

Package: SILM (via r-universe)

September 29, 2026

Type Package

Title Simultaneous Inference for High-Dimensional Linear Models

Version 2.0.0

Description Simultaneous inference for high-dimensional linear models based on the de-biased (de-sparsified) lasso and the bootstrap: the procedures of Zhang and Cheng (2017) [<doi:10.1080/01621459.2016.1166114>](https://doi.org/10.1080/01621459.2016.1166114) for simultaneous confidence intervals, support recovery, testing for sparse signals and stepdown multiple testing; and the bootstrapped de-sparsified lasso of Dezeure, Buehlmann and Zhang (2017) [<doi:10.1007/s11749-017-0554-2>](https://doi.org/10.1007/s11749-017-0554-2) with residual, wild and xyz-paired bootstrap, Westfall-Young multiple testing, simultaneous confidence intervals and group tests. Includes lasso.proj() and boot.lasso.proj(), ports of the corresponding functions of the archived 'hdi' package (hdi's results can be reproduced exactly through their arguments), and a scaled lasso following Sun and Zhang (2012) [<doi:10.1093/biomet/ass043>](https://doi.org/10.1093/biomet/ass043).

Depends R (>= 4.1.0)

Imports glmnet (>= 4.1), lars, MASS, parallel, stats, utils

Suggests knitr, rmarkdown, testthat (>= 3.1.5), withr

VignetteBuilder knitr

Config/testthat/edition 3

License GPL-3

Copyright See file inst/COPYRIGHTS.

URL <https://github.com/zhangxiany-tamu/SILM>

BugReports <https://github.com/zhangxiany-tamu/SILM/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Repository <https://zhangxiany-tamu.r-universe.dev>

Date/Publication 2026-09-29 19:11:55 UTC
RemoteUrl <https://github.com/zhangxiany-tamu/SILM>
RemoteRef HEAD
RemoteSha 998f08dff96387e917059fd19a7ea8fd90030094

Contents

boot.lasso.proj	2
confint.silm_proj	7
groupTest	9
lasso.proj	10
print.silm_proj	13
Sim.CI	14
SR	17
ST	20
Step	23
Theta.hat	25
Index	28

boot.lasso.proj	<i>Bootstrapped de-sparsified lasso: p-values and confidence intervals</i>
-----------------	--

Description

Bootstraps the entire de-sparsified lasso (initial lasso fit included) to obtain individual p-values, p-values adjusted for multiple testing with the Westfall-Young max-T procedure, and bootstrap confidence intervals (Dezeure, Bühlmann and Zhang, 2017). This is a port of boot.lasso.proj() from the archived package 'hdi' (version 0.1-10) with the same arguments and defaults, plus the SILM additions below; the results differ from hdi's only where the section "Compatibility with hdi 0.1-10" says so.

Usage

```
boot.lasso.proj(  
  x,  
  y,  
  family = "gaussian",  
  standardize = TRUE,  
  multiplecorr.method = "WY",  
  parallel = FALSE,  
  ncores = getOption("mc.cores", 2L),  
  betainit = "cv lasso",  
  sigma = NULL,  
  Z = NULL,  
  verbose = FALSE,
```

```

    return.Z = FALSE,
    robust = FALSE,
    B = 1000,
    boot.shortcut = FALSE,
    return.bootdist = FALSE,
    wild = FALSE,
    gaussian.stub = FALSE,
    boot.type = if (wild) "wild" else "residual",
    multiplier = c("gaussian", "mammen"),
    boot.H0c = identical(multiplecorr.method, "WY"),
    groups = NULL,
    robust.divisor = c("n-s", "n"),
    scaled.lasso = c("equivariant", "legacy")
)

```

Arguments

<code>x</code>	Design matrix (n x p), without intercept column.
<code>y</code>	Response vector of length n.
<code>family</code>	Only "gaussian" is supported.
<code>standardize</code>	Should the columns of x be scaled to unit variance (they are always centred)?
<code>multiplecorr.method</code>	"WY" (default; Westfall-Young max-T procedure based on the bootstrap under the complete null hypothesis) or any method of stats::p.adjust() .
<code>parallel, ncores</code>	Compute the nodewise regressions (and, in <code>boot.lasso.proj()</code> , the bootstrap refits) in parallel (forking; not on Windows). Does not change the results. With a BLAS library that uses OpenMP threads, forked workers can occasionally stall; setting <code>OMP_NUM_THREADS=1</code> before starting R avoids this.
<code>betainit</code>	"cv lasso" (default) or "scaled lasso".
<code>sigma</code>	Optional noise standard deviation, overriding the estimate (not used with <code>robust = TRUE</code> , whose standard errors do not involve it, nor for <code>family = "binomial"</code>).
<code>Z</code>	Optional matrix of nodewise residuals (e.g. from a previous call with <code>return.Z = TRUE</code>) to skip the nodewise lasso. This also skips the random draw of its cross-validation folds.
<code>verbose</code>	Print progress information.
<code>return.Z</code>	Return the nodewise residuals Z.
<code>robust</code>	Use the robust (sandwich) standard errors of Dezeure, Bühlmann and Zhang (2017), valid under heteroscedastic errors.
<code>B</code>	Number of bootstrap samples.
<code>boot.shortcut</code>	Refit the bootstrap lasso at the lambda of the original fit instead of re-tuning it (faster).
<code>return.bootdist</code>	Return the bootstrap distributions (needed for confint()).
<code>wild</code>	Use the wild bootstrap with Gaussian multipliers.

gaussian.stub	Developer option of hdi: replace the bootstrap distribution by independent standard normal draws.
boot.type	"residual", "wild" or "xyz" (xyz-paired bootstrap). Defaults to "wild" if wild = TRUE, else "residual".
multiplier	Multipliers of the wild bootstrap: "gaussian" (default, as in hdi) or "mammen".
boot.H0c	Also compute the bootstrap under the complete null hypothesis (always done, and required, for multicorr.method = "WY"); needed for <code>groupTest()</code> .
groups	Optional group (vector of indices or names) or list of groups for which bootstrap summaries are stored, so that simultaneous intervals and group tests for them are available without <code>return.bootdist = TRUE</code> . A character name must identify exactly one column; use numeric indices when predictor names are duplicated.
robust.divisor	Normalisation of the robust standard error, only used with robust = TRUE: "n-s" (default) is equation (5) of Dezeure, Bühlmann and Zhang (2017), which multiplies the standard error with divisor n by $\sqrt{n/(n - \hat{s})}$, where $\hat{s}$ is the number of variables selected by the initial lasso (for a numeric <code>betainit</code> , its number of non-zero entries, which must be smaller than n ; p for a dense estimate); "n" is hdi's normalisation (Section 3.3.2 of the paper), reproduces hdi and has no such restriction. The default was chosen by a pre-registered study that compared the two on the paper's simulation designs and a heteroscedastic stress design (Gaussian linear models with lasso initial fits), under a decision rule fixed in advance (calibration first, then power): over 7 designs (100 replications each), "n-s" reduced the mean calibration shortfall from 0.055 to 0.031 (95% bootstrap interval of the difference: -0.028 to -0.018), mainly by bringing the familywise error rate closer to 5% (for example Westfall-Young 0.12 to 0.05 and Holm 0.11 to 0.05 in the paper's Toeplitz design), at a mean power loss of 0.027 (see <code>validation/calibration/DEFAULTS.md</code> and <code>validation/calibration/defaults-results/REPORT</code> in the source repository). In <code>lasso.proj()</code> it also applies to <code>family = "binomial"</code> (the linearised model) and to a numeric <code>betainit</code> , which the study did not cover.
scaled.lasso	Numerical solver when <code>betainit = "scaled lasso"</code> : "equivariant" (default) normalizes the response and uses a relative noise-level tolerance of $1e-8$; "legacy" reproduces <code>scalreg</code> 1.0.1's absolute tolerance $1e-4$ and sensitivity to response units. The default can also slightly change ordinary-scale results. Ignored for other initial estimators. In <code>boot.lasso.proj()</code> the same setting is used for all bootstrap refits.

Details

The residual bootstrap (default) resamples the centred lasso residuals; the wild bootstrap (wild = TRUE) multiplies them by independent standard normal variables and is also valid for heteroscedastic errors. For every bootstrap response the initial lasso is refitted (with the same tuning method, or with the original lambda if `boot.shortcut = TRUE`) and the de-sparsified lasso and its standard error are recomputed; the nodewise residuals Z are computed once. Westfall-Young adjusted p-values use a second bootstrap under the complete null hypothesis (all coefficients zero), with the same resampled errors.

The paper recommends the robust standard error (robust = TRUE) in practice, and the wild bootstrap under heteroscedastic errors. The default robust = FALSE is kept for compatibility with hdi. With

robust = TRUE, the robust standard errors of the original and of every bootstrap fit use the divisor $n - \hat{s}$ of the paper's equation (5) by default (see robust.divisor).

Value

An object of class c("silm_boot_lasso_proj", "silm_proj"): a list with the elements of hdi's result, in the same order, pval, pval.corr, sigmahat, standardize, sds, bhat, se, betahat, family, method("boot.lasso.proj"), B, boot.shortcut, lambda, call, Z (if return.Z = TRUE), cboot.dist and cboot.dist.underH0c (if return.bootdist = TRUE: $p \times B$ matrices of $s.e.j T_j^*$, where $T_j^* = (\hat{b}_j^* - \hat{\beta}_j)/s.e.j^*$ is the studentized centred bootstrap statistic, and the same under the complete null hypothesis, on the scale of bhat, as in hdi), followed by the SILM additions boot.type, multiplier, robust, robust.divisor (no effect when robust = FALSE), gaussian.stub, B.eff (number of bootstrap samples used; smaller than B only when xyz-paired samples had to be discarded), tstat (the studentized statistics $\hat{b}_j/s.e.j$), boot.summary (per bootstrap sample, the maximum, minimum and maximal absolute value of the T_j^* , and the maximal absolute value under the complete null hypothesis), group.summary (the same per group in groups) and boot.index (xyz-paired bootstrap with return.bootdist = TRUE: the resampled rows, one column per column of cboot.dist).

Compatibility with hdi 0.1-10

For the same data, arguments and random seed (and robust.divisor = "n" if robust = TRUE, and scaled.lasso = "legacy" if using scaled-lasso initialization), boot.lasso.proj() returns the same pval, pval.corr, bhat, se, betahat, sigmahat, lambda and bootstrap distributions as hdi::boot.lasso.proj() run sequentially, and leaves the random number generator in the same state; this is verified against the archived hdi package (with the same version of glmnet). The differences are:

- Scaled-lasso initialization and refits now use a response-normalized path and relative convergence tolerance. scaled.lasso = "legacy" restores the historical solver and its sensitivity to response units.
- robust = TRUE: the robust standard error of the original fit and of every bootstrap fit uses the divisor $n - \hat{s}$ of equation (5) of Dezeure, Bühlmann and Zhang (2017) by default (robust.divisor = "n-s", with $\hat{s}$ the size of the respective lasso fit); hdi divided by n (Section 3.3.2 of the paper). se and, through the studentized bootstrap statistics, the p-values and bootstrap distributions differ from hdi's; robust.divisor = "n" reproduces hdi exactly. Results with robust = FALSE (the default) are not affected.
- The result has class c("silm_boot_lasso_proj", "silm_proj") instead of "hdi", and additional elements (see Value).
- With parallel = TRUE, hdi drew the cross-validation folds of the bootstrap refits in the worker processes, so its results were not reproducible. SILM draws them in the main process in the same order as a sequential run: parallel results are identical to sequential ones and to hdi's sequential results.
- Arguments are checked before the computations start; parallel execution falls back to sequential on Windows.

Documented properties of hdi that are kept: a user-supplied sigma only affects the standard errors of the original fit, not those of the bootstrap fits (a warning is given); boot.shortcut has no effect

with `betainit = "scaled lasso"` (a warning is given) and uses `glmnet`'s linear interpolation along its lambda path; the individual p-values are $(2c + 1) / (B + 1)$, where c is the smaller of the two tail counts of the bootstrap distribution.

SILM additions (not in hdi)

- `boot.type = "wild"` with `multiplier = "mammen"`: the wild bootstrap with Mammen's two-point multipliers (Section 4.1), which also match the third moment of the errors. The paper found no advantage over Gaussian multipliers; in our replication of its small heteroscedastic example ($n = 50$, `robust.divisor = "n"`) they under-covered (average 0.86 vs 0.94), so the Gaussian default is recommended.
- `boot.type = "xyz"`: the xyz-paired bootstrap (Section 4.2), which resamples rows of the design, the response and the nodewise residuals after a correction that makes the bootstrap errors orthogonal to them. It is defined with `robust = TRUE`; the paper reports that it is less competitive than the Gaussian wild bootstrap. When the lasso is re-tuned by cross-validation, the folds are formed by original observation, since resampled rows are duplicated. Bootstrap samples with a degenerate normaliser $Z_j^{*\top} X_j^* / n$ are discarded (`B.eff`).
- Simultaneous confidence intervals (eq. 10) via `confint(fit, type = "simultaneous")` and group tests via `groupTest()`.

Validity (the paper's Theorems 1-3): the residual bootstrap is valid for individual inference, and for simultaneous inference under homoscedastic errors; the wild bootstrap with `robust = TRUE` is valid for individual and simultaneous inference under heteroscedastic errors and is the paper's preferred method in that case.

Finite-sample behaviour

Validity is asymptotic. Two simulation studies in the source repository (`validation/calibration/`; $n = 100$, $p = 120$ or 500 , robust standard errors, $B = 499$, 100-200 replications per setting) found:

- Individual coverage averaged over all coefficients was 0.91-0.96, but lower for the non-zero coefficients (0.79-0.93), for the reason explained in `Sim.CI()`.
- With the default `robust.divisor = "n-s"`, the Westfall-Young FWER was 0.03-0.12 in the designs with 3 signals and 0.33 with 15 signals (" n ": 0.05-0.21 and 0.57); joint coverage (eq. 10) was 0.77-0.96.
- `boot.shortcut = TRUE` (refitting at the original lambda) was about 7 times faster than the default full refit and at least as well calibrated (FWER 0.05 vs 0.04 and 0.06 vs 0.065 with " $n-s$ "; 0.12 vs 0.14 and 0.075 vs 0.155 with " n "). The default stays the full refit, as in hdi and the paper.
- The bootstrap p-values and intervals are empirical quantiles, so small B gives slightly lower coverage; use the default $B = 1000$ or more for final results.

References

- Dezeure, R., Bühlmann, P. and Zhang, C.-H. (2017). High-dimensional simultaneous inference with the bootstrap. *TEST*, 26, 685-719.
- Dezeure, R., Bühlmann, P., Meier, L. and Meinshausen, N. (2015). High-dimensional inference: confidence intervals, p-values and R-software hdi. *Statistical Science*, 30, 533-558.

See Also

[lasso.proj\(\)](#), [confint.silm_proj\(\)](#), [groupTest\(\)](#), [Sim.CI\(\)](#)

Examples

```
set.seed(1)
x <- matrix(rnorm(50 * 15), 50, 15)
y <- x[, 1] + rnorm(50)
fit <- boot.lasso.proj(x, y, B = 50, boot.shortcut = TRUE, return.bootdist = TRUE)
confint(fit, parm = 1:3)
```

confint.silm_proj	<i>Confidence intervals from the de-sparsified lasso</i>
-------------------	--

Description

Individual or simultaneous confidence intervals for the coefficients.

Usage

```
## S3 method for class 'silm_proj'
confint(
  object,
  parm,
  level = 0.95,
  type = c("individual", "simultaneous"),
  group = NULL,
  simult.stat = c("maxmin", "abs"),
  ...
)
```

Arguments

object	A result of lasso.proj() or boot.lasso.proj() .
parm	Coefficients (indices or names) for which intervals are returned; by default all coefficients (or all of group). Use numeric indices when predictor names are duplicated; a name must identify one coefficient unambiguously.
level	Confidence level (joint level for simultaneous intervals).
type	"individual" (default) or "simultaneous".
group	For simultaneous intervals: the group G of coefficients covered jointly (indices, names or a logical vector); must contain parm. Defaults to parm. Names must identify coefficients unambiguously.
simult.stat	"maxmin" (default; eq. 10) or "abs".
...	Not used.

Details

Individual intervals (type = "individual") come from the asymptotic Gaussian distribution for `lasso.proj()` and from the bootstrap distribution for `boot.lasso.proj()` (which requires `return.bootdist = TRUE`); they are identical to those of the archived 'hdi' package when the fit used hdi's standard errors (robust = FALSE, or robust = TRUE with robust.divisor = "n"). With the default robust.divisor = "n-s", the robust intervals differ from hdi's: those of `lasso.proj()` are wider by the factor $\sqrt{n/(n-\hat{s})}$, and the bootstrap intervals change because $\hat{s}e.j$ and every $\hat{s}e.j^*$ are rescaled. For the bootstrap, the interval for β_j is $[\hat{b}_j - q_{j;1-\alpha/2}^*, \hat{b}_j - q_{j;\alpha/2}^*]$, where $q_{j;\nu}^*$ are quantiles of the bootstrap distribution of $\hat{s}e.j T_j^*$ (Dezeure, Bühlmann and Zhang, 2017, eq. 9). As in hdi, quantile type 1 is used when $B(1 - level)/2$ is an integer and type 7 otherwise; because of floating-point rounding this test fails for common levels such as 0.95, so type 7 is used in practice.

Simultaneous intervals (type = "simultaneous", `boot.lasso.proj()` only; not available in hdi) cover all $\beta_j, j \in G$, jointly with probability approximately level (Dezeure, Bühlmann and Zhang, 2017, Section 3.2, eq. 10):

- `simult.stat = "maxmin"`: $[\hat{b}_j - \hat{s}e.j q_{\max;G}^*(1 - \alpha/2), \hat{b}_j - \hat{s}e.j q_{\min;G}^*(\alpha/2)]$, where $q_{\max;G}^*$ and $q_{\min;G}^*$ are quantiles of the bootstrap distributions of $\max_{j \in G} T_j^*$ and $\min_{j \in G} T_j^*$;
- `simult.stat = "abs"`: $\hat{b}_j \mp \hat{s}e.j q_{abs;G}^*(1 - \alpha)$, with the quantile of $\max_{j \in G} |T_j^*|$.

Here $T_j^* = (\hat{b}_j - \beta_j)/\hat{s}e.j^*$ is the studentized centred bootstrap statistic (the factor $\sqrt{n}$ in the display of the region $C(1 - \alpha)$ on p. 693 of the paper is a typo). The quantile type follows the rule above. The group G defaults to parm (all coefficients if both are missing); intervals over all coefficients, or over a group given in groups when fitting, are available without `return.bootdist = TRUE`. The residual bootstrap is valid for simultaneous inference only under homoscedastic errors; use the wild (or xyz-paired) bootstrap otherwise (the paper's Theorems 1 and 3).

Value

A matrix with columns lower and upper and one row per coefficient. Simultaneous intervals carry an attribute "simultaneous" with the group, the statistic, the level and the bootstrap quantiles.

References

Dezeure, R., Bühlmann, P. and Zhang, C.-H. (2017). High-dimensional simultaneous inference with the bootstrap. *TEST*, 26, 685-719.

See Also

`lasso.proj()`, `boot.lasso.proj()`, `groupTest()`

Examples

```
set.seed(1)
x <- matrix(rnorm(50 * 15), 50, 15)
y <- x[, 1] + rnorm(50)
fit <- boot.lasso.proj(x, y, B = 50, boot.shortcut = TRUE, wild = TRUE,
                      robust = TRUE, return.bootdist = TRUE)
confint(fit, parm = 1:3)                                # individual
confint(fit, type = "simultaneous")[1:3, ]              # all 15 jointly
```

```
confint(fit, type = "simultaneous", group = 1:5) # the first 5 jointly
```

groupTest	<i>Group tests with the bootstrap</i>
-----------	---------------------------------------

Description

P-values for the null hypotheses $H_{0,G} : \beta_j = 0$ for all $j \in G$, based on the max-type statistic $\max_{j \in G} |T_j|$, $T_j = \hat{b}_j / s.e._j$, calibrated by the bootstrap under the complete null hypothesis (Dezeure, Bühlmann and Zhang, 2017, Sections 3.2 and 4.3):

$$P_G = \frac{1 + \#\{b : \max_{j \in G} |T_{jb}^{*0}| \geq \max_{j \in G} |T_j|\}}{B + 1},$$

with the same conventions as the Westfall-Young adjusted p-values of `boot.lasso.proj()`; in particular, the p-value of the group of all coefficients equals the smallest adjusted p-value. This test is not available in hdi's `boot.lasso.proj()`.

Usage

```
groupTest(object, group, ...)
```

Arguments

object	A result of <code>boot.lasso.proj()</code> .
group	A group of coefficients (indices, names or a logical vector), or a list of groups. Names must identify coefficients unambiguously; use numeric indices when predictor names are duplicated.
...	Not used.

Details

No adjustment is made across several groups. The fit must contain the bootstrap under the complete null hypothesis (`multiplecorr.method = "WY"`, the default, or `boot.H0c = TRUE`) and, for groups other than all coefficients, either `return.bootdist = TRUE` or the group in groups.

Value

The p-values, one per group (named if group is a named list).

References

Dezeure, R., Bühlmann, P. and Zhang, C.-H. (2017). High-dimensional simultaneous inference with the bootstrap. *TEST*, 26, 685-719.

See Also

`boot.lasso.proj()`, `confint.silm_proj()`

Examples

```
set.seed(1)
x <- matrix(rnorm(50 * 15), 50, 15)
y <- x[, 1] + rnorm(50)
fit <- boot.lasso.proj(x, y, B = 50, boot.shortcut = TRUE, return.bootdist = TRUE)
groupTest(fit, list(signal = 1:3, noise = 4:15))
```

lasso.proj

P-values and confidence intervals based on the de-sparsified lasso

Description

Computes the de-sparsified (de-biased) lasso of van de Geer et al. (2014) and Zhang and Zhang (2014), individual p-values based on its asymptotic Gaussian distribution, and p-values adjusted for multiple testing. This is a port of `lasso.proj()` from the archived package 'hdi' (version 0.1-10) and has the same arguments and defaults, plus a few additions; the results differ from hdi's only where the section "Compatibility with hdi 0.1-10" says so. Confidence intervals are obtained with [confint\(\)](#).

Usage

```
lasso.proj(
  x,
  y,
  family = "gaussian",
  standardize = TRUE,
  multiplecorr.method = "holm",
  N = 10000,
  parallel = FALSE,
  ncores = getOption("mc.cores", 2L),
  betainit = "cv lasso",
  sigma = NULL,
  Z = NULL,
  verbose = FALSE,
  return.Z = FALSE,
  suppress.grouptesting = FALSE,
  robust = FALSE,
  do.ZnZ = FALSE,
  legacy = FALSE,
  robust.divisor = c("n-s", "n"),
  scaled.lasso = c("equivariant", "legacy")
)
```

Arguments

x	Design matrix (n x p), without intercept column.
y	Response vector of length n.

family	"gaussian" or "binomial".
standardize	Should the columns of x be scaled to unit variance (they are always centred)?
multiplecorr.method	Multiple testing adjustment: "WY" (a Westfall-Young type procedure based on simulated Gaussian vectors) or any method of <code>stats::p.adjust()</code> (default "holm").
N	Number of Monte Carlo samples for <code>multiplecorr.method</code> = "WY".
parallel, ncores	Compute the nodewise regressions (and, in <code>boot.lasso.proj()</code> , the bootstrap refits) in parallel (forking; not on Windows). Does not change the results. With a BLAS library that uses OpenMP threads, forked workers can occasionally stall; setting <code>OMP_NUM_THREADS=1</code> before starting R avoids this.
betainit	Initial estimator: "cv lasso" (default; lasso with <code>lambda.1se</code> from 10-fold cross-validation), "scaled lasso", or a numeric vector of coefficients for the centred (and scaled) design, which requires <code>sigma</code> . With <code>robust = TRUE</code> and the default <code>robust.divisor = "n-s"</code> , a numeric <code>betainit</code> must have fewer than n non-zero entries (see <code>robust.divisor</code>).
sigma	Optional noise standard deviation, overriding the estimate (not used with <code>robust = TRUE</code> , whose standard errors do not involve it, nor for <code>family = "binomial"</code>).
Z	Optional matrix of nodewise residuals (e.g. from a previous call with <code>return.Z = TRUE</code>) to skip the nodewise lasso. This also skips the random draw of its cross-validation folds.
verbose	Print progress information.
return.Z	Return the nodewise residuals Z.
suppress.grouptesting	In hdi this skipped the preparation of the group tests. Here it only decides whether the corresponding random draws are consumed (FALSE, as hdi's default) or not.
robust	Use the robust (sandwich) standard errors of Dezeure, Bühlmann and Zhang (2017), valid under heteroscedastic errors.
do.ZnZ	Choose the nodewise tuning parameter with the Z&Z rule.
legacy	Reproduce hdi exactly for <code>family = "binomial"</code> (with <code>robust = TRUE</code> , also set <code>robust.divisor = "n"</code> ; see the section "Compatibility with hdi 0.1-10").
robust.divisor	Normalisation of the robust standard error, only used with <code>robust = TRUE</code> : "n-s" (default) is equation (5) of Dezeure, Bühlmann and Zhang (2017), which multiplies the standard error with divisor n by $\sqrt{n/(n - \hat{s})}$, where $\hat{s}$ is the number of variables selected by the initial lasso (for a numeric <code>betainit</code> , its number of non-zero entries, which must be smaller than n; p for a dense estimate); "n" is hdi's normalisation (Section 3.3.2 of the paper), reproduces hdi and has no such restriction. The default was chosen by a pre-registered study that compared the two on the paper's simulation designs and a heteroscedastic stress design (Gaussian linear models with lasso initial fits), under a decision rule fixed in advance (calibration first, then power): over 7 designs (100 replications each), "n-s" reduced the mean calibration shortfall from 0.055 to 0.031 (95% bootstrap interval of the difference: -0.028 to -0.018), mainly by bringing the familywise

error rate closer to 5% (for example Westfall-Young 0.12 to 0.05 and Holm 0.11 to 0.05 in the paper's Toeplitz design), at a mean power loss of 0.027 (see `validation/calibration/DEFAULTS.md` and `validation/calibration/defaults-results/REPORT` in the source repository). In `lasso.proj()` it also applies to `family = "binomial"` (the linearised model) and to a numeric `betainit`, which the study did not cover.

`scaled.lasso` Numerical solver when `betainit = "scaled lasso"`: "equivariant" (default) normalizes the response and uses a relative noise-level tolerance of $1e-8$; "legacy" reproduces `scalreg` 1.0.1's absolute tolerance $1e-4$ and sensitivity to response units. The default can also slightly change ordinary-scale results. Ignored for other initial estimators. In `boot.lasso.proj()` the same setting is used for all bootstrap refits.

Details

The columns of x are centred (and scaled if `standardize = TRUE`) and y is centred, so the model may contain an intercept. The nodewise lasso that estimates the matrix Z uses one tuning parameter for all regressions, chosen by 10-fold cross-validation (or with the Z&Z rule if `do.ZnZ = TRUE`). For `family = "binomial"`, the method is applied to the linearised (IRLS) working model of a logistic lasso fit, as in `hdi`.

Value

An object of class `c("silml_lasso_proj", "silml_proj")`: a list with elements `pval` (individual p-values), `pval.corr` (adjusted p-values), `groupTest` and `clusterGroupTest` (both `NULL`), `sigmahat`, `standardize`, `sds` (column standard deviations of x , or ones), `bhat` (de-sparsified lasso estimates), `se` (their standard errors), `betahat` (initial estimate), `family`, `method` ("lasso.proj"), `call`, `Z` (if `return.Z = TRUE`), and the SILM additions `robust`, `multiplecorr.method`, `legacy`, `robust.divisor` (the normalisation of the robust standard errors; it has no effect when `robust = FALSE`) and `tstat` (the statistics `bhat / se` on the internal scale).

Compatibility with `hdi` 0.1-10

For the same data, arguments and random seed (and `robust.divisor = "n"` if `robust = TRUE`, and `scaled.lasso = "legacy"` if using scaled-lasso initialization), `lasso.proj()` returns the same `pval`, `pval.corr`, `bhat`, `se`, `betahat` and `sigmahat` as `hdi::lasso.proj()` and leaves the random number generator in the same state; this is verified against the archived `hdi` package (with the same version of `glmnet`; `multiplecorr.method = "WY"` also depends on the BLAS/LAPACK used by `MASS::mvrnorm()`). The differences are:

- Scaled-lasso initialization now uses a response-normalized path and relative convergence tolerance. Use `scaled.lasso = "legacy"` for the historical solver, which is sensitive to response units.
- `robust = TRUE`: the robust standard error uses the divisor $n - \hat{s}$ of equation (5) of Dezeure, Bühlmann and Zhang (2017) by default (`robust.divisor = "n-s"`), so its standard errors are larger by the factor $\sqrt{n/(n - \hat{s})}$ and its p-values are at least as large as `hdi`'s. `hdi` divided by n (Section 3.3.2 of the paper); `robust.divisor = "n"` reproduces `hdi` exactly. Results with `robust = FALSE` (the default) are not affected.
- The result has class `c("silml_lasso_proj", "silml_proj")` instead of "hdi", and the elements `groupTest` and `clusterGroupTest` (functions for group tests in `hdi`) are `NULL`: those

tests are not provided. The draws hdi made to prepare them are still consumed from the random number stream unless `suppress.grouptest` = TRUE, so later results in a session are unaffected.

- `family = "binomial"`: hdi removed the intercept of the working model by mean-centring, which leaves it in the model when the weights vary and biases the estimates and the confidence intervals. SILM projects out the intercept direction $\sqrt{w}$ instead. Use `legacy = TRUE` to obtain hdi's results. For `family = "binomial"` a logical or two-level factor response is also accepted (with or without legacy); for `"gaussian"`, a logical response is used as 0/1, as in hdi.
- Arguments are checked before the computations start.

Documented properties of hdi that are kept: a numeric `betainit` refers to the centred and (if `standardize = TRUE`) scaled design; with the "cv lasso" initial fit, `sigmahat` is $\|y - \hat{y}\| / \sqrt{n - \hat{s} - 1}$ (the intercept of the glmnet fit is counted); the WY adjustment uses the homoscedastic covariance `crossprod(Z)` even when `robust = TRUE`, and its adjusted p-values can be 0 (below the Monte Carlo resolution $1/N$).

References

- van de Geer, S., Bühlmann, P., Ritov, Y. and Dezeure, R. (2014). On asymptotically optimal confidence regions and tests for high-dimensional models. *Annals of Statistics*, 42, 1166-1202.
- Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society, Series B*, 76, 217-242.
- Dezeure, R., Bühlmann, P., Meier, L. and Meinshausen, N. (2015). High-dimensional inference: confidence intervals, p-values and R-software hdi. *Statistical Science*, 30, 533-558.
- Dezeure, R., Bühlmann, P. and Zhang, C.-H. (2017). High-dimensional simultaneous inference with the bootstrap. *TEST*, 26, 685-719.

See Also

[boot.lasso.proj\(\)](#), [confint.silm_proj\(\)](#)

Examples

```
set.seed(1)
x <- matrix(rnorm(50 * 20), 50, 20)
y <- x[, 1] - x[, 2] + rnorm(50)
fit <- lasso.proj(x, y)
fit
confint(fit, parm = 1:3)
```

print.silm_proj	<i>Print results of lasso.proj and boot.lasso.proj</i>
-----------------	--

Description

Print results of lasso.proj and boot.lasso.proj

Usage

```
## S3 method for class 'silm_proj'
print(x, n = 10L, ...)
```

Arguments

x	A result of <code>lasso.proj()</code> or <code>boot.lasso.proj()</code> .
n	Maximal number of coefficients shown (those with the smallest adjusted p-values).
...	Not used.

Details

Duplicated predictor names are made unique in the printed display; the names stored in x are unchanged.

Value

x, invisibly.

Sim.CI

Simultaneous Confidence Intervals

Description

Constructs simultaneous confidence intervals for β_j , $j \in \text{set}$, based on the de-biased lasso and a Gaussian multiplier bootstrap of the maximum statistic (Zhang and Cheng, 2017, Section 2). Both the non-studentized and the studentized versions are returned.

Usage

```
Sim.CI(
  X,
  Y,
  set,
  M = 500,
  alpha = 0.95,
  nodewise = c("ZnZ", "cv"),
  center = FALSE,
  Theta = NULL,
  parallel = FALSE,
  ncores = getOption("mc.cores", 2L),
  scaled.lasso = c("equivariant", "legacy")
)
```

Arguments

<code>X</code>	Design matrix ($n \times p$).
<code>Y</code>	Response variable (a numeric vector of length n , or an $n \times 1$ matrix).
<code>set</code>	The set of variables of interest: column indices of X , or a logical vector of length p .
<code>M</code>	The number of bootstrap replications (default 500).
<code>alpha</code>	The confidence level of the simultaneous intervals (default 0.95).
<code>nodewise</code>	Tuning rule for the nodewise lasso that estimates Θ (the inverse of the Gram matrix) when $p > \text{floor}(n/2)$: "ZnZ" (default) refines the lambda minimising 10-fold cross-validation error pooled over all nodewise regressions with the rule of Zhang and Zhang (2014); "cv" uses that cross-validated lambda itself, as described in Zhang and Cheng (2017, Section 5). See the section "Nodewise tuning" below.
<code>center</code>	Logical. If TRUE, the columns of X and Y are centred before the analysis. The default FALSE uses the data as given (as SILM 1.0.0 did) and warns when they do not appear to be centred.
<code>Theta</code>	Optional precomputed $\hat{\Theta}$ (a $p \times p$ matrix, e.g. from <code>Theta.hat()</code>) to avoid recomputing it. Note that supplying it skips the random draw of the nodewise cross-validation folds, and that nodewise is then not used: the tuning rule is the one <code>Theta</code> was computed with. <code>Theta.hat()</code> records it (attribute "nodewise"), and a warning is given when nodewise is supplied explicitly and differs.
<code>parallel, ncores</code>	Compute the nodewise regressions in parallel on <code>ncores</code> cores (forking; not on Windows). This does not change the results. With a BLAS library that uses OpenMP threads, forked workers can occasionally stall; setting the environment variable <code>OMP_NUM_THREADS=1</code> before starting R avoids this.
<code>scaled.lasso</code>	Numerical solver for the scaled lasso. The default "equivariant" normalizes the response before building the lasso path and stops at a relative noise-level change of $1e-8$, so estimates and intervals scale with the response units. "legacy" reproduces scalreg 1.0.1's absolute tolerance $1e-4$ and its sensitivity to response units; use it only to reproduce archived results. This setting can also change ordinary-scale results slightly because convergence is tighter.

Details

Note on alpha. Unlike `ST()` and `Step()`, where alpha is the significance level, here alpha is the **confidence level** of the simultaneous intervals (default 0.95, i.e. 95% intervals). This is kept for compatibility with SILM 1.0.0; a value below 0.5 triggers a warning.

See `SR()` for the model assumptions (no intercept, centred data) and the estimation of Θ .

Value

A list with three components:

de-biased Lasso

the de-biased lasso estimates of β_j , $j \in \text{set}$;

`band.nst` a 2 x `|set|` matrix whose rows `low.nst` and `up.nst` are the lower and upper bounds of the intervals from the non-studentized statistic;

`band.st` the same for the studentized statistic (rows `low.st`, `up.st`).

Finite-sample behaviour

The intervals are asymptotically valid. In finite samples the de-biased lasso keeps a bias term, $(I - \hat{\Theta}\hat{\Sigma})(\hat{\beta} - \beta)$, that the bootstrap does not reflect. It mainly affects the intervals for the non-zero coefficients, and it grows with strong correlation, with signals of mixed signs and with the number of signals. In the defaults study of the source repository (`validation/calibration/defaults-results/REPORT.md`; $n = 100$, the paper's designs, 200 replications each), the joint coverage of 95% intervals over the zero coefficients was at least 0.86 in every setting, but over S_0 it was 0.84-0.97 for $s_0 = 3$ and 0.47-0.76 for $s_0 = 15$ with positive signals (with `nodewise = "cv"`: 0.80-0.96 and 0.31-0.60), and 0.45 for the signals (1.5, -1, 2) in the Toeplitz design with $p = 500$ ("cv": 0.21). Treat intervals for sets that contain strong signals with caution unless s_0 is small.

Nodewise tuning

SILM 1.0.0 obtained Θ from an internal function of the 'hdi' package. With hdi 0.1-6, current when SILM 1.0.0 was released (January 2019), the tuning parameter was the cross-validated lambda described in Zhang and Cheng (2017, Section 5) ("cv"). hdi 0.1-7 (March 2019) silently changed the default of that internal function to the Z&Z rule, so from then until SILM was archived (July 2026) SILM installations computed `nodewise = "ZnZ"` without saying so.

The default "ZnZ" of SILM 2.0.0 was chosen by a pre-registered study that compared the two rules on the simulation designs of Zhang and Cheng (2017) and on stress designs, under a decision rule fixed in advance (calibration first, then power): over 20 `Sim.CI()/Step()` settings, 8 `SR()` settings and 3 `ST()` settings (200 replications each), "ZnZ" halved the mean calibration shortfall (coverage below 95%, error rates above 5%) from 0.100 to 0.053 (95% bootstrap interval of the difference: -0.051 to -0.044). The joint coverage of `Sim.CI()` for all p coefficients was higher or equal in all 20 settings (for example 0.77 with "cv" and 0.90 with "ZnZ" for signals (1.5, -1, 2) at $p = 500$), and exact support recovery by `SR()` improved; the price is about 12% wider intervals and 0.02-0.07 less power of `Step()` and `ST()` (see `validation/calibration/DEFAULTS.md` and `validation/calibration/defaults-results/REPORT.md` in the source repository). With the default, the results equal those of SILM 1.0.0 as installed from 2019 to 2026 (with hdi 0.1-7 to 0.1-10; for `ST()` together with `legacy = TRUE`). Use `nodewise = "cv"` for the tuning of the paper; it reproduces SILM 1.0.0 with hdi 0.1-6. Exact archived reproduction also requires `scaled.lasso = "legacy"`. Both `nodewise` settings reproduce the respective archived versions exactly (same numbers under the same random seed).

`lasso.proj()` and `boot.lasso.proj()` keep hdi's default for their `nodewise` residuals (do `ZnZ = FALSE`, the cross-validated lambda).

References

- Zhang, X., and Cheng, G. (2017) Simultaneous Inference for High-dimensional Linear Models, *Journal of the American Statistical Association*, 112, 757-768.
- Sun, T. and Zhang, C.-H. (2012). Scaled sparse linear regression. *Biometrika*, 99, 879-898.
- Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society, Series B*, 76, 217-242.

See Also

[SR\(\)](#), [Step\(\)](#), [ST\(\)](#); `boot.lasso.proj()` for the bootstrapped de-sparsified lasso of Dezeure, Bühlmann and Zhang (2017).

Examples

```
## The function is intended for large n and p.
## Use small p here for illustration purpose only.
set.seed(1)
n <- 100
p <- 10
s0 <- 3
set <- 1:s0
Sigma <- matrix(NA, p, p)
for (i in 1:p) Sigma[i,] <- 0.9^(abs(i-(1:p)))
X <- matrix(rnorm(n*p), n, p)
X <- t(t(chol(Sigma))%*%t(X))
beta <- rep(0,p)
beta[1:s0] <- runif(s0,0,2)
Y <- X%*%beta+rt(n,4)/sqrt(2)
Sim.CI(X, Y, set)
```

SR

*Support Recovery Procedure***Description**

Implements the support recovery procedure of Zhang and Cheng (2017, Section 3.1): variable j is selected when its studentized de-biased lasso statistic $\sqrt{n}|\check{\beta}_j|/\hat{\omega}_{jj}^{1/2}$ exceeds $\sqrt{2\log p}$.

Usage

```
SR(
  X,
  Y,
  nodewise = c("ZnZ", "cv"),
  center = FALSE,
  Theta = NULL,
  parallel = FALSE,
  ncores = getOption("mc.cores", 2L),
  scaled.lasso = c("equivariant", "legacy")
)
```

Arguments

X Design matrix (n x p).

Y Response variable (a numeric vector of length n, or an n x 1 matrix).

nodewise	Tuning rule for the nodewise lasso that estimates Θ (the inverse of the Gram matrix) when $p > \text{floor}(n/2)$: "ZnZ" (default) refines the lambda minimising 10-fold cross-validation error pooled over all nodewise regressions with the rule of Zhang and Zhang (2014); "cv" uses that cross-validated lambda itself, as described in Zhang and Cheng (2017, Section 5). See the section "Nodewise tuning" below.
center	Logical. If TRUE, the columns of X and Y are centred before the analysis. The default FALSE uses the data as given (as SILM 1.0.0 did) and warns when they do not appear to be centred.
Theta	Optional precomputed $\hat{\Theta}$ (a $p \times p$ matrix, e.g. from <code>Theta.hat()</code>) to avoid recomputing it. Note that supplying it skips the random draw of the nodewise cross-validation folds, and that nodewise is then not used: the tuning rule is the one Theta was computed with. <code>Theta.hat()</code> records it (attribute "nodewise"), and a warning is given when nodewise is supplied explicitly and differs.
parallel, ncores	Compute the nodewise regressions in parallel on ncores cores (forking; not on Windows). This does not change the results. With a BLAS library that uses OpenMP threads, forked workers can occasionally stall; setting the environment variable <code>OMP_NUM_THREADS=1</code> before starting R avoids this.
scaled.lasso	Numerical solver for the scaled lasso. The default "equivariant" normalizes the response before building the lasso path and stops at a relative noise-level change of $1e-8$, so estimates and intervals scale with the response units. "legacy" reproduces scalreg 1.0.1's absolute tolerance $1e-4$ and its sensitivity to response units; use it only to reproduce archived results. This setting can also change ordinary-scale results slightly because convergence is tighter.

Details

The model is $Y = X\beta + \epsilon$ **without intercept**: the columns of X and the response Y are assumed to be centred (use `center = TRUE` to centre them), and X must not contain a column of ones. The initial estimator is the scaled lasso (Sun and Zhang, 2012). When $p \leq \text{floor}(n/2)$, $\hat{\Theta}$ is the inverse of the Gram matrix, so the de-biased estimator equals the least-squares estimator (without intercept); otherwise $\hat{\Theta}$ is estimated by the nodewise lasso.

Value

A list with two integer vectors: "de-biased Lasso", the variables selected by the support recovery procedure, and "scaled Lasso", the support of the scaled lasso.

Nodewise tuning

SILM 1.0.0 obtained Θ from an internal function of the 'hdi' package. With hdi 0.1-6, current when SILM 1.0.0 was released (January 2019), the tuning parameter was the cross-validated lambda described in Zhang and Cheng (2017, Section 5) ("cv"). hdi 0.1-7 (March 2019) silently changed the default of that internal function to the Z&Z rule, so from then until SILM was archived (July 2026) SILM installations computed `nodewise = "ZnZ"` without saying so.

The default "ZnZ" of SILM 2.0.0 was chosen by a pre-registered study that compared the two rules on the simulation designs of Zhang and Cheng (2017) and on stress designs, under a decision

rule fixed in advance (calibration first, then power): over 20 `Sim.CI()`/`Step()` settings, 8 `SR()` settings and 3 `ST()` settings (200 replications each), "ZnZ" halved the mean calibration shortfall (coverage below 95%, error rates above 5%) from 0.100 to 0.053 (95% bootstrap interval of the difference: -0.051 to -0.044). The joint coverage of `Sim.CI()` for all p coefficients was higher or equal in all 20 settings (for example 0.77 with "cv" and 0.90 with "ZnZ" for signals (1.5, -1, 2) at $p = 500$), and exact support recovery by `SR()` improved; the price is about 12% wider intervals and 0.02-0.07 less power of `Step()` and `ST()` (see `validation/calibration/DEFAULTS.md` and `validation/calibration/defaults-results/REPORT.md` in the source repository). With the default, the results equal those of `SILM 1.0.0` as installed from 2019 to 2026 (with hdi 0.1-7 to 0.1-10; for `ST()` together with `legacy = TRUE`). Use `nodewise = "cv"` for the tuning of the paper; it reproduces `SILM 1.0.0` with hdi 0.1-6. Exact archived reproduction also requires `scaled.lasso = "legacy"`. Both `nodewise` settings reproduce the respective archived versions exactly (same numbers under the same random seed).

`lasso.proj()` and `boot.lasso.proj()` keep hdi's default for their `nodewise` residuals (do. `ZnZ = FALSE`, the cross-validated `lambda`).

References

- Zhang, X., and Cheng, G. (2017) Simultaneous Inference for High-dimensional Linear Models, *Journal of the American Statistical Association*, 112, 757-768.
- Sun, T. and Zhang, C.-H. (2012). Scaled sparse linear regression. *Biometrika*, 99, 879-898.
- Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society, Series B*, 76, 217-242.

See Also

`Sim.CI()`, `Step()`, `ST()`, `Theta.hat()`

Examples

```
## The function is intended for large n and p.
## Use small p here for illustration purpose only.
set.seed(1)
n <- 100
p <- 10
s0 <- 7
set <- 1:s0
Sigma <- matrix(NA, p, p)
for (i in 1:p) Sigma[i,] <- 0.9^(abs(i-(1:p)))
X <- matrix(rnorm(n*p), n, p)
X <- t(t(chol(Sigma))%*%t(X))
beta <- rep(0,p)
beta[1:s0] <- runif(s0,1,2)
Y <- X%*%beta+rt(n,4)/sqrt(2)
SR(X, Y)
```

Description

Tests $H_{0,G} : \beta_j = 0$ for all $j \in G = \text{test.set}$, a possibly large group of variables, with the three-step procedure of Zhang and Cheng (2017, Sections 3.2 and 5.3): (1) the sample is split at random into D_1 of size `sub.size` and D_2 ; (2) on D_1 the variables are screened (the cross-validated lasso selection, completed by the variables most correlated with the lasso residuals, $|D_2| - 1$ in total); (3) on D_2 the maximum of the de-biased lasso statistics over the screened variables in G is compared with a Gaussian multiplier bootstrap critical value. If no variable of G survives the screening, the test statistic is 0 and the hypothesis is not rejected.

Usage

```
ST(
  X.f,
  Y.f,
  sub.size,
  test.set,
  M = 500,
  alpha = 0.05,
  nodewise = c("ZnZ", "cv"),
  center = FALSE,
  legacy = FALSE,
  parallel = FALSE,
  ncores = getOption("mc.cores", 2L),
  scaled.lasso = c("equivariant", "legacy")
)
```

Arguments

<code>X.f</code>	Design matrix (n x p).
<code>Y.f</code>	Response variable (length n).
<code>sub.size</code>	The size of the screening sub-sample D_1 : a number of observations (<code>floor(sub.size)</code> is used), or a proportion of n if smaller than 1. Must leave at least 3 observations for screening and 10 for testing, for the cross-validation steps. Zhang and Cheng (2017) use $n/5$ to $n/3$.
<code>test.set</code>	The group of variables to be tested: column indices, or a logical vector of length p. Entries that are not column indices are ignored with a warning (SILM 1.0.0 ignored them silently).
<code>M</code>	The number of bootstrap replications (default 500).
<code>alpha</code>	The significance level (default 0.05).
<code>nodewise</code>	Tuning rule for the nodewise lasso that estimates Θ on D_2 : "ZnZ" (default) or "cv"; see SR() .

center	Logical. If TRUE, $X.f$ and $Y.f$ are centred separately within each sub-sample before screening and testing. The default FALSE uses the data as given and warns when they do not appear to be centred.
legacy	Logical. If TRUE, reproduce SILM 1.0.0 exactly (with the matching <code>nodewise</code> and <code>scaled.lasso = "legacy"</code> ; see "Nodewise tuning"), including three behaviours that were corrected in SILM 2.0.0: the decision of the studentized test was spelled "reject"; when <code>sub.size</code> left exactly $ D_2 - 1$ variables selected by the screening lasso, one additional variable was kept; and when no variable of <code>test.set</code> survived the screening, the statistics were $-\text{Inf}$ (now 0, with a warning).
parallel, ncores	Compute the nodewise regressions in parallel on <code>ncores</code> cores (forking; not on Windows). This does not change the results. With a BLAS library that uses OpenMP threads, forked workers can occasionally stall; setting the environment variable <code>OMP_NUM_THREADS=1</code> before starting R avoids this.
scaled.lasso	Numerical solver for the scaled lasso. The default "equivariant" normalizes the response before building the lasso path and stops at a relative noise-level change of $1\text{e-}8$, so estimates and intervals scale with the response units. "legacy" reproduces <code>scalreg</code> 1.0.1's absolute tolerance $1\text{e-}4$ and its sensitivity to response units; use it only to reproduce archived results. This setting can also change ordinary-scale results slightly because convergence is tighter.

Details

See [SR\(\)](#) for the model assumptions (no intercept, centred data). In ST the matrix Θ is always estimated by the nodewise lasso.

Value

A list of four elements, in this order (note that the names are duplicated, as in SILM 1.0.0, so use positions):

1. the non-studentized test statistic;
2. its decision, "reject" or "fail to reject";
3. the studentized test statistic;
4. its decision.

Nodewise tuning

SILM 1.0.0 obtained Θ from an internal function of the 'hdi' package. With hdi 0.1-6, current when SILM 1.0.0 was released (January 2019), the tuning parameter was the cross-validated lambda described in Zhang and Cheng (2017, Section 5) ("cv"). hdi 0.1-7 (March 2019) silently changed the default of that internal function to the Z&Z rule, so from then until SILM was archived (July 2026) SILM installations computed `nodewise = "ZnZ"` without saying so.

The default "ZnZ" of SILM 2.0.0 was chosen by a pre-registered study that compared the two rules on the simulation designs of Zhang and Cheng (2017) and on stress designs, under a decision rule fixed in advance (calibration first, then power): over 20 `Sim.CI()/Step()` settings, 8 `SR()`

settings and 3 `ST()` settings (200 replications each), "ZnZ" halved the mean calibration shortfall (coverage below 95%, error rates above 5%) from 0.100 to 0.053 (95% bootstrap interval of the difference: -0.051 to -0.044). The joint coverage of `Sim.CI()` for all p coefficients was higher or equal in all 20 settings (for example 0.77 with "cv" and 0.90 with "ZnZ" for signals (1.5, -1, 2) at $p = 500$), and exact support recovery by `SR()` improved; the price is about 12% wider intervals and 0.02-0.07 less power of `Step()` and `ST()` (see `validation/calibration/DEFAULTS.md` and `validation/calibration/defaults-results/REPORT.md` in the source repository). With the default, the results equal those of `SILM 1.0.0` as installed from 2019 to 2026 (with hdi 0.1-7 to 0.1-10; for `ST()` together with `legacy = TRUE`). Use `nodewise = "cv"` for the tuning of the paper; it reproduces `SILM 1.0.0` with hdi 0.1-6. Exact archived reproduction also requires `scaled.lasso = "legacy"`. Both `nodewise` settings reproduce the respective archived versions exactly (same numbers under the same random seed).

`lasso.proj()` and `boot.lasso.proj()` keep hdi's default for their `nodewise` residuals (do `ZnZ = FALSE`, the cross-validated `lambda`).

References

- Zhang, X., and Cheng, G. (2017) Simultaneous Inference for High-dimensional Linear Models, *Journal of the American Statistical Association*, 112, 757-768.
- Sun, T. and Zhang, C.-H. (2012). Scaled sparse linear regression. *Biometrika*, 99, 879-898.
- Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society, Series B*, 76, 217-242.

See Also

`SR()`, `Sim.CI()`, `Step()`

Examples

```
## The function is intended for large n and p.
## Use small p here for illustration purpose only.
set.seed(1)
n <- 100
p <- 10
s0 <- 3
set <- 1:s0
Sigma <- matrix(NA, p, p)
for (i in 1:p) Sigma[i,] <- 0.9^(abs(i-(1:p)))
X <- matrix(rnorm(n*p), n, p)
X <- t(t(chol(Sigma))%*%t(X))
beta <- rep(0,p)
beta[1:s0] <- runif(s0,0,2)
Y <- X%*%beta+rt(n,4)/sqrt(2)
test.set <- (s0+1):p
sub.size <- n*0.3
ST(X, Y, sub.size, test.set)
test.set <- s0:p
ST(X, Y, sub.size, test.set)
```

Step	<i>Stepdown Method for Multiple Testing</i>
------	---

Description

Tests $H_{0,j} : \beta_j = 0$ against two-sided alternatives for all $j = 1, \dots, p$ with the bootstrap-assisted stepdown procedure of Zhang and Cheng (2017, Section 3.3, in the two-sided form used in Section 5.4), which controls the familywise error rate asymptotically at level alpha. Critical values are recomputed with fresh bootstrap draws at every step. Both the non-studentized and the studentized statistics are used.

Usage

```
Step(
  X,
  Y,
  M = 500,
  alpha = 0.05,
  nodewise = c("ZnZ", "cv"),
  center = FALSE,
  Theta = NULL,
  parallel = FALSE,
  ncores = getOption("mc.cores", 2L),
  scaled.lasso = c("equivariant", "legacy")
)
```

Arguments

X	Design matrix (n x p).
Y	Response variable (a numeric vector of length n, or an n x 1 matrix).
M	The number of bootstrap replications per step (default 500).
alpha	The significance level (familywise error rate; default 0.05).
nodewise	Tuning rule for the nodewise lasso that estimates Θ (the inverse of the Gram matrix) when $p > \text{floor}(n/2)$: "ZnZ" (default) refines the lambda minimising 10-fold cross-validation error pooled over all nodewise regressions with the rule of Zhang and Zhang (2014); "cv" uses that cross-validated lambda itself, as described in Zhang and Cheng (2017, Section 5). See the section "Nodewise tuning" below.
center	Logical. If TRUE, the columns of X and Y are centred before the analysis. The default FALSE uses the data as given (as SILM 1.0.0 did) and warns when they do not appear to be centred.
Theta	Optional precomputed $\hat{\Theta}$ (a p x p matrix, e.g. from <code>Theta.hat()</code>) to avoid recomputing it. Note that supplying it skips the random draw of the nodewise cross-validation folds, and that nodewise is then not used: the tuning rule is the one Theta was computed with. <code>Theta.hat()</code> records it (attribute "nodewise"), and a warning is given when nodewise is supplied explicitly and differs.

`parallel, ncores`

Compute the nodewise regressions in parallel on `ncores` cores (forking; not on Windows). This does not change the results. With a BLAS library that uses OpenMP threads, forked workers can occasionally stall; setting the environment variable `OMP_NUM_THREADS=1` before starting R avoids this.

`scaled.lasso`

Numerical solver for the scaled lasso. The default "equivariant" normalizes the response before building the lasso path and stops at a relative noise-level change of $1e-8$, so estimates and intervals scale with the response units. "legacy" reproduces `scalreg 1.0.1`'s absolute tolerance $1e-4$ and its sensitivity to response units; use it only to reproduce archived results. This setting can also change ordinary-scale results slightly because convergence is tighter.

Details

See `SR()` for the model assumptions (no intercept, centred data) and the estimation of Θ .

Value

A list with two integer vectors, "non-studentized test" and "studentized test": the indices of the rejected hypotheses.

Finite-sample behaviour

In the defaults study of the source repository (validation/calibration/defaults-results/REPORT.md; $n = 100$, the paper's designs, 200 replications each) the familywise error rate of both tests was at most 0.05 in every setting except the exchangeable designs with $s_0 = 15$, where it was 0.12-0.15 (0.21-0.24 with `nodewise = "cv"`). See `Sim.CI()` for the bias of the de-biased lasso behind such deviations.

Nodewise tuning

SILM 1.0.0 obtained Θ from an internal function of the 'hdi' package. With hdi 0.1-6, current when SILM 1.0.0 was released (January 2019), the tuning parameter was the cross-validated lambda described in Zhang and Cheng (2017, Section 5) ("cv"). hdi 0.1-7 (March 2019) silently changed the default of that internal function to the Z&Z rule, so from then until SILM was archived (July 2026) SILM installations computed `nodewise = "ZnZ"` without saying so.

The default "ZnZ" of SILM 2.0.0 was chosen by a pre-registered study that compared the two rules on the simulation designs of Zhang and Cheng (2017) and on stress designs, under a decision rule fixed in advance (calibration first, then power): over 20 `Sim.CI()/Step()` settings, 8 `SR()` settings and 3 `ST()` settings (200 replications each), "ZnZ" halved the mean calibration shortfall (coverage below 95%, error rates above 5%) from 0.100 to 0.053 (95% bootstrap interval of the difference: -0.051 to -0.044). The joint coverage of `Sim.CI()` for all p coefficients was higher or equal in all 20 settings (for example 0.77 with "cv" and 0.90 with "ZnZ" for signals (1.5, -1, 2) at $p = 500$), and exact support recovery by `SR()` improved; the price is about 12% wider intervals and 0.02-0.07 less power of `Step()` and `ST()` (see validation/calibration/DEFAULTS.md and validation/calibration/defaults-results/REPORT.md in the source repository). With the default, the results equal those of SILM 1.0.0 as installed from 2019 to 2026 (with hdi 0.1-7 to 0.1-10; for `ST()` together with `legacy = TRUE`). Use `nodewise = "cv"` for the tuning of the paper; it reproduces SILM 1.0.0 with hdi 0.1-6. Exact archived reproduction also requires `scaled.lasso =`

"legacy". Both nodewise settings reproduce the respective archived versions exactly (same numbers under the same random seed).

[lasso.proj\(\)](#) and [boot.lasso.proj\(\)](#) keep hdi's default for their nodewise residuals (do.ZnZ = FALSE, the cross-validated lambda).

References

Zhang, X., and Cheng, G. (2017) Simultaneous Inference for High-dimensional Linear Models, *Journal of the American Statistical Association*, 112, 757-768.

Sun, T. and Zhang, C.-H. (2012). Scaled sparse linear regression. *Biometrika*, 99, 879-898.

Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society, Series B*, 76, 217-242.

See Also

[SR\(\)](#), [Sim.CI\(\)](#), [ST\(\)](#)

Examples

```
## The function is intended for large n and p.
## Use small p here for illustration purpose only.
set.seed(1)
n <- 100
p <- 10
s0 <- 3
set <- 1:s0
Sigma <- matrix(NA, p, p)
for (i in 1:p) Sigma[i,] <- 0.9^(abs(i-(1:p)))
X <- matrix(rnorm(n*p), n, p)
X <- t(t(chol(Sigma))%*%t(X))
beta <- rep(0,p)
beta[1:s0] <- runif(s0,1,2)
Y <- X%*%beta+rt(n,4)/sqrt(2)
Step(X, Y, M=500, alpha=0.05)
```

Theta.hat

Estimate of the inverse Gram matrix used by SR, Sim.CI and Step

Description

Returns the matrix $\hat{\Theta}$ that [SR\(\)](#), [Sim.CI\(\)](#) and [Step\(\)](#) use to de-bias the scaled lasso: the inverse of the Gram matrix $X^T X/n$ when $p \leq \text{floor}(n/2)$, and the nodewise lasso estimate otherwise (Zhang and Cheng, 2017). Because $\hat{\Theta}$ depends only on the design, it can be computed once and passed to several calls through their Theta argument.

Usage

```
Theta.hat(
  X,
  nodewise = c("ZnZ", "cv"),
  center = FALSE,
  parallel = FALSE,
  ncores = getOption("mc.cores", 2L)
)
```

Arguments

<code>X</code>	Design matrix (n x p).
<code>nodewise</code>	Tuning rule for the nodewise lasso that estimates Θ (the inverse of the Gram matrix) when $p > \text{floor}(n/2)$: "ZnZ" (default) refines the lambda minimising 10-fold cross-validation error pooled over all nodewise regressions with the rule of Zhang and Zhang (2014); "cv" uses that cross-validated lambda itself, as described in Zhang and Cheng (2017, Section 5). See the section "Nodewise tuning" below.
<code>center</code>	Logical. If TRUE, the columns of X are centred before Θ is estimated (use the same value as in the calls that use the result). No check for centring is made.
<code>parallel, ncores</code>	Compute the nodewise regressions in parallel on ncores cores (forking; not on Windows). This does not change the results. With a BLAS library that uses OpenMP threads, forked workers can occasionally stall; setting the environment variable <code>OMP_NUM_THREADS=1</code> before starting R avoids this.

Details

The nodewise lasso draws the cross-validation folds from the random number generator. Supplying Theta therefore skips that draw, so the bootstrap draws of a subsequent call differ from those of a call that computes Θ itself under the same seed.

Value

A $p \times p$ matrix with attributes `method` ("inverse-gram" or "nodewise"), `center`, and, for the nodewise lasso, `lambda` (the tuning parameter used) and `nodewise` (the tuning rule). [SR\(\)](#), [Sim.CI\(\)](#) and [Step\(\)](#) warn when they are given this matrix together with a different `nodewise`.

Nodewise tuning

SILM 1.0.0 obtained Θ from an internal function of the 'hdi' package. With hdi 0.1-6, current when SILM 1.0.0 was released (January 2019), the tuning parameter was the cross-validated lambda described in Zhang and Cheng (2017, Section 5) ("cv"). hdi 0.1-7 (March 2019) silently changed the default of that internal function to the Z&Z rule, so from then until SILM was archived (July 2026) SILM installations computed `nodewise = "ZnZ"` without saying so.

The default "ZnZ" of SILM 2.0.0 was chosen by a pre-registered study that compared the two rules on the simulation designs of Zhang and Cheng (2017) and on stress designs, under a decision rule fixed in advance (calibration first, then power): over 20 [Sim.CI\(\)](#)/[Step\(\)](#) settings, 8 [SR\(\)](#)

settings and 3 ST() settings (200 replications each), "ZnZ" halved the mean calibration shortfall (coverage below 95%, error rates above 5%) from 0.100 to 0.053 (95% bootstrap interval of the difference: -0.051 to -0.044). The joint coverage of Sim.CI() for all p coefficients was higher or equal in all 20 settings (for example 0.77 with "cv" and 0.90 with "ZnZ" for signals (1.5, -1, 2) at p = 500), and exact support recovery by SR() improved; the price is about 12% wider intervals and 0.02-0.07 less power of Step() and ST() (see validation/calibration/DEFAULTS.md and validation/calibration/defaults-results/REPORT.md in the source repository). With the default, the results equal those of SILM 1.0.0 as installed from 2019 to 2026 (with hdi 0.1-7 to 0.1-10; for ST() together with legacy = TRUE). Use nodewise = "cv" for the tuning of the paper; it reproduces SILM 1.0.0 with hdi 0.1-6. Exact archived reproduction also requires scaled.lasso = "legacy". Both nodewise settings reproduce the respective archived versions exactly (same numbers under the same random seed).

`lasso.proj()` and `boot.lasso.proj()` keep hdi's default for their nodewise residuals (do.ZnZ = FALSE, the cross-validated lambda).

References

- Zhang, X., and Cheng, G. (2017) Simultaneous Inference for High-dimensional Linear Models, *Journal of the American Statistical Association*, 112, 757-768.
- Sun, T. and Zhang, C.-H. (2012). Scaled sparse linear regression. *Biometrika*, 99, 879-898.
- Zhang, C.-H. and Zhang, S. S. (2014). Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society, Series B*, 76, 217-242.

Examples

```
set.seed(1)
X <- matrix(rnorm(40 * 60), 40, 60)
Y <- X[, 1] + rnorm(40)
Theta <- Theta.hat(X)
attr(Theta, "method")
SR(X, Y, Theta = Theta)
```

Index

`boot.lasso.proj`, 2
`boot.lasso.proj()`, 4, 7–9, 12–14, 16, 19, 22, 25, 27

`confint()`, 3, 10
`confint.silm_proj`, 7
`confint.silm_proj()`, 7, 9, 13

`groupTest`, 9
`groupTest()`, 4, 6–8

`lasso.proj`, 10
`lasso.proj()`, 7, 8, 14, 16, 19, 22, 25, 27

`print.silm_proj`, 13

`Sim.CI`, 14
`Sim.CI()`, 6, 7, 19, 22, 24–26
`SR`, 17
`SR()`, 15, 17, 20–22, 24–26
`ST`, 20
`ST()`, 15, 17, 19, 25
`stats::p.adjust()`, 3, 11
`Step`, 23
`Step()`, 15, 17, 19, 22, 25, 26

`Theta.hat`, 25
`Theta.hat()`, 15, 18, 19, 23