Package ‘glmnetUtils’

October 13, 2022

Type Package
Version 1.1.8
Title Utilities for ‘Glmnet’
Description Provides a formula interface for the 'glmnet' package for elasticnet regression, a method for cross-validating the alpha parameter, and other quality-of-life tools.
Imports stats, graphics, grDevices, glmnet, parallel, Matrix
Suggests knitr, rmarkdown, MASS, doParallel, testthat
NeedsCompilation no
VignetteBuilder knitr
Copyright Microsoft
License GPL-2
URL https://github.com/hongooi73/glmnetUtils
Collate 'glmnetUtils.r' 'cvGlmnetFormula.r' 'cvaGlmnetFormula.r'
          'glmnetFormula.r' 'makeModelComponents.r'
RoxygenNote 7.1.1
Author Microsoft [cph],
            Hong Ooi [aut, cre]
Maintainer Hong Ooi <hongooi73@gmail.com>
Repository CRAN
Date/Publication 2021-02-25 15:30:02 UTC

R topics documented:

  cv.glmnet .......................................................... 2
  cva.glmnet ......................................................... 4
  glmnet .......................................................... 9
  glmnet.model.matrix ............................................ 12
  glmnetUtils ..................................................... 13

Index 14
cv.glmnet

Formula interface for elastic net cross-validation with cv.glmnet

Description

Formula interface for elastic net cross-validation with cv.glmnet

Usage

cv.glmnet(x, ...)

## Default S3 method:
cv.glmnet(x, y, ...)

## S3 method for class 'formula'
cv.glmnet(formula, data, alpha = 1, nfolds = 10, ..., weights = NULL, offset = NULL, subset = NULL, na.action = getOption("na.action"), drop.unused.levels = FALSE, xlev = NULL, sparse = FALSE, use.model.frame = FALSE, gamma = c(0, 0.25, 0.5, 0.75, 1), relax = FALSE)

## S3 method for class 'cv.glmnet.formula'
predict(object, newdata, na.action = na.pass, ...)

## S3 method for class 'cv.glmnet.formula'
coef(object, ...)

## S3 method for class 'cv.relaxed.formula'
predict(object, newdata, na.action = na.pass, ...)

## S3 method for class 'cv.glmnet.formula'
coef(object, ...)
cv.glmnet

Arguments

\[ x \]
For the default method, a matrix of predictor variables.

\[ \ldots \]
For \texttt{cv.glmnet.formula} and \texttt{cv.glmnet.default}, other arguments to be passed to \texttt{glmnet::cv.glmnet}; for the predict and \texttt{coef} methods, arguments to be passed to their counterparts in package \texttt{glmnet}.

\[ y \]
For the default method, a response vector or matrix (for a multinomial response).

\[ \text{formula} \]
A model formula; interaction terms are allowed and will be expanded per the usual rules for linear models.

\[ \text{data} \]
A data frame or matrix containing the variables in the formula.

\[ \text{alpha} \]
The elastic net mixing parameter. See \texttt{glmnet::glmnet} for more details.

\[ \text{n folds} \]
The number of crossvalidation folds to use. See \texttt{glmnet::cv.glmnet} for more details.

\[ \text{weights} \]
An optional vector of case weights to be used in the fitting process. If missing, defaults to an unweighted fit.

\[ \text{offset} \]
An optional vector of offsets, an \textit{a priori} known component to be included in the linear predictor.

\[ \text{subset} \]
An optional vector specifying the subset of observations to be used to fit the model.

\[ \text{na.action} \]
A function which indicates what should happen when the data contains missing values. For the \texttt{predict} method, \texttt{na.action = na.pass} will predict missing values with \texttt{NA}; \texttt{na.omit} or \texttt{na.exclude} will drop them.

\[ \text{drop.unused.levels} \]
Should factors have unused levels dropped? Defaults to \texttt{FALSE}.

\[ \text{xlev} \]
A named list of character vectors giving the full set of levels to be assumed for each factor.

\[ \text{sparse} \]
Should the model matrix be in sparse format? This can save memory when dealing with many factor variables, each with many levels.

\[ \text{use.model.frame} \]
Should the base \texttt{model.frame} function be used when constructing the model matrix? This is the standard method that most R modelling functions use, but has some disadvantages. The default is to avoid \texttt{model.frame} and construct the model matrix term-by-term; see \texttt{discussion}.

\[ \text{gamma} \]
For \texttt{cv.glmnet.formula}, the values of the parameter for mixing the relaxed (non-regularised) fit with the regularized fit. Not used if \texttt{relax=FALSE}. Requires \texttt{glmnet 3.0} or later.

\[ \text{relax} \]
For \texttt{cv.glmnet.formula}, whether to perform a relaxed fit after the regularised one. Requires \texttt{glmnet 3.0} or later.

\[ \text{object} \]
For the \texttt{predict} and \texttt{coef} methods, an object of class \texttt{cv.glmnet.formula}.

\[ \text{newdata} \]
For the \texttt{predict} method, a data frame containing the observations for which to calculate predictions.
Details

The `cv.glmnet` function in this package is an S3 generic with a formula and a default method. The former calls the latter, and the latter is simply a direct call to the `cv.glmnet` function in package `glmnet`. All the arguments to `glmnet::cv.glmnet` are (or should be) supported.

There are two ways in which the matrix of predictors can be generated. The default, with `use.model.frame = FALSE`, is to process the additive terms in the formula independently. With wide datasets, this is much faster and more memory-efficient than the standard R approach which uses the `model.frame` and `model.matrix` functions. However, the resulting model object is not exactly the same as if the standard approach had been used; in particular, it lacks a bona fide `terms` object. If you require interoperability with other packages that assume the standard model object structure, set `use.model.frame = TRUE`. See discussion for more information on this topic.

The `predict` and `coef` methods are wrappers for the corresponding methods in the `glmnet` package. The former constructs a predictor model matrix from its `newdata` argument and passes that as the `newx` argument to `glmnet:::predict.cv.glmnet`.

Value

For `cv.glmnet.formula`, an object of class either `cv.glmnet.formula` or `cv.relaxed.formula`, based on the value of the `relax` argument. This is basically the same object created by `glmnet::cv.glmnet`, but with extra components to allow formula usage.

See Also

`glmnet::cv.glmnet`, `glmnet::predict.cv.glmnet`, `glmnet::coef.cv.glmnet`, `model.frame`, `model.matrix`

Examples

```r
cv.glmnet(mpg ~ ., data=mtcars)

# Not run:
# Leukemia example dataset from Trevor Hastie's website
download.file("https://web.stanford.edu/~hastie/glmnet/glmnetData/Leukemia.RData",
    "Leukemia.RData")
load("Leukemia.RData")
leuk <- do.call(data.frame, Leukemia)
cv.glmnet(y ~ ., leuk, family="binomial")
```

## End(Not run)

---

`cva.glmnet` *Do elastic net cross-validation for alpha and lambda simultaneously*

Description

Do elastic net cross-validation for alpha and lambda simultaneously
Usage

cva.glmnet(x, ...)

## Default S3 method:
cva.glmnet(
  x,
  y,
  alpha = seq(0, 1, len = 11)^3,
  nfolds = 10,
  foldid = sample(rep(seq_len(nfolds), length = nrow(x))),
  ..., outerParallel = NULL,
  checkInnerParallel = TRUE
)

## S3 method for class 'formula'
cva.glmnet(
  formula,
  data,
  ..., weights = NULL,
  offset = NULL,
  subset = NULL,
  na.action = getOption("na.action"),
  drop.unused.levels = FALSE,
  xlev = NULL,
  sparse = FALSE,
  use.model.frame = FALSE
)

## S3 method for class 'cva.glmnet'
predict(
  object,
  newx,
  alpha,
  which = match(TRUE, abs(object$alpha - alpha) < 1e-08),
  ...,
)

## S3 method for class 'cva.glmnet.formula'
predict(
  object,
  newdata,
  alpha,
  which = match(TRUE, abs(object$alpha - alpha) < 1e-08),
  na.action = na.pass,
  ...,
)
## S3 method for class 'cva.glmnet'
coef(
   object,
   alpha,
   which = match(TRUE, abs(object$alpha - alpha) < 1e-08),
   ...
)

## S3 method for class 'cva.glmnet.formula'
print(x, ...)

## S3 method for class 'cva.glmnet'
plot(x, ..., legend.x = xlim[1], legend.y = xlim[2], log.x = TRUE)

minlossplot(x, ...)

## S3 method for class 'cva.glmnet'
minlossplot(x, ..., cv.type = c("1se", "min"))

### Arguments

- **x**
  - A matrix of predictor variables; or for the plotting methods, an object returned by `cva.glmnet`.
- **...**
  - Further arguments to be passed to lower-level functions. In the case of `cva.glmnet`, these arguments are passed to `cv.glmnet`; for `predict` and `coef`, they are passed to `predict.cv.glmnet`; and for `plot` and `minlossplot`, to `plot`.
- **y**
  - A response vector or matrix (for a multinomial response).
- **alpha**
  - A vector of alpha values for which to do cross-validation. The default is a sequence of 11 values more closely spaced around alpha = 0. For the `predict` and `coef` methods, the specific value of alpha for which to return predictions/regression coefficients.
- **nfolds**
  - The number of cross-validation folds to use. Defaults to 10.
- **foldid**
  - Vector of fold IDs for cross-validation. See `glmnet::cv.glmnet`.
- **outerParallel**
  - Method of parallelising the outer loop over alpha. See 'Details' below. If `NULL`, the loop is run sequentially.
- **checkInnerParallel**
  - If the outer loop is run in parallel, check that the inner loop over lambda will not be in contention for cores.
- **formula**
  - A model formula; interaction terms are allowed and will be expanded per the usual rules for linear models.
- **data**
  - A data frame or matrix containing the variables in the formula.
- **weights**
  - An optional vector of case weights to be used in the fitting process. If missing, defaults to an unweighted fit.
- **offset**
  - An optional vector of offsets, an *a priori* known component to be included in the linear predictor.
subset
An optional vector specifying the subset of observations to be used to fit the model.

na.action
A function which indicates what should happen when the data contains missing values. For the predict method, na.action = na.pass will predict missing values with NA; na.omit or na.exclude will drop them.

drop.unused.levels
Should factors have unused levels dropped? Defaults to FALSE.

xlev
A named list of character vectors giving the full set of levels to be assumed for each factor.

sparse
Should the model matrix be in sparse format? This can save memory when dealing with many factor variables, each with many levels.

use.model.frame
Should the base model.frame function be used when constructing the model matrix? This is the standard method that most R modelling functions use, but has some disadvantages. The default is to avoid model.frame and construct the model matrix term-by-term; see discussion.

object
For the predict and coef methods, an object returned by cva.glmnet.

newx
For the predict method, a matrix of predictor variables.

which
An alternative way of specifying alpha; the index number of the desired value within the alpha vector. If both which and alpha are supplied, the former takes precedence.

newdata
For the predict and coef methods, a data frame containing the observations for which to calculate predictions.

legend.x, legend.y
Location for the legend. Defaults to the top-left corner of the plot. Set either of these to NULL to omit the legend.

log.x
Whether to plot the X-axis (lambda) on the log scale. Defaults to TRUE, which for most lambda sequences produces a more reasonable looking plot. If your lambda sequence includes zero, set this to FALSE.

cv.type
For minlossplot, which cross-validated loss value to plot for each value of alpha. This can be either "min" which is the minimum loss, or "1se" which is the highest loss within 1 standard error of the minimum. The default is "1se".

Details
The cva.glmnet function does simultaneous cross-validation for both the alpha and lambda parameters in an elastic net model. The procedure is as outlined in the documentation for glmnet::cv.glmnet: it creates a vector foldid allocating the observations into folds, and then calls cv.glmnet in a loop over different values of alpha, but the same values of foldid each time.

Optionally this loop over alpha can be parallelised; currently, cva.glmnet knows about two methods of doing so:

- Via parLapply in the parallel package. To use this, set outerParallel to a valid cluster object created by makeCluster.
Via \textit{rxExec} as supplied by Microsoft R Server’s RevoScaleR package. To use this, set \texttt{outerParallel} to a valid compute context created by \texttt{RxComputeContext}, or a character string specifying such a context.

If the outer loop is run in parallel, \texttt{cva.glmnet} can check if the inner loop (over lambda) is also set to run in parallel, and disable this if it would lead to contention for cores. This is done if it is likely that the parallelisation is local on a multicore machine, i.e. if \texttt{outerParallel} is a \texttt{SOCKcluster} object running on “localhost”, or if the RevoScaleR compute context is local parallel.

There are two ways in which the matrix of predictors can be generated. The default, with \texttt{use.model.frame = FALSE}, is to process the additive terms in the formula independently. With wide datasets, this is much faster and more memory-efficient than the standard R approach which uses the \texttt{model.frame} and \texttt{model.matrix} functions. However, the resulting model object is not exactly the same as if the standard approach had been used; in particular, it lacks a bona fide \texttt{terms} object. If you require interoperability with other packages that assume the standard model object structure, set \texttt{use.model.frame = TRUE}. See discussion for more information on this topic.

The \texttt{predict} method computes predictions for a specific alpha value given a \texttt{cva.glmnet} object. It looks up the supplied alpha (possibly supplied indirectly via the \texttt{which} argument) in the object’s stored alpha vector, and calls \texttt{glmnet:::predict.cv.glmnet} on the corresponding \texttt{cv.glmnet} fit. All the arguments to that function are (or should be) supported.

The \texttt{coef} method is similar, returning the coefficients for the selected alpha value via \texttt{glmnet:::coef.cv.glmnet}.

The \texttt{plot} method for \texttt{cva.glmnet} objects plots the average cross-validated loss by lambda, for each value of alpha. Each line represents one \texttt{cv.glmnet} fit, corresponding to one value of alpha. Note that the specific lambda values can vary substantially by alpha.

The \texttt{minlossplot} function gives the best (lowest) cross-validated loss for each value of alpha.

\textbf{Value}

For \texttt{cva.glmnet.default}, an object of class \texttt{cva.glmnet}. This is a list containing the following:

- \texttt{alpha} The vector of alpha values
- \texttt{nfold} The number of folds
- \texttt{modlist} A list of \texttt{cv.glmnet} objects, containing the cross-validation results for each value of alpha

The function \texttt{cva.glmnet.formula} adds a few more components to the above, to facilitate working with formulas.

For the \texttt{predict} method, a vector or matrix of predicted values.

For the \texttt{coef} method, a vector of regularised regression coefficients.

\textbf{See Also}

- \texttt{glmnet::cv.glmnet}
- \texttt{glmnet::predict.cv.glmnet, glmnet::coef.cv.glmnet}
- \texttt{cva.glmnet, glmnet::cv.glmnet, plot}
**glmnet**

**Examples**

```r
cva <- cva.glmnet(mpg ~ ., data=mtcars)
predict(cva, mtcars, alpha=1)
```

```r
## Not run:
# Leukemia example dataset from Trevor Hastie's website
download.file("https://web.stanford.edu/~hastie/glmnet/glmnetData/Leukemia.RData", "Leukemia.RData")
load("Leukemia.Rdata")
leuk <- do.call(data.frame, Leukemia)
leuk.cva <- cva.glmnet(y ~ ., leuk, family="binomial")
leuk.pred <- predict(leuk.cva, leuk, which=6)
## End(Not run)
```

---

**glmnet**

Formula interface for elastic net modelling with glmnet

**Description**

Formula interface for elastic net modelling with glmnet

**Usage**

```r
glmnet(x, ...)
```

```r
## Default S3 method:
glmnet(x, y, ...)
```

```r
## S3 method for class 'formula'
glmnet(
  formula,
  data,
  family = c("gaussian", "binomial", "poisson", "multinomial", "cox", "mgaussian"),
  alpha = 1,
  ..., 
  weights = NULL,
  offset = NULL,
  subset = NULL,
  na.action =getOption("na.action"),
  drop.unused.levels = FALSE,
  xlev = NULL,
  sparse = FALSE,
  use.model.frame = FALSE,
  relax = FALSE
)
```
## S3 method for class 'glmnet.formula'
predict(object, newdata, offset = NULL, na.action = na.pass, ...)

## S3 method for class 'glmnet.formula'
coef(object, ...)

## S3 method for class 'glmnet.formula'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  print.deviance.ratios = FALSE,
  ...
)

## S3 method for class 'relaxed.formula'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  print.deviance.ratios = FALSE,
  ...
)

## S3 method for class 'relaxed.formula'
predict(object, newdata, offset = NULL, na.action = na.pass, ...)

## S3 method for class 'relaxed.formula'
coef(object, ...)

### Arguments

**x**
For the default method, a matrix of predictor variables.

**...**
For \texttt{glmnet.formula} and \texttt{glmnet.default}, other arguments to be passed to \texttt{glmnet::glmnet}; for the \texttt{predict} and \texttt{coef} methods, arguments to be passed to their counterparts in package \texttt{glmnet}.

**y**
For the default method, a response vector or matrix (for a multinomial response).

**formula**
A model formula; interaction terms are allowed and will be expanded per the usual rules for linear models.

**data**
A data frame or matrix containing the variables in the formula.

**family**
The model family. See \texttt{glmnet::glmnet} for how to specify this argument.

**alpha**
The elastic net mixing parameter. See \texttt{glmnet::glmnet} for more details.

**weights**
An optional vector of case weights to be used in the fitting process. If missing, defaults to an unweighted fit.

**offset**
An optional vector of offsets, an \textit{a priori} known component to be included in the linear predictor.

**subset**
An optional vector specifying the subset of observations to be used to fit the model.
na.action  A function which indicates what should happen when the data contains missing values. For the predict method, na.action = na.pass will predict missing values with NA; na.omit or na.exclude will drop them.

drop.unused.levels  Should factors have unused levels dropped? Defaults to FALSE.

xlev  A named list of character vectors giving the full set of levels to be assumed for each factor.

sparse  Should the model matrix be in sparse format? This can save memory when dealing with many factor variables, each with many levels.

use.model.frame  Should the base model.frame function be used when constructing the model matrix? This is the standard method that most R modelling functions use, but has some disadvantages. The default is to avoid model.frame and construct the model matrix term-by-term; see discussion.

relax  For glmnet.formula, whether to perform a relaxed (non-regularised) fit after the regularised one. Requires glmnet 3.0 or later.

object  For the predict and coef methods, an object of class glmnet.formula.

newdata  For the predict method, a data frame containing the observations for which to calculate predictions.

digits  Significant digits in printed output.

print.deviance.ratios  Whether to print the table of deviance ratios, as per glmnet::print.glmnet.

Details

The glmnet function in this package is an S3 generic with a formula and a default method. The former calls the latter, and the latter is simply a direct call to the glmnet function in package glmnet. All the arguments to glmnet::glmnet are (or should be) supported.

There are two ways in which the matrix of predictors can be generated. The default, with use.model.frame = FALSE, is to process the additive terms in the formula independently. With wide datasets, this is much faster and more memory-efficient than the standard R approach which uses the model.frame and model.matrix functions. However, the resulting model object is not exactly the same as if the standard approach had been used; in particular, it lacks a bona fide terms object. If you require interoperability with other packages that assume the standard model object structure, set use.model.frame = TRUE. See discussion for more information on this topic.

The predict and coef methods are wrappers for the corresponding methods in the glmnet package. The former constructs a predictor model matrix from its newdata argument and passes that as the newx argument to glmnet::predict.glmnet.

Value

For glmnet.formula, an object of class either glmnet.formula or relaxed.formula, based on the value of the relax argument. This is basically the same object created by glmnet::glmnet, but with extra components to allow formula usage.
See Also

`glmnet::glmnet, glmnet::predict.glmnet, glmnet::coef.glmnet, model.frame, model.matrix`

Examples

```r
glmnet(mpg ~ ., data=mtcars)

glmnet(Species ~ ., data=iris, family="multinomial")

## Not run:

# Leukemia example dataset from Trevor Hastie's website
download.file("https://web.stanford.edu/~hastie/glmnet/glmnetData/Leukemia.RData", "Leukemia.RData")
load("Leukemia.RData")
leuk <- do.call(data.frame, Leukemia)
glmnet(y ~ ., leuk, family="binomial")

## End(Not run)
```

glmnet.model.matrix

Model matrix options for glmnet

Description

This page describes the options available for generating the model matrix.

Details

There are two ways in which glmnetUtils can generate a model matrix out of a formula and data frame. The first is to use the standard R machinery comprising `model.frame` and `model.matrix`; and the second is to build the matrix one variable at a time. These options are discussed and contrasted below.

Using `model.frame`

This is the simpler option, and the one that is most compatible with other R modelling functions. The `model.frame` function takes a formula and data frame and returns a `model frame`: a data frame with special information attached that lets R make sense of the terms in the formula. For example, if a formula includes an interaction term, the model frame will specify which columns in the data relate to the interaction, and how they should be treated. Similarly, if the formula includes expressions like \( \exp(x) \) or \( I(x^2) \) on the RHS, `model.frame` will evaluate these expressions and include them in the output.

The major disadvantage of using `model.frame` is that it generates a `terms` object, which encodes how variables and interactions are organised. One of the attributes of this object is a matrix with one row per variable, and one column per main effect and interaction. At minimum, this is (approximately) a \( p \times p \) square matrix where \( p \) is the number of main effects in the model. For wide datasets with \( p > 10000 \), this matrix can approach or exceed a gigabyte in size. Even if there is
enough memory to store such an object, generating the model matrix can take a significant amount of time.

Another issue with the standard R approach is the treatment of factors. Normally, `model.matrix` will turn an $N$-level factor into an indicator matrix with $N - 1$ columns, with one column being dropped. This is necessary for unregularised models as fit with `lm` and `glm`, since the full set of $N$ columns is linearly dependent. With the usual treatment contrasts, the interpretation is that the dropped column represents a baseline level, while the coefficients for the other columns represent the difference in the response relative to the baseline.

This may not be appropriate for a regularised model as fit with `glmnet`. The regularisation procedure shrinks the coefficients towards zero, which forces the estimated differences from the baseline to be smaller. But this only makes sense if the baseline level was chosen beforehand, or is otherwise meaningful as a default; otherwise it is effectively making the levels more similar to an arbitrarily chosen level.

**Manually building the model matrix**

To deal with the problems above, `glmnetUtils` by default will avoid using `model.frame`, instead building up the model matrix term-by-term. This avoids the memory cost of creating a `terms` object, and can be noticeably faster than the standard approach. It will also include one column in the model matrix for all levels in a factor; that is, no baseline level is assumed. In this situation, the coefficients represent differences from the overall mean response, and shrinking them to zero is meaningful (usually).

This works in an additive fashion, ie the formula $\sim a + b \colon c + d \times e$ is treated as consisting of three terms, $a$, $b \colon c$ and $d \times e$ each of which is processed independently of the others. A dot in the formula includes all main effect terms, ie $\sim . + a \times b + f(x)$ expands to $\sim a + b + x + a \times b + f(x)$ (assuming $a$, $b$ and $x$ are the only columns in the data). Note that a formula like $\sim (a + b) + (c + d)$ will be treated as two terms, $a + b$ and $c + d$.

The code can handle fairly complex formulas, but it is not as sophisticated as base `model.frame` and `model.matrix`. In particular, terms that are to be omitted from the model must be at the end of the formula: $\sim . - c$ works, but not $\sim -c + .$.
Index

coeff.cv.glmnet.formula (cv.glmnet), 2
coeff.cv.relaxed.formula (cv.glmnet), 2
coeff.cva.glmnet (cva.glmnet), 4
coeff.glmnet.formula (glmnet), 9
coeff.relaxed.formula (glmnet), 9
cv.glmnet, 2
cva.glmnet, 4, 8
discussion, 3, 4, 7, 8, 11
glmnet, 9
glmnet.model.frame
  (glmnet.model.matrix), 12
glmnet.model.matrix, 12
glmnet.modelFrame
  (glmnet.model.matrix), 12
glmnet.modelMatrix
  (glmnet.model.matrix), 12
glmnet::coeff.cv.glmnet, 4, 8
glmnet::coeff.glmnet, 12
glmnet::cv.glmnet, 3, 4, 6–8
glmnet::glmnet, 3, 10, 12
glmnet::predict.cv.glmnet, 4, 8
glmnet::predict.glmnet, 12
glmnet::print.glmnet, 11
glmnetUtils, 13
glmnetUtils-package (glmnetUtils), 13
makeCluster, 7
minlossplot (cva.glmnet), 4
model.frame, 3, 4, 7, 11, 12
model.matrix, 4, 12
parLapply, 7
plot, 8
plot.cva.glmnet (cva.glmnet), 4
predict.cv.glmnet.formula (cv.glmnet), 2
predict.cv.relaxed.formula (cv.glmnet), 2
predict.cva.glmnet (cva.glmnet), 4
predict.glmnet.formula (glmnet), 9
predict.relaxed.formula (glmnet), 9
print.cv.relaxed.formula (glmnet), 2
print.cva.glmnet.formula (cva.glmnet), 4
print.glmnet.formula (glmnet), 9
print.relaxed.formula (glmnet), 9
terms, 4, 8, 11, 12
treatment contrasts, 13