Package 'picasso'

July 25, 2026

Type Package

Title Sparse Learning with Convex and Concave Penalties

Version 2.0.1

Date 2026-07-25

Maintainer Tuo Zhao <tourzhao@gatech.edu>

Depends R (>= 3.1.0), Matrix

Imports methods

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Description Fast regularization paths for sparse Gaussian, binomial, Poisson, square-root-lasso, and multinomial models with lasso, smoothly clipped absolute deviation, or minimax concave penalties. Computation uses pathwise coordinate optimization, active-set updates, warm starts, screening rules, Proximal Newton iterations, quadratic majorization, and adaptive local linear approximation where appropriate. Core solvers are implemented in C++, and coefficient paths are returned as Matrix-compatible objects.

License GPL-3

Encoding UTF-8

Copyright See inst/COPYRIGHTS for bundled third-party copyright and license notices.

Repository CRAN

NeedsCompilation yes

Author Jason Ge [aut],
Xingguo Li [aut],
Haoming Jiang [aut],
Mengdi Wang [aut],
Tong Zhang [aut],
Han Liu [aut],
Tuo Zhao [aut, cre],
Gael Guennebaud [ctb] (Contributor to bundled Eigen headers),

Benoit Jacob [ctb] (Contributor to bundled Eigen headers),
Eigen Library Authors [cph] (Copyright holders of bundled Eigen headers
in src/include/eigen3)

Date/Publication 2026-07-25 21:00:02 UTC

Contents

`picasso-package` *picasso: Sparse Learning with Convex and Non-Convex Penalties*

Description

Computationally efficient tools for fitting sparse generalized linear models with convex or non-convex penalties. Supported penalties include lasso, smoothly clipped absolute deviation (SCAD), and minimax concave penalty (MCP). Supported families are Gaussian, binomial, Poisson, square-root-lasso, and multinomial. Computation combines pathwise coordinate optimization with warm starts, active-set updates, screening rules, Proximal Newton iterations, and adaptive local linear approximation for applicable nonconvex models.

Details

Core optimization routines are implemented in C++. Gaussian models use active-set coordinate descent with an automatic residual/covariance backend; binomial and Poisson models use active-set Proximal Newton/IRLS, square-root-lasso uses a global quadratic majorizer with active-set coordinate updates, and multinomial uses class-coupled Proximal Newton/IRLS. The R interface returns coefficient paths as **Matrix** objects and provides prediction, assessment, confusion-matrix, and cross-validation helpers.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso.

---

assess.picasso *Model Assessment for picasso Fits*

Description

Compute prediction-quality metrics across the regularization path. `assess.picasso` evaluates deviance, MSE, MAE, or classification error on a test set; `confusion.picasso` returns confusion matrices for binary and multinomial classification. For scalar families, large lambda paths are evaluated in bounded link-predictor blocks; small paths still use one matrix multiplication.

Usage

```
assess.picasso(object, newx, newy, newoffset = NULL, ...)

## S3 method for class 'assess.picasso'
print(x, ...)

confusion.picasso(object, newx, newy, lambda.idx = NULL,
  newoffset = NULL, ...)
```

Arguments

`object`	A fitted `picasso` object with a `$family` field.
`newx`	Numeric design matrix for the test set, $n \times d$.
`newy`	Test-set response vector. Binomial and multinomial values must match class labels seen during fitting. Numeric zero/one remains accepted as an encoded binomial response; legacy fits without a stored class map require that encoding.
`lambda.idx`	Integer vector of lambda indices for which to compute confusion matrices. Defaults to all lambdas.

newoffset	Optional finite numeric vector with one offset per row of newx. It is required for binomial or Poisson assessment, and for binomial confusion matrices, when the model was fitted with an offset.
x	An "assess.picasso" object.
...	Currently unused.

Value

assess.picasso returns an object of class "assess.picasso" containing named numeric vectors (one value per lambda):

lambda Lambda values.

deviance Family-specific test loss: half mean squared error for Gaussian and square-root-lasso, Bernoulli or multinomial negative log-likelihood for categorical models, and conventional mean Poisson deviance for Poisson.

mse Mean squared error (Gaussian, square-root-lasso, or Poisson).

mae Mean absolute error (Gaussian or square-root-lasso).

class Misclassification rate (binomial or multinomial).

confusion.picasso returns a list of table objects, one per selected lambda value, with predicted classes in rows and observed classes in columns. Multinomial tables retain all fitted class levels on both axes, including levels absent from the supplied test subset; binomial tables always retain levels 0 and 1.

print.assess.picasso prints the range of each metric across the fitted path and returns its input invisibly.

Scalar-family assessment targets about 8 MiB for each link-predictor and coefficient block, subject to a minimum of one lambda column. Metric calculation may use additional arrays of the same bounded block shape. This changes only working-memory use, not the returned metrics or their ordering.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso, cv.picasso

Examples

```
set.seed(1)
n <- 100; d <- 30
X <- matrix(rnorm(n * d), n, d)
Y <- rbinom(n, 1, 0.5)
fit <- picasso(X, Y, family = "binomial", nlambda = 20)
a <- assess.picasso(fit, X, Y)
print(a)
```

coef.gaussian *Extract Coefficients from an Object with S3 Class "gaussian"*

Description

Extract estimated coefficient vectors from selected points on the regularization path.

Usage

```
## S3 method for class 'gaussian'
coef(object, lambda.idx = NULL, beta.idx = NULL, ...)
```

Arguments

`object`	An object with S3 class "gaussian".
`lambda.idx`	Indices of regularization parameters in the solution path to extract. By default, at most the first three fitted path points are used.
`beta.idx`	Indices of coefficient entries to extract. By default, at most the first three coefficients are used.
`...`	Arguments to be passed to methods.

Value

A numeric matrix of extracted coefficients. Columns correspond to `lambda.idx`. Rows contain the intercept (`"(Intercept)"`) and the selected coefficients (`"beta[i]"` for `i` in `beta.idx`).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

`picasso` and `picasso-package`.

coef.logit *Extract Coefficients from an Object with S3 Class "logit"*

Description

Extract estimated coefficient vectors from selected points on the regularization path.

Usage

```
## S3 method for class 'logit'
coef(object, lambda.idx = NULL, beta.idx = NULL, ...)
```

Arguments

`object`	An object with S3 class `"logit"`.
`lambda.idx`	Indices of regularization parameters in the solution path to extract. By default, at most the first three fitted path points are used.
`beta.idx`	Indices of coefficient entries to extract. By default, at most the first three coefficients are used.
`...`	Arguments to be passed to methods.

Value

A numeric matrix of extracted coefficients. Columns correspond to `lambda.idx`. Rows contain the intercept (`"(Intercept)"`) and the selected coefficients (`"beta[i]"` for `i` in `beta.idx`).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

`picasso` and `picasso-package`.

`coef.poisson` *Extract Coefficients from an Object with S3 Class* `"poisson"`

Description

Extract estimated coefficient vectors from selected points on the regularization path.

Usage

```
## S3 method for class 'poisson'
coef(object, lambda.idx = NULL, beta.idx = NULL, ...)
```

Arguments

`object`	An object with S3 class `"poisson"`.
`lambda.idx`	Indices of regularization parameters in the solution path to extract. By default, at most the first three fitted path points are used.
`beta.idx`	Indices of coefficient entries to extract. By default, at most the first three coefficients are used.
`...`	Arguments to be passed to methods.

Value

A numeric matrix of extracted coefficients. Columns correspond to `lambda.idx`. Rows contain the intercept (`"(Intercept)"`) and the selected coefficients (`"beta[i]"` for `i` in `beta.idx`).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

---

coef.sqrtlasso *Extract Coefficients from an Object with S3 Class "sqrtlasso"*

Description

Extract estimated coefficient vectors from selected points on the regularization path.

Usage

```
## S3 method for class 'sqrtlasso'
coef(object, lambda.idx = NULL, beta.idx = NULL, ...)
```

Arguments

object	An object with S3 class "sqrtlasso".
lambda.idx	Indices of regularization parameters in the solution path to extract. By default, at most the first three fitted path points are used.
beta.idx	Indices of coefficient entries to extract. By default, at most the first three coefficients are used.
...	Arguments to be passed to methods.

Value

A numeric matrix of extracted coefficients. Columns correspond to lambda.idx. Rows contain the intercept ("(Intercept)") and the selected coefficients ("beta[i]" for i in beta.idx).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

`cv.picasso` *Cross-Validation for picasso*

Description

Performs K-fold cross-validation to select the regularization parameter for `picasso` models. Multinomial models use a fixed full-data class map and softmax negative log-likelihood or misclassification loss.

Usage

```
cv.picasso(X, Y, ..., nfolds = 10, foldid = NULL,
  type.measure = "default", fast.mode = FALSE)

## S3 method for class 'cv.picasso'
print(x, ...)

## S3 method for class 'cv.picasso'
plot(x, sign.lambda = 1, ...)

## S3 method for class 'cv.picasso'
coef(object, s = c("lambda.min", "lambda.1se"), ...)

## S3 method for class 'cv.picasso'
predict(object, newdata, s = c("lambda.min", "lambda.1se"),
  type = "response", ...)
```

Arguments

`X`	Numeric design matrix, $n \times d$.
`Y`	Response vector. A one-column Gaussian response matrix is accepted for compatibility.
`...`	Additional arguments passed to `picasso`.
`nfolds`	Number of cross-validation folds. Default is `10`.
`foldid`	Optional integer vector of fold assignments (length n). Labels must be consecutive positive integers starting at one. If `NULL`, assignments are generated randomly; binomial and multinomial assignments are stratified by class. (The Python `cross_validate` counterpart instead expects zero-based labels, matching its zero-based lambda indices.)
`type.measure`	Loss measure for cross-validation. `"default"` uses `"class"` for binomial and multinomial models and `"deviance"` otherwise. Multinomial models support only `"class"` and `"deviance"`. Additional options: `"mse"`, `"mae"`, `"deviance"`, `"class"`. MSE and MAE are evaluated on the response scale: fitted probabilities for binomial models and fitted means for Poisson models.

fast.mode	Logical precision preset passed consistently to the full-data and fold fits. With the default FALSE, prec defaults to 1e-7 but may be supplied through ...; TRUE uses the benchmark-calibrated fast tolerance 4e-4 for Poisson and 1e-4 for other non-Gaussian families. Gaussian retains its glmnet-aligned 1e-7 objective-change tolerance.
x	A "cv.picasso" object.
object	A "cv.picasso" object.
s	Which lambda to use: "lambda.min" (default) selects the lambda that minimises CV error; "lambda.1se" uses the largest lambda within one standard error of the minimum.
sign.lambda	Multiplier for the x-axis of the CV plot (1 for $\log(\lambda)$, -1 for $-\log(\lambda)$). Zero lambda values are displayed at a finite position below the positive path.
newdata	Matrix of new observations for prediction.
type	Prediction type; passed to the underlying predict method.

Details

Every binomial or multinomial training fold must contain every class observed in the full data; automatic stratification therefore requires at least two observations per class. A custom foldid that violates this rule or a failed, truncated, or otherwise unusable fold fit produces an explicit error rather than silently omitting the fold. Normal deviance-tail stopping may shorten the initial full-data generated path; that retained path is then passed explicitly to every fold and must be covered in full. An unusually deep scalar path can make a training fold reach normal saturation before completing that fixed sequence. This is reported as a truncated-fold error; request fewer lambdas or a larger lambda.min.ratio when it occurs.

Binomial factor responses use the full-data two-level map in every fold and are scored as zero/one outcomes. When predicting from an offset-fitted binomial or Poisson cv.picasso object, pass newoffset through ... to the underlying prediction method.

For Gaussian models, the full-data type.gaussian = "auto" decision is also reused in every fold, so changing training-fold sample sizes cannot switch the native backend.

Value

cv.picasso returns an object of class "cv.picasso" with components:

lambda Ordered lambda sequence evaluated by every fold. A generated full-data fit may first stop normally and thereby establish a shorter path; every fold must then cover that fixed sequence completely.

cvm Mean cross-validated error for each lambda.

cvsd Standard error of the mean cross-validated error.

cvup, cvlo Upper and lower error-bar bounds.

nzero Number of nonzero coefficients at each lambda. For multinomial models this is the total across all classes.

lambda.min Lambda that minimises cvm.

lambda.1se Largest lambda within one standard error of the minimum.

name Name of the loss measure used.

foldid Fold assignment used for each observation.

family Model family.

fast.mode Logical precision preset used for all fits.

prec Effective convergence tolerance used for all fits.

picasso.fit Full-data `picasso` fit.

`print.cv.picasso` returns its input invisibly, and `plot.cv.picasso` draws the CV curve and returns `NULL` invisibly. `coef.cv.picasso` and `predict.cv.picasso` return the corresponding underlying fit-method result at `lambda.min` or `lambda.1se`.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

`picasso`

Examples

```
set.seed(1)
n <- 100; d <- 50
X <- matrix(rnorm(n * d), n, d)
Y <- X[, 1:5] %*% rnorm(5) + rnorm(n)
cv <- cv.picasso(X, Y, family = "gaussian", nlambda = 20)
print(cv$lambda.min)
```

---

eyedata *Bardet-Biedl Syndrome Gene Expression Data from Scheetz et al. (2006)*

Description

Gene expression data (200 genes, 120 samples) from microarray experiments on mammalian eye tissue reported by Scheetz et al. (2006).

Usage

```
data(eyedata)
```

Format

A list with components:

x A 120 by 200 matrix of gene probes.

y A numeric response vector of length 120 representing the expression level of the TRIM32 gene.

Details

This data set contains 120 samples and 200 predictors.

Author(s)

Xingguo Li, Tuo Zhao, Tong Zhang and Han Liu
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

References

1. T. Scheetz, K. Kim, R. Swiderski, A. Philp, T. Braun, K. Knudtson, A. Dorrance, G. DiBona, J. Huang, T. Casavant, V. Sheffield, and E. Stone. Regulation of gene expression in the mammalian eye and its relevance to eye disease. *Proceedings of the National Academy of Sciences of the United States of America*, 2006.

See Also

picasso-package.

Examples

```
data(eyedata)
image(eyedata$x)
```

`multinomial-methods` *Methods for Multinomial picasso Fits*

Description

Fit multinomial models with picasso using `family = "multinomial"`, then print, plot, extract coefficients, or predict from the fitted class-coupled regularization path.

Usage

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

## S3 method for class 'multinomial'
plot(x, which.class = 1, ...)

## S3 method for class 'multinomial'
coef(object, lambda.idx = NULL, beta.idx = NULL, ...)

## S3 method for class 'multinomial'
predict(object, newdata, lambda.idx = NULL,
  type = "response", s = NULL, ...)
```

Arguments

x	A fitted object with S3 class "multinomial".
object	A fitted object with S3 class "multinomial".
which.class	One-based class index whose coefficient path is plotted. The default is 1.
lambda.idx	One-based lambda indices for extraction or prediction. If NULL, the first three available lambdas (or the entire path when shorter) are used. Supply only one of lambda.idx and s.
beta.idx	One-based feature indices to extract. If NULL, the first three available features are used.
newdata	Finite numeric matrix of new observations with the same number of columns as the fitted design.
type	Prediction type. "response" returns softmax probabilities, "link" returns class-specific linear predictors, "class" returns the fitted label with the largest finite link-scale score (first fitted class on an exact tie), and "nonzero" returns nonzero feature indices separately by class.
s	Optional finite non-negative lambda *values*. Coefficients are linearly interpolated between fitted lambdas for response, link, and class predictions; values outside the path are clamped to an endpoint. type = "nonzero" uses the nearest fitted lambda instead.
...	Additional arguments (currently unused).

Details

The public fitting call is picasso(X, Y, family = "multinomial", ...). Responses must contain at least three observed classes; numeric, character, and factor labels are accepted, and unused factor levels are dropped.

The solver first identifies a strong working set and then solves the class-coupled restricted problem with Proximal Newton/IRLS iterations until the weighted-L1 subproblem converges. Full KKT checks update the active set and certify every retained subproblem. Pathwise warm starts, sequential strong screening, adaptive inexact Newton tolerances, vectorized coordinate kernels, and accepted-probability reuse reduce repeated work.

MCP and SCAD use adaptive local linear approximation. Every fit performs the minimum three total stages (one L1 master and two weighted-L1 updates). Raising lla.max.stages permits further stages, which stop once target stationarity is at most prec; exhausting the budget yields the usable status lla_stationarity_limit.

After at least five lambdas, an automatically generated path stops normally when explained deviance exceeds 0.999 or its improvement is below 1e-5. Explicit lambda sequences disable only this saturation rule; dfmax or a hard failure can still truncate them. A hard failure after one or more completed lambdas returns the committed prefix with a warning and failure metadata; failure before the first completed lambda is an error.

With no intercept, standardized predictors are scaled about the origin rather than centered. The zero model then uses uniform class probabilities for the automatic lambda path and null deviance.

Value

`print.multinomial` returns `x` invisibly. `plot.multinomial` draws one class's coefficient paths and returns `NULL` invisibly.

`coef.multinomial` returns a list of K numeric matrices, one per class. Rows contain the intercept followed by selected coefficients, and columns correspond to selected lambdas.

For one selected lambda, `predict.multinomial` returns an $n_{\text{new}} \times K$ matrix for `"response"` or `"link"`, a factor of length n_{new} for `"class"`, or a list of K integer vectors for `"nonzero"`. Multiple `lambda.idx` or `s` values return a list of those objects.

The fitted object returned by `picasso` contains class-specific `beta` and `intercept` lists, `lambda`, `df`, `levels`, `K`, `dev.ratio`, status/failure metadata, per-lambda diagnostics, `path.early.stopped`, and `requested.nlambda`; see `picasso`.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

`picasso`, `cv.picasso`, and `assess.picasso`.

Examples

```
set.seed(1)
n <- 120; d <- 12
X <- matrix(rnorm(n * d), n, d)
score <- cbind(X[, 1], -X[, 1] + X[, 2], -X[, 2])
Y <- factor(c("red", "green", "blue")[max.col(score)])
fit <- picasso(X, Y, family = "multinomial", nlambda = 12)
prob <- predict(fit, X[1:5, ], lambda.idx = fit$nlambda,
                type = "response")
pred <- predict(fit, X[1:5, ], lambda.idx = fit$nlambda,
                type = "class")
```

`picasso` *Fit Sparse Regression Models with Convex and Non-Convex Penalties*

Description

`picasso` fits regularization paths for sparse Gaussian, logistic (`family = "binomial"`), Poisson, sqrt-lasso, and multinomial models using lasso, MCP, or SCAD penalties.

Usage

```
picasso(X, Y, lambda = NULL, nlambda = 100, lambda.min.ratio =
                0.05, family = "gaussian", method = "l1",
                type.gaussian = "auto", gamma = 3, df = NULL,
                dfmax = NULL, standardize = TRUE, intercept = TRUE,
                prec = 1e-07, max.ite = 1000, verbose = FALSE,
                offset = NULL, lla.max.stages = 3L,
                fast.mode = FALSE)
```

Arguments

X	Numeric design matrix with n rows (samples) and d columns (features).
Y	Response vector of length n. Use finite numeric responses for `family = "gaussian"` and `family = "sqrtlasso"`, exactly two levels for `family = "binomial"`, non-negative integer counts with at least one positive count for `family = "poisson"`, and at least three observed class labels for `family = "multinomial"`. Binomial and multinomial labels may be numeric, character, or factor values. Multinomial fits drop unused factor levels; binomial factor inputs must already contain exactly two levels.
lambda	Optional finite, non-negative, strictly decreasing sequence of regularization values. If `NULL`, the sequence is generated automatically from `nlambda` and `lambda.min.ratio`.
nlambda	Positive integer number of regularization values to generate when `lambda = NULL`. The default is `100`.
lambda.min.ratio	Smallest generated `lambda` value as a fraction of $\lambda_{\max}$ when `lambda = NULL`. It must lie strictly between zero and one; the default is `0.05`. For non-convex penalties in logistic and Poisson models, values much smaller than `0.05` may lead to ill-conditioning.
family	Model family. Supported values are `"gaussian"`, `"binomial"`, `"poisson"`, `"sqrtlasso"`, and `"multinomial"`. The default is `"gaussian"`.
method	Penalty type. Supported values are `"l1"` (lasso), `"mcp"`, and `"scad"`. The default is `"l1"`.
type.gaussian	Gaussian solver mode. `"naive"` maintains residuals and performs O(n) work per coordinate. `"covariance"` lazily caches Gram-matrix columns and then updates gradients in O(d) per coordinate. The default, `"auto"`, uses covariance updates only when an at-least-eight-point path can amortize the cache and the problem is small, path-sparse, or sufficiently tall; wide designs and designs above 1024 features use naive updates. Within those guards it selects covariance for $d <= 160$, $\lambda_{min}/\lambda_{max} >= 0.1$, a path ratio of at least 0.05 with $n >= 4d$, or any problem with $n >= 16d$. The 1024-feature guard caps a fully populated numeric cache at 8 MiB. Explicit `"naive"` and `"covariance"` requests bypass automatic selection.
gamma	Concavity parameter for MCP and SCAD penalties. MCP requires `gamma > 1` and SCAD requires `gamma > 2`. The default is `3`.
df	Reserved for backward compatibility with older Gaussian covariance-update implementations. It is currently not used by the active solver backend.

`dfmax` Nonzero-count threshold for early stopping on the lambda path. The crossing model is committed before stopping, so this is not a hard upper bound. If `NULL` (default), no count limit is imposed. Scalar-family solvers check the threshold after the minimum five-point path prefix; multinomial checks it after each committed lambda and counts nonzero class-feature entries. Gaussian, binomial, Poisson, and sqrt-lasso paths also apply their family-specific deviance saturation checks after at least 5 lambda values. Automatically generated multinomial paths stop when explained deviance exceeds 0.999 or its gain from the preceding lambda is below 1e-5. Explicit multinomial lambda sequences disable only that saturation rule; they may still return a prefix after `dfmax` stopping or a hard failure.

`standardize` If `TRUE`, scale columns of `X` before fitting. Columns are centered only when `intercept = TRUE`; a no-intercept model preserves the original zero point. The default is `TRUE`.

`intercept` If `TRUE`, include an intercept term. The default is `TRUE`.

`prec` Convergence tolerance used when `fast.mode = FALSE`. The default is `1e-7`. A custom positive value may be supplied in high-precision mode.

`max.ite` Maximum number of native solver iterations. For Gaussian models it bounds coordinate-descent sweeps at each lambda. For binomial, Poisson, sqrt-lasso, and multinomial models, it bounds active-set subproblem work within each weighted-L1 problem and is distinct from `lla.max.stages`. The default is `1000`.

`lla.max.stages` Maximum total number of local-linear-approximation (LLA) stages for MCP and SCAD binomial, Poisson, sqrt-lasso, and multinomial fits. The count includes the initial L1 master, so the default `3L` means one L1 master plus two weighted-L1 updates. Three stages are generally enough to obtain a useful model. Set a higher value when stricter stationarity is needed; after the minimum three stages, adaptive LLA stops once `prec` is met or the requested maximum is exhausted. The input is still validated for L1 fits but does not change their optimization. It is ignored by the direct Gaussian MCP/SCAD coordinate-descent solver.

`verbose` If `TRUE`, print tracing information during fitting. The default is `FALSE`.

`offset` Optional numeric vector of length n added to the linear predictor before the link function. Supported for `family = "binomial"` and `family = "poisson"`. Pass `NULL` (default) for no offset. Response/link predictions, assessment, and binomial class prediction from an offset-fitted model require a corresponding `newoffset` vector. Support extraction with `type = "nonzero"` does not.

`fast.mode` Logical precision preset. The default `FALSE` preserves PICASSO's high-precision tolerance (`prec = 1e-7`). Set to `TRUE` to use benchmark-calibrated glmnet-like stopping/KKT accuracy: `4e-4` for Poisson and `1e-4` for binomial, sqrt-lasso, and multinomial models. Gaussian retains `1e-7` because its scaled objective-change test already follows glmnet's convention; loosening it to `1e-4` was not accuracy-equivalent. Because stopping criteria differ by solver, these are achieved-accuracy presets, not a literal copy of glmnet's `thresh`. Supply custom `prec` values only when this option is `FALSE`.

Details

When `lambda` is not supplied, `picasso` constructs a logarithmically spaced path between $\lambda_{\max}$ and $\lambda_{\min} = \text{lambda.min.ratio} \cdot \lambda_{\max}$.

The method solves a penalized optimization problem:

$$\min_{\beta_0,\beta} \ \mathcal{L}(\beta_0, \beta) + \sum_{j=1}^{d} p_{\lambda,\gamma}(|\beta_j|),$$

where $p_{\lambda,\gamma}$ is lasso, MCP, or SCAD. For multinomial models, the penalty term is instead $\sum_{k=1}^{K} \sum_{j=1}^{d} p_{\lambda,\gamma}(|B_{jk}|)$; intercepts are never penalized.

Loss functions by family are:

- `"gaussian"`: $\mathcal{L}(\beta_0, \beta) = \frac{1}{2n}\|Y - \beta_0\mathbf{1} - X\beta\|_2^2$.
- `"binomial"`: $\mathcal{L}(\beta_0, \beta) = \frac{1}{n}\sum_{i=1}^{n}[\log(1 + \exp(\eta_i)) - y_i\eta_i]$.
- `"poisson"`: $\mathcal{L}(\beta_0, \beta) = \frac{1}{n}\sum_{i=1}^{n}[\exp(\eta_i) - y_i\eta_i]$.
- `"sqrtlasso"`: $\mathcal{L}(\beta_0, \beta) = \frac{1}{\sqrt{n}}\|Y - \beta_0\mathbf{1} - X\beta\|_2$.
- `"multinomial"`: $\mathcal{L}(b, B) = -\frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{\exp(b_{y_i} + x_i^\top B_{y_i})}{\sum_{k=1}^{K}\exp(b_k + x_i^\top B_k)}\right)$.

For binomial and Poisson models, $\eta_i = \beta_0 + x_i^\top\beta + o_i$, where o_i is the supplied offset or zero.

Gaussian paths use active-set coordinate descent with residual or lazy covariance updates. Binomial and Poisson weighted-L1 subproblems use active-set Proximal Newton/IRLS, square-root-lasso uses a global quadratic majorizer with active-set coordinate updates, and multinomial uses class-coupled active-set Proximal Newton/IRLS. MCP and SCAD use adaptive LLA around those weighted-L1 solvers except for the direct Gaussian nonconvex coordinate-descent implementation.

Value

A fitted object of class `"gaussian"`, `"logit"`, `"poisson"`, `"sqrtlasso"`, or `"multinomial"` depending on `family`.

Common components include:

beta A **Matrix** object of dimension $d \times \text{nlambda}$ containing fitted coefficients. Its concrete representation may be sparse or dense. Multinomial fits instead return a list of K such objects, one per class.

intercept Numeric vector of intercepts. Multinomial fits return a list of K numeric vectors, one per class.

lambda Regularization sequence used for fitting.

nlambda Number of regularization values.

df Degrees of freedom (number of nonzero coefficients) along the path.

nulldev Null-model loss or deviance under the fitted intercept convention and family-specific scale.

dev.ratio Fraction of null deviance explained at each lambda, analogous to R^2 for Gaussian fits. Values are clipped to zero through one.

method Penalty type used for fitting.

alg Identifier for the resolved native algorithm.

ite Iteration information returned by the solver.

verbose Input `verbose` value.

runtime Elapsed fitting time.

family Input `family` value.

fast.mode Logical precision preset used for fitting.

prec Effective convergence tolerance used by the native solver.

type.gaussian.requested For Gaussian fits, the requested solver mode.

type.gaussian For Gaussian fits, the resolved native solver mode.

status Native termination label. Gaussian iteration-limit failures retain only the previously converged lambda prefix.

status.code Integer form of `status`.

failure Details for the first uncommitted hard failure, otherwise `NULL`.

Binomial, Poisson, sqrt-lasso, and multinomial fits additionally include:

runt Native per-lambda runtime.

lla.max.stages Requested adaptive-LLA stage budget.

diagnostics Per-lambda runtime, penalized objective, weighted-L1 KKT residual, and target stationarity. Scalar-family diagnostics also report solver iterations, active-set size, and completed LLA stages; multinomial diagnostics instead report outer iterations, inner sweeps, coordinate updates, and active-set size.

For non-Gaussian MCP/SCAD fits, status `lla_stationarity_limit` means that every returned model is usable, but at least one point reached its LLA-stage budget before meeting `prec`.

Native path fitting polls for a pending user interrupt (`Ctrl-C`) at every lambda boundary, so a long path can be stopped without waiting for completion; the interrupt is then raised as a standard R `interrupt` condition. A single very expensive lambda still runs to completion before the poll.

Multinomial fits also include `path.early.stopped` and `requested.nlambda`. A successful automatically generated path may contain fewer lambdas than requested after its deviance has saturated. Explicit paths do not apply saturation stopping, but may still be truncated by `dfmax` or a hard failure. Every retained weighted-L1 subproblem meets its KKT tolerance; for MCP/SCAD, inspect target stationarity when `status.code == 10`.

Binomial and Poisson fits include `offset.used`, a logical flag recording whether `offset` was explicitly supplied. Binomial fits include the two response `levels` used for zero/one encoding; multinomial fits include `K` and the retained class `levels`.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

References

1. J. Friedman, T. Hastie, H. Hofling, and R. Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 2007.
2. C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 2010.
3. J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 2001.
4. R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B*, 2012.
5. J. Ge, X. Li, H. Jiang, H. Liu, T. Zhang, M. Wang, and T. Zhao. PICASSO: A sparse learning library for high-dimensional data analysis in R and Python. *Journal of Machine Learning Research*, 20(44):1–5, 2019. https://www.jmlr.org/papers/v20/17-722.html.

See Also

picasso-package.

Examples

```
#####################################################################
## Sparse linear regression
## Generate the design matrix and regression coefficient vector
n = 100 # sample number
d = 80 # sample dimension
c = 0.5 # correlation parameter
s = 20  # support size of coefficient
set.seed(2016)
X = scale(matrix(rnorm(n*d),n,d)+c*rnorm(n))/sqrt(n-1)*sqrt(n)
beta = c(runif(s), rep(0, d-s))

## Generate response using Gaussian noise, and fit sparse linear models
noise = rnorm(n)
Y = X%*%beta + noise

## automatic Gaussian backend selection (the default)
fitted.l1.auto = picasso(X, Y, nlambda=100)

## l1 regularization solved with naive update
fitted.l1.naive = picasso(X, Y, nlambda=100, type.gaussian="naive")

## covariance updates can be faster when n >> d
fitted.l1.covariance  = picasso(X, Y, nlambda=100, type.gaussian="covariance")

## early stopping: stop when more than 10 nonzero coefficients
fitted.l1.dfmax = picasso(X, Y, nlambda=100, dfmax=10)

## mcp regularization
fitted.mcp = picasso(X, Y, nlambda=100, method="mcp")

## scad regularization
```

```
fitted.scad = picasso(X, Y, nlambda=100, method="scad")

## lambdas used
print(fitted.l1.naive$lambda)

## number of nonzero coefficients for each lambda
print(fitted.l1.naive$df)

## coefficients and intercept for the i-th lambda
i = 30
print(fitted.l1.naive$lambda[i])
print(fitted.l1.naive$beta[,i])
print(fitted.l1.naive$intercept[i])

## Visualize the solution path
plot(fitted.l1.naive)
plot(fitted.l1.covariance)
plot(fitted.mcp)
plot(fitted.scad)


#################################################################
## Sparse logistic regression
## Generate the design matrix and regression coefficient vector
n <- 100  # sample number
d <- 80   # sample dimension
c <- 0.5   # parameter controlling the correlation between columns of X
s <- 20    # support size of coefficient
set.seed(2016)
X <- scale(matrix(rnorm(n*d),n,d)+c*rnorm(n))/sqrt(n-1)*sqrt(n)
beta <- c(runif(s), rep(0, d-s))

## Generate response and fit sparse logistic models
p = 1/(1+exp(-X%*%beta))
Y = rbinom(n, rep(1,n), p)

## l1 regularization
fitted.l1 = picasso(X, Y, nlambda=100, family="binomial", method="l1")

## mcp regularization
fitted.mcp = picasso(X, Y, nlambda=100, family="binomial", method="mcp")

## scad regularization
fitted.scad = picasso(X, Y, nlambda=100, family="binomial", method="scad")

## lambdas used
print(fitted.l1$lambda)

## number of nonzero coefficients for each lambda
print(fitted.l1$df)

## coefficients and intercept for the i-th lambda
i = 30
```

```
print(fitted.l1$lambda[i])
print(fitted.l1$beta[,i])
print(fitted.l1$intercept[i])

## Visualize the solution path
plot(fitted.l1)

## Predicted probabilities for the first 5 observations at lambda[30]
param.l1 = predict(fitted.l1, X[1:5, ], lambda.idx = 30,
                   type = "response")


#################################################################
## Sparse poisson regression
## Generate the design matrix and regression coefficient vector
n <- 100  # sample number
d <- 80   # sample dimension
c <- 0.5   # parameter controlling the correlation between columns of X
s <- 20    # support size of coefficient
set.seed(2016)
X <- scale(matrix(rnorm(n*d),n,d)+c*rnorm(n))/sqrt(n-1)*sqrt(n)
beta <- c(runif(s), rep(0, d-s))/sqrt(s)

## Generate response and fit sparse poisson models
p = X%*%beta+rnorm(n)
Y = rpois(n, exp(p))

## l1 regularization
fitted.l1 = picasso(X, Y, nlambda=100, family="poisson", method="l1")

## mcp regularization
fitted.mcp = picasso(X, Y, nlambda=100, family="poisson", method="mcp")

## scad regularization
fitted.scad = picasso(X, Y, nlambda=100, family="poisson", method="scad")

## lambdas used
print(fitted.l1$lambda)

## number of nonzero coefficients for each lambda
print(fitted.l1$df)

## coefficients and intercept for the i-th lambda
i = 30
print(fitted.l1$lambda[i])
print(fitted.l1$beta[,i])
print(fitted.l1$intercept[i])

## Visualize the solution path
plot(fitted.l1)


#################################################################
```

```
## Square-root-lasso
set.seed(2)
X <- matrix(rnorm(80 * 12), 80, 12)
Y <- X[, 1] - X[, 2] + rnorm(80)
fitted.sqrt <- picasso(
  X, Y, family = "sqrtlasso", nlambda = 12, method = "l1"
)
predict(fitted.sqrt, X[1:3, ], lambda.idx = fitted.sqrt$nlambda)


################################################################
## Multinomial regression with retained class labels
score <- cbind(X[, 1], -X[, 1] + X[, 2], -X[, 2])
Y <- factor(c("red", "green", "blue")[max.col(score)])
fitted.multi <- picasso(
  X, Y, family = "multinomial", nlambda = 12, method = "l1"
)
predict(fitted.multi, X[1:3, ], lambda.idx = fitted.multi$nlambda,
        type = "class")
```

`plot.gaussian` *Plot Method for an Object with S3 Class* `"gaussian"`

Description

Plot coefficient paths across regularization parameters.

Usage

```
## S3 method for class 'gaussian'
plot(x, ...)
```

Arguments

`x`	An object with S3 class `"gaussian"`.
`...`	Arguments to be passed to methods.

Value

No return value, called for side effects (drawing the coefficient-path plot).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

`picasso` and `picasso-package`.

plot.logit *Plot Method for an Object with S3 Class "logit"*

Description

Plot coefficient paths across regularization parameters.

Usage

```
## S3 method for class 'logit'
plot(x, ...)
```

Arguments

x	An object with S3 class "logit".
...	Arguments to be passed to methods.

Value

No return value, called for side effects (drawing the coefficient-path plot).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

plot.poisson *Plot Method for an Object with S3 Class "poisson"*

Description

Plot coefficient paths across regularization parameters.

Usage

```
## S3 method for class 'poisson'
plot(x, ...)
```

Arguments

x	An object with S3 class "poisson".
...	Arguments to be passed to methods.

Value

No return value, called for side effects (drawing the coefficient-path plot).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

---

plot.sqrtlasso *Plot Method for an Object with S3 Class "sqrtlasso"*

Description

Plot coefficient paths across regularization parameters.

Usage

```
## S3 method for class 'sqrtlasso'
plot(x, ...)
```

Arguments

x	An object with S3 class "sqrtlasso".
...	Arguments to be passed to methods.

Value

No return value, called for side effects (drawing the coefficient-path plot).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

`predict.gaussian` *Prediction Method for an Object with S3 Class* `"gaussian"`

Description

Predict responses for new data using fitted models.

Usage

```
## S3 method for class 'gaussian'
predict(object, newdata, lambda.idx = NULL, Y.pred.idx = NULL,
  type = "response", s = NULL, ...)
```

Arguments

`object`	An object with S3 class `"gaussian"`.
`newdata`	Nonempty finite numeric matrix of new observations for prediction ($n_{new} \times d$) with the same number of columns as the fitted design.
`lambda.idx`	Positive integer indices of regularization parameters along the solution path used for prediction. By default, at most the first three fitted path points are used.
`Y.pred.idx`	Optional row indices to subset returned predictions. `NULL` returns every prediction row.
`type`	Type of prediction. `"response"` (default) returns predicted values; `"link"` returns the linear predictor; `"nonzero"` returns a list of nonzero variable indices per selected lambda.
`s`	Optional nonempty vector of finite non-negative lambda *values* (not indices). When supplied, `lambda.idx` is ignored. If a requested value falls between two path lambdas the coefficients are linearly interpolated and a message is issued. Values outside the path range are clamped to the nearest endpoint. For `type = "nonzero"`, the nearest fitted lambda is used instead because support sets cannot be interpolated.
`...`	Arguments to be passed to methods.

Details

`predict.gaussian` returns predicted responses for `newdata` using fitted coefficients from `object`:

$$\hat{Y} = \hat{\beta}_0 + X_{new}\hat{\beta}.$$

Value

Return type depends on `type`:

- `"response"` or `"link"` (default): numeric matrix of predicted values $\hat{\beta}_0 + X\hat{\beta}$.
- `"nonzero"`: list of integer vectors of nonzero coefficient indices, one element per selected lambda.

Rows correspond to observations; columns correspond to `lambda.idx` (or `s` values when `s` is specified).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

`picasso` and `picasso-package`.

`predict.logit` *Prediction Method for an Object with S3 Class* `"logit"`

Description

Predict responses for new data using fitted models.

Usage

```
## S3 method for class 'logit'
predict(object, newdata, lambda.idx = NULL, p.pred.idx = NULL,
  type = "response", s = NULL, newoffset = NULL, ...)
```

Arguments

`object`	An object with S3 class `"logit"`.
`newdata`	Nonempty finite numeric matrix of new observations for prediction ($n_{new} \times d$) with the same number of columns as the fitted design.
`lambda.idx`	Positive integer indices of regularization parameters along the solution path used for prediction. By default, at most the first three fitted path points are used.
`p.pred.idx`	Optional row indices to subset returned predictions. `NULL` returns every prediction row.
`type`	Type of prediction. `"response"` (default) returns probabilities; `"link"` returns the log-odds; `"class"` returns 1 when the link-scale score is positive and 0 otherwise; `"nonzero"` returns nonzero variable indices.
`s`	Optional nonempty vector of finite non-negative lambda *values*; see `predict.gaussian`.

newoffset	Optional finite numeric vector with one offset per row of newdata. It is required for response, link, or class prediction when object was fitted with offset; type = "nonzero" does not use it. For a model fitted without offset, the default is a vector of zeros.
...	Arguments to be passed to methods.

Details

predict.logit returns predicted Bernoulli probabilities for newdata using fitted coefficients from object:

$$\hat{p} = \frac{e^{o_{new}+\hat{\beta}_0+X_{new}\hat{\beta}}}{1+e^{o_{new}+\hat{\beta}_0+X_{new}\hat{\beta}}}.$$

Value

Return type depends on type:

- "response" (default): numeric matrix of predicted probabilities $\hat{p} = \sigma(o_{new} + \hat{\beta}_0 + X\hat{\beta})$.
- "link": numeric matrix of log-odds $o_{new} + \hat{\beta}_0 + X\hat{\beta}$.
- "class": integer matrix of predicted class labels (0 or 1).
- "nonzero": list of integer vectors of nonzero coefficient indices, one element per selected lambda.

Rows correspond to observations; columns correspond to lambda.idx (or s values when s is specified).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

---

predict.poisson *Prediction Method for an Object with S3 Class "poisson"*

Description

Predict responses for new data using fitted models.

Usage

```
## S3 method for class 'poisson'
predict(object, newdata, lambda.idx = NULL, p.pred.idx = NULL,
  type = "response", s = NULL, newoffset = NULL, ...)
```

Arguments

object	An object with S3 class "poisson".
newdata	Nonempty finite numeric matrix of new observations for prediction ($n_{new} \times d$) with the same number of columns as the fitted design.
lambda.idx	Positive integer indices of regularization parameters along the solution path used for prediction. By default, at most the first three fitted path points are used.
p.pred.idx	Optional row indices to subset returned predictions. NULL returns every prediction row.
type	Type of prediction. "response" (default) returns predicted Poisson means; "link" returns the log-mean; "nonzero" returns nonzero variable indices.
s	Optional nonempty vector of finite non-negative lambda *values*; see predict.gaussian.
newoffset	Optional finite numeric vector with one offset per row of newdata. It is required for response or link prediction when object was fitted with offset; type = "nonzero" does not use it. For a model fitted without offset, the default is a vector of zeros.
...	Arguments to be passed to methods.

Details

predict.poisson returns predicted Poisson means for newdata using fitted coefficients from object:

$$\hat{\mu} = e^{o_{new} + \hat{\beta}_0 + X_{new}\hat{\beta}}.$$

Value

Return type depends on type:

- "response" (default): numeric matrix of predicted Poisson means $\hat{\mu} = \exp(o_{new} + \hat{\beta}_0 + X\hat{\beta})$.
- "link": numeric matrix of log-means $o_{new} + \hat{\beta}_0 + X\hat{\beta}$.
- "nonzero": list of integer vectors of nonzero coefficient indices, one element per selected lambda.

Rows correspond to observations; columns correspond to lambda.idx (or s values when s is specified).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

---

predict.sqrtlasso *Prediction Method for an Object with S3 Class* "sqrtlasso"

Description

Predict responses for new data using fitted models.

Usage

```
## S3 method for class 'sqrtlasso'
predict(object, newdata, lambda.idx = NULL, Y.pred.idx = NULL,
  type = "response", s = NULL, ...)
```

Arguments

object	An object with S3 class "sqrtlasso".
newdata	Nonempty finite numeric matrix of new observations for prediction ($n_{new} \times d$) with the same number of columns as the fitted design.
lambda.idx	Positive integer indices of regularization parameters along the solution path used for prediction. By default, at most the first three fitted path points are used.
Y.pred.idx	Optional row indices to subset returned predictions. NULL returns every prediction row.
type	Type of prediction. "response" (default) returns predicted values; "link" returns the linear predictor; "nonzero" returns nonzero variable indices.
s	Optional nonempty vector of finite non-negative lambda *values*; see predict.gaussian.
...	Arguments to be passed to methods.

Details

predict.sqrtlasso returns predicted responses for newdata using fitted coefficients from object:

$$\hat{Y} = \hat{\beta}_0 + X_{new}\hat{\beta}.$$

Value

Return type depends on type:

- "response" or "link" (default): numeric matrix of predicted values $\hat{\beta}_0 + X\hat{\beta}$.
- "nonzero": list of integer vectors of nonzero coefficient indices, one element per selected lambda.

Rows correspond to observations; columns correspond to lambda.idx (or s values when s is specified).

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

---

print.gaussian *Print Method for an Object with S3 Class* "gaussian"

Description

Print a concise summary for an object with S3 class "gaussian".

Usage

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

Arguments

x	An object with S3 class "gaussian".
...	Arguments to be passed to methods.

Details

This method prints tuning parameters and fit statistics for a fitted model object.

Value

The input object x, returned invisibly.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

print.logit *Print Method for an Object with S3 Class "logit"*

Description

Print a concise summary for an object with S3 class "logit".

Usage

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

Arguments

x	An object with S3 class "logit".
...	Arguments to be passed to methods.

Details

This method prints tuning parameters and fit statistics for a fitted logistic model object.

Value

The input object x, returned invisibly.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

print.poisson *Print Method for an Object with S3 Class "poisson"*

Description

Print a concise summary for an object with S3 class "poisson".

Usage

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

Arguments

x	An object with S3 class "poisson".
...	Arguments to be passed to methods.

Details

This method prints tuning parameters and fit statistics for a fitted Poisson model object.

Value

The input object x, returned invisibly.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

---

print.sqrtlasso *Print Method for an Object with S3 Class "sqrtlasso"*

Description

Print a concise summary for an object with S3 class "sqrtlasso".

Usage

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

Arguments

x	An object with S3 class "sqrtlasso".
...	Arguments to be passed to methods.

Details

This method prints tuning parameters and fit statistics for a fitted model object.

Value

The input object x, returned invisibly.

Author(s)

Jason Ge, Xingguo Li, Haoming Jiang, Mengdi Wang, Tong Zhang, Han Liu and Tuo Zhao
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

See Also

picasso and picasso-package.

Index