Package: cvLM (via r-universe)

October 29, 2026

Type Package

Title Cross-Validation for Linear and Ridge Regression Models

Version 2.0.0

Date 2026-02-01

Description Implements cross-validation methods for linear and ridge regression models. The package provides grid-based selection of the ridge penalty parameter using Singular Value Decomposition (SVD) and supports K-fold cross-validation, Leave-One-Out Cross-Validation (LOOCV), and Generalized Cross-Validation (GCV). Computations are implemented in C++ via 'RcppArmadillo' with optional parallelization using 'RcppParallel'. The methods are suitable for high-dimensional settings where the number of predictors exceeds the number of observations.

License MIT + file LICENSE

Imports stats, Rcpp (>= 1.0.13), RcppParallel (>= 5.1.8)

Suggests boot, RhpcBLASctl, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo, RcppParallel

SystemRequirements GNU make, C++17

URL https://github.com/phipnye/CV-LM

Config/testthat/edition 3

NeedsCompilation yes

Author Philip Nye [aut, cre]

Maintainer Philip Nye <phipnye@proton.me>

Config/pak/sysreqs make

Repository https://cran.r-universe.dev

Date/Publication 2026-02-03 09:20:23 UTC

RemoteUrl https://github.com/cran/cvLM

RemoteRef HEAD

RemoteSha 254763182dff55486443a74fa3f9c2f3d6c4d258

Contents

cvLM-package 2
grid.search 5
reg.table 7

Index **10**

`cvLM-package` *Cross-validation for linear and ridge regression models*

Description

`cvLM` provides a high-performance framework for cross-validating linear models using `RcppArmadillo` and `RcppParallel`. It implements numerically stable algorithms to compute K-fold, Leave-One-Out (LOO), and Generalized Cross-Validation (GCV) metrics for both linear and ridge regression.

Usage

```
cvLM(object, ...)

## S3 method for class 'formula'
cvLM(object, data, subset, na.action, K.vals = 10L,
      lambda = 0, generalized = FALSE, seed = 1L, n.threads = 1L,
      tol = 1e-7, center = TRUE, ...)

## S3 method for class 'lm'
cvLM(object, data, K.vals = 10L, lambda = 0,
      generalized = FALSE, seed = 1L, n.threads = 1L,
      tol = 1e-7, center = TRUE, ...)

## S3 method for class 'glm'
cvLM(object, data, K.vals = 10L, lambda = 0,
      generalized = FALSE, seed = 1L, n.threads = 1L,
      tol = 1e-7, center = TRUE, ...)
```

Arguments

`object`	a `formula`, `lm`, or `glm` object.
`data`	a data frame, list or environment containing the variables in the model. See `model.frame`.
`subset`	a specification of the rows/observations to be used. For `lm` and `glm` methods, this is inherited from `object`. See `model.frame`.
`na.action`	a function indicating how `NA` values should be handled. For `lm` and `glm` methods, this is inherited from `object`. See `model.frame`.
`K.vals`	an integer vector specifying the number of folds. For Leave-One-Out CV, set `K.vals` equal to the number of observations.

`lambda`	a non-negative numeric scalar for the ridge regularization parameter. Defaults to 0 (OLS).
`generalized`	if `TRUE`, the Generalized Cross-Validation (GCV) statistic is computed. `K.vals` is ignored.
`seed`	an integer used to initialize the random number generator for reproducible fold splits.
`n.threads`	an integer specifying the number of threads for parallel computation. Set to -1 to use the `RcppParallel` default (`defaultNumThreads`). If `n.threads > 1`, the package attempts to restrict the underlying BLAS to a single thread using `RhpcBLASctl` to prevent oversubscription.
`tol`	a numeric scalar specifying the tolerance for rank estimation in the underlying orthogonal and singular value decompositions. See 'Details' below.
`center`	if `TRUE` (the default), the predictors and response are mean-centered prior to fitting. See 'Details' below.
`...`	additional arguments passed to or from other methods (currently disregarded).

Details

Numerical Stability and Underdetermined Systems

The newest release of `cvLM` prioritizes numerical robustness. For Ordinary Least Squares ($\lambda = 0$), the package employs Complete Orthogonal Decomposition (COD). In the presence of column rank-deficiency—where predictors are perfectly collinear or the number of parameters p exceeds the number of observations n—COD allows for the identification of the unique minimum L_2 norm solution, providing a stable platform for cross-validation in high-dimensional or ill-conditioned settings. The numerical rank is determined by the `tol` argument; a pivot is considered nonzero if its absolute value is strictly greater than the threshold relative to the largest pivot:

$$|r_{ii}| > \text{tol} \times |r_{\text{max}}|$$

This approach differs from R's `lm`, which handles column rank-deficiency by zeroing out coefficients for redundant columns, and may result in different coefficient estimates and subsequent predictions when the design matrix is not of full-column rank.

For ridge regression ($\lambda > 0$), the package utilizes Singular Value Decomposition (SVD). By decomposing the design matrix into $X = UDV^T$, the ridge solution can be computed with high precision, avoiding the potential information loss associated with forming the cross-product matrix X^TX required for Cholesky-based methods used in previous versions. Similar to the COD, a singular value is considered nonzero only if it satisfies:

$$\sigma_i > \text{tol} \times \sigma_{\text{max}}$$

While this numerical stability comes at a speed cost, it allows for consistency with methods used for efficient shrinkage parameter search used by `grid.search`.

Regularization and Data Centering

Following the methodology in *The Elements of Statistical Learning* (Hastie et al.), `cvLM` now defaults to centering the data (`center = TRUE`). Centering the predictors X and the response y by their respective means is mathematically equivalent to excluding the intercept term from the regularization penalty.

In the **uncentered** case (`center = FALSE`), the objective function penalizes all coefficients, including the intercept β_0:

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|^2 + \lambda\|\beta\|^2$$

In the **centered** case (`center = TRUE`), the optimization problem is effectively reformulated as:

$$\hat{\beta} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

Efficiency of LOOCV and GCV

For Leave-One-Out (LOOCV) and Generalized Cross-Validation (GCV), `cvLM` avoids the computational cost of n refits by utilizing closed-form solutions derived through use of the hat matrix.

Parallel Computation and Oversubscription

`cvLM` uses `RcppParallel` to distribute K-fold cross-validation tasks across multiple threads. However, many modern R installations link to multi-threaded BLAS libraries. Running multi-threaded BLAS operations within multi-threaded folds can lead to oversubscription, where the total number of active threads exceeds the physical CPU capacity, causing significant performance penalties.

To mitigate this, if the `RhpcBLASctl` package is installed, `cvLM` will automatically attempt to set the BLAS thread count to 1 during parallel execution. It is highly recommended to install `RhpcBLASctl` when using `n.threads > 1`.

Value

A `data.frame` with the following columns:

`K` the number of folds requested by the user. Note that if the internal algorithm determines a more appropriate number of folds to use (based on the number of rows), a warning is issued and a modified fold count is used for the computation; however, this column retains the input value.

`CV` the cross-validation metric calculated during the procedure.

`seed` the integer seed used for fold randomization.

Author(s)

Philip Nye, <phipnye@gmail.com>

References

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra." Computational Statistics and Data Analysis, 71, 1054—1063. doi:10.1016/j.csda.2013.02.005.

Aggarwal, C. C. (2020). Linear Algebra and Optimization for Machine Learning: A Textbook. Springer Cham. doi:10.1007/9783030403447.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer New York, NY. doi:10.1007/978038784858-7.

Golub, G. H., & Van Loan, C. F. (2013). Matrix Computations (4th ed.). Johns Hopkins University Press, Baltimore, MD. doi:10.56021/9781421407944.

See Also

grid.search

Examples

```
## Not run:
data(mtcars)
n <- nrow(mtcars)

# 10-fold CV for a linear regression model
cvLM(mpg ~ ., data = mtcars, K.vals = 10)

# Comparing 5-fold, 10-fold, and Leave-One-Out CV configurations using 2 threads
cvLM(mpg ~ ., data = mtcars, K.vals = c(5, 10, nrow(mtcars)), n.threads = 2)

# Ridge regression with analytic GCV (using lm interface)
fitted.lm <- lm(mpg ~ ., data = mtcars)
cvLM(fitted.lm, data = mtcars, lambda = 0.5, generalized = TRUE)

## End(Not run)
```

---

grid.search *Efficient Grid Search for Optimal Ridge Regularization*

Description

Performs an optimized grid search to find the regularization parameter λ that minimizes the cross-validation metric for ridge regression.

Usage

```
grid.search(formula, data, subset, na.action, K = 10L,
            generalized = FALSE, seed = 1L, n.threads = 1L,
            tol = 1e-7, max.lambda = 10000, precision = 0.1,
            center = TRUE)
```

Arguments

formula	a formula specifying the model.
data	a data frame, list or environment containing the variables in the model. See model.frame.
subset	a specification of the rows/observations to be used. See model.frame.
na.action	a function indicating how NA values should be handled. See model.frame.
K	an integer specifying the number of folds. For Leave-One-Out CV, set K equal to the number of observations.
generalized	if TRUE, the Generalized Cross-Validation (GCV) statistic is computed. K is ignored.

seed	an integer used to initialize the random number generator for reproducible K-fold splits.
n.threads	an integer specifying the number of threads. For K-fold CV, parallelization occurs across folds; for GCV/LOOCV, it occurs across the lambda grid. Set to -1 to use the RcppParallel default (defaultNumThreads).
tol	numeric scalar specifying the tolerance for rank estimation in the SVD. See cvLM.
max.lambda	numeric scalar for the maximum λ value in the search grid.
precision	numeric scalar specifying the step size (increment) between λ values in the grid.
center	if TRUE (the default), the predictors and response are mean-centered, effectively excluding the intercept from the ridge penalty. See cvLM.

Details

grid.search is designed for high-performance parameter tuning. Unlike naive implementations that refit the model for every grid point, this function utilizes Singular Value Decomposition (SVD) of the design matrix to evaluate the entire grid analytically.

For Generalized Cross-Validation (GCV) and Leave-One-Out (LOOCV), the SVD is computed once. Each λ in the grid is then evaluated by updating the diagonal "shrinkage" matrix, reducing the cost of each grid point evaluation from $O(np^2)$ to $O(\min(n, p))$.

The search begins at $\lambda = 0$ and increments by precision until max.lambda is reached (inclusive). The function returns the λ that achieves the minimum cross-validation metric across the scheme.

Value

A list with the following components:

CV the minimum cross-validation metric.

lambda the value of λ associated with the minimum metric.

See Also

cvLM

Examples

```
## Not run:
data(mtcars)
grid.search(
  formula = mpg ~ .,
  data = mtcars,
  K = 5L,           # Use 5-fold CV
  max.lambda = 100, # Search values between 0 and 100
  precision = 0.01  # Increment in steps of 0.01
)

grid.search(
  formula = mpg ~ .,
```

```
  data = mtcars,
  K = nrow(mtcars),    # Use LOOCV
  max.lambda = 10000,  # Search values between 0 and 10000
  precision = 0.5      # Increment in steps of 0.5
)

## End(Not run)
```

---

reg.table *Create Regression Tables in LaTeX or HTML*

Description

reg.table generates formatted regression tables in either LaTeX or HTML format for one or more linear regression models. The tables include coefficient estimates, standard errors, and various model statistics.

Usage

```
reg.table(models, type = c("latex", "html"), split.size = 4L, n.digits = 3L,
          big.mark = "", caption = "Regression Results", spacing = 5L,
          stats = c("AIC", "BIC", "CV", "r.squared", "adj.r.squared",
                    "fstatistic", "nobs"),
          ...)
```

Arguments

models	A list of linear regression models (objects of class lm or glm with gaussian family and identity link function).
type	A string specifying the output format. Must be one of "latex" or "html".
split.size	An integer specifying the number of models per table. Tables with more models than this number will be split.
n.digits	An integer specifying the number of decimal places for numerical values.
big.mark	A string to use for thousands separators in numeric formatting.
caption	A string for the table caption.
spacing	A numeric specifying the spacing between columns in the output table.
stats	A character vector specifying which model statistics to include in the table. Options include: AIC Akaike Information Criterion BIC Bayesian Information Criterion CV Cross-validation metric r.squared R-squared adj.r.squared Adjusted R-squared fstatistic F-statistic nobs Number of observations
...	Additional arguments passed to cvLM.

Details

The function generates tables summarizing linear regression models, either in LaTeX or HTML format. The tables include coefficients, standard errors, and optional statistics such as AIC, BIC, cross-validation scores, and more. The output is formatted based on the specified type and options. The names of the models list will be used as column headers if provided.

Value

A character vector of length equal to the number of tables with each element containing the LaTeX or HTML code for a regression table.

Examples

```
## Not run:
data(mtcars)

# Create a list of 6 different linear regression models with names
# Names get used for column names of the table
models <- list(
  `Model 1` = lm(mpg ~ wt + hp, data = mtcars),
  `Model 2` = lm(mpg ~ wt + qsec, data = mtcars),
  `Model 3` = lm(mpg ~ wt + hp + qsec, data = mtcars),
  `Model 4` = lm(mpg ~ wt + cyl, data = mtcars),
  `Model 5` = lm(mpg ~ wt + hp + cyl, data = mtcars),
  `Model 6` = lm(mpg ~ hp + qsec + drat, data = mtcars)
)

# Example usage for LaTeX
reg.table(
  models,
  "latex",
  split.size = 3L,
  n.digits = 3L,
  big.mark = ",",
  caption = "My regression results",
  spacing = 7.5,
  stats = c("AIC", "BIC", "CV"),
  data = mtcars,
  K.vals = 5L,
  seed = 123L
)

# Example usage for HTML
reg.table(
  models,
  "html",
  split.size = 3L,
  n.digits = 3L,
  big.mark = ",",
  caption = "My regression results",
  spacing = 7.5,
  stats = c("AIC", "BIC", "CV"),
```

```
  data = mtcars,
  K.vals = 5L,
  seed = 123L
)

## End(Not run)
```

Index

∗ **package**
cvLM-package, 2

cvLM, *6*, *7*
cvLM *(*cvLM-package*)*, 2
cvLM-package, 2
cvLM.formula *(*cvLM-package*)*, 2
cvLM.glm *(*cvLM-package*)*, 2
cvLM.lm *(*cvLM-package*)*, 2

data.frame, *4*
defaultNumThreads, *3*, *6*

formula, *2*, *5*

glm, *2*
grid.search, *3*, 5, *5*

list, *6*
lm, *2*

model.frame, *2*, *5*

reg.table, 7