr <- lasso(log10(RUT)~log10(VISC)+ASPH+BASE+FINES+VOIDS+RUN,
  data=asphalt)
plot(rr)
plot(rr, type='criteria')
```

`predict.lassogrp` *Fitted values and Prediction for 'lassogrp' Objects*

## Description

Calculate fitted values or predictions for 'lassogrp' objects

## Usage

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

## S3 method for class 'lassogrp':
predict(object, newdata = NULL,
  type = c("link", "response"), na.action = na.pass, ...)
```

## Arguments

| | |
|---|---|
| `object` | an object of class 'lassogrp' containing a lasso fit |
| `newdata` | data.frame containing the new regressors for which the predictions are required |
| `type` | type of prediction |
| `na.action` | action for missing values, see `?lm` |
| `...` | not used (needed for compatibility) |

## Details

The fitted values are equal to response predictions for the observations used for fitting

## Value

matrix of fitted or predicted values. Columns correspond to lambda values, rows, to observations in 'newdata'

## Author(s)

Werner Stahel, `<stahel@stat.math.ethz.ch>`

## Examples

```
data(asphalt)
dd <- asphalt
rr <- lasso(log10(RUT)~log10(VISC)+ASPH+BASE+FINES+VOIDS+RUN, data=dd)
fitted(rr)
predict(rr, newdata=
  data.frame(VISC=2, ASPH=5, BASE=5, VOIDS=5, FINES=70, RUN=1))
```

`print.lassogrp` *Print lassogrp or lassofit Objects*

## Description

Printing method for lassogrp or lassofit Objects

## Usage

```
# lassogrp objects:
## S3 method for class 'lassogrp':
print(x, coefficients = TRUE, doc = options("doc")[[1]], ...)
# lassofit objects:
## S3 method for class 'lassofit':
print(x, residuals = FALSE, doc = options("doc")[[1]], ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to be printed |
| `coefficients` | logical: should coefficients be shown? |
| `residuals` | logical: should a summary of residuals be shown? |
| `doc` | should documentation of `x` be printed (if any)? |
| | **0** do not show documentation |
| | **1** show `attr(x,'tit')` |
| | **2** show `attr(x,'descr') in addition` |
| | \item...further arguments, passed to print |
| | none |
| | Werner A. Stahel, ETH Zurich, `<stahel@stat.math.ethz.ch>` |
| | data(ashpalt) rr <- lasso(log10(RUT)~log10(VISC)+ASPH+BASE+FINES+VOIDS+RUN, data=dd) print(rr, coefficients=FALSE) |
| | rrf <- extract.lassogrp(rr, 13) print(rrf, residuals=TRUE) |
| | misc |

# Index