# Package: susieR (via r-universe)

**Encoding** UTF-8

**Type** Package

**Title** Sum of Single Effects Linear Regression

**Description** Implements methods for variable selection in linear
regression based on the ``Sum of Single Effects'' (SuSiE) model,
as described in Wang et al (2020) <DOI:10.1101/501114> and Zou
et al (2021) <DOI:10.1101/2021.11.03.467167>. These methods
provide simple summaries, called ``Credible Sets'', for
accurately quantifying uncertainty in which variables should be
selected. The methods are motivated by genetic fine-mapping
applications, and are particularly well-suited to settings
where variables are highly correlated and detectable effects
are sparse. The fitting algorithm, a Bayesian analogue of
stepwise selection methods called ``Iterative Bayesian Stepwise
Selection'' (IBSS), is simple and fast, allowing the SuSiE model
be fit to large data sets (thousands of samples and hundreds of
thousands of variables).

**Date** 2024-08-01

**Version** 0.12.42

**URL** https://github.com/stephenslab/susieR

**BugReports** https://github.com/stephenslab/susieR/issues

**License** BSD_3_clause + file LICENSE

**Depends** R (>= 3.0.0)

**Imports** methods, graphics, grDevices, stats, Matrix, matrixStats,
mixsqp, reshape, crayon, ggplot2

**Suggests** curl, testthat, microbenchmark, knitr, rmarkdown, Rfast,
cowplot

**LazyData** yes

**LazyDataCompression** xz

**NeedsCompilation** no

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**Repository** https://stephenslab.r-universe.dev

**RemoteUrl** https://github.com/stephenslab/susier

**RemoteRef** HEAD

**RemoteSha** ced6a9c83a45f792d4d2ef2a9ae0846e164bf92c

# Contents

---

| coef.susie | *Extract regression coefficients from susie fit* |
|------------|--------------------------------------------------|

---

## Description

Extract regression coefficients from susie fit

## Usage

```
## S3 method for class 'susie'
coef(object, ...)
```

## Arguments

object        A susie fit.

...           Additional arguments passed to the generic coef method.

## Value

A p+1 vector, the first element being an intercept, and the remaining p elements being estimated regression coefficients.

---

compute_ss                    *Compute sufficient statistics for input to* susie_suff_stat

---

## Description

This is a synonym for compute_suff_stat included for historical reasons (deprecated).

## Usage

```
compute_ss(X, y, standardize = FALSE)
```

## Arguments

X            An n by p matrix of covariates.

y            An n vector.

standardize  Logical flag indicating whether to standardize columns of X to unit variance prior to computing summary data.

## Value

A list of sufficient statistics (X'X, X'y, y'y and n)

## Examples

```
data(N2finemapping)
ss = compute_ss(N2finemapping$X, N2finemapping$Y[,1])
```

---

compute_suff_stat                    *Compute sufficient statistics for input to* `susie_suff_stat`

---

### Description

Computes the sufficient statistics $X'X, X'y, y'y$ and $n$ after centering (and possibly standardizing) the columns of $X$ and centering $y$ to have mean zero. We also store the column means of $X$ and mean of $y$.

### Usage

```
compute_suff_stat(X, y, standardize = FALSE)
```

### Arguments

| | |
|---|---|
| X | An n by p matrix of covariates. |
| y | An n vector. |
| standardize | Logical flag indicating whether to standardize columns of X to unit variance prior to computing summary data |

### Value

A list of sufficient statistics (`XtX`, `Xty`, `yty`, `n`) and `X_colmeans`, `y_mean`.

### Examples

```
data(N2finemapping)
ss = compute_suff_stat(N2finemapping$X, N2finemapping$Y[,1])
```

---

estimate_s_rss                    *Estimate s in* `susie_rss` *Model Using Regularized LD*

---

### Description

The estimated s gives information about the consistency between the z scores and LD matrix. A larger $s$ means there is a strong inconsistency between z scores and LD matrix. The "null-mle" method obtains mle of $s$ under $z|R \ N(0, (1-s)R + sI), 0 < s < 1$. The "null-partialmle" method obtains mle of $s$ under $U^T z|R \ N(0, sI)$, in which $U$ is a matrix containing the of eigenvectors that span the null space of R; that is, the eigenvectors corresponding to zero eigenvalues of R. The estimated $s$ from "null-partialmle" could be greater than 1. The "null-pseudomle" method obtains mle of $s$ under pseudolikelihood $L(s) = \prod_{j=1}^{p} p(z_j|z_{-j}, s, R), 0 < s < 1$.

### Usage

```
estimate_s_rss(z, R, n, r_tol = 1e-08, method = "null-mle")
```

## Arguments

| | |
|---|---|
| z | A p-vector of z scores. |
| R | A p by p symmetric, positive semidefinite correlation matrix. |
| n | The sample size. (Optional, but highly recommended.) |
| r_tol | Tolerance level for eigenvalue check of positive semidefinite matrix of R. |
| method | a string specifies the method to estimate $s$. |

## Value

A number between 0 and 1.

## Examples

```
set.seed(1)
n = 500
p = 1000
beta = rep(0,p)
beta[1:4] = 0.01
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
input_ss = compute_suff_stat(X,y,standardize = TRUE)
ss = univariate_regression(X,y)
R = cor(X)
attr(R,"eigen") = eigen(R,symmetric = TRUE)
zhat = with(ss,betahat/sebetahat)

# Estimate s using the unadjusted z-scores.
s0 = estimate_s_rss(zhat,R)

# Estimate s using the adjusted z-scores.
s1 = estimate_s_rss(zhat,R,n)
```

---

FinemappingConvergence

*Simulated Fine-mapping Data with Convergence Problem.*

---

## Description

Data simulated using real genotypes from 50,000 individuals and 200 SNPs. Two of the SNPs have non-zero effects on the multivariate response. The response data are generated under a linear regression model. The simulated response and the columns of the genotype matrix are centered.

## Format

FinemappingConvergence is a list with the following elements:

**XtX** Summary statistics computed using the centered and scaled genotype matrix.

**Xty** Summary statistics computed using the centered and scaled genotype data, and the centered simulated response.

**yty** yty is computed using the centered simulated response.

**n** The sample size (50,000).

**true_coef** The coefficients used to simulate the responses.

**z** z-scores from a simple (single-SNP) linear regression.

## See Also

A similar data set with more SNPs is used in the "Refine SuSiE model" vignette.

## Examples

```
data(FinemappingConvergence)
```

---

| get_cs_correlation | *Get Correlations Between CSs, using Variable with Maximum PIP From Each CS* |
|---|---|

---

## Description

This function evaluates the correlation between single effect CSs. It is not part of the SuSiE inference. Rather, it is designed as a diagnostic tool to assess how correlated the reported CS are.

## Usage

```
get_cs_correlation(model, X = NULL, Xcorr = NULL, max = FALSE)
```

## Arguments

| | |
|---|---|
| model | A SuSiE fit, typically an output from [susie](#) or one of its variants. |
| X | n by p matrix of values of the p variables (covariates) in n samples. When provided, correlation between variables will be computed and used to remove CSs whose minimum correlation among variables is smaller than min_abs_corr. |
| Xcorr | p by p matrix of correlations between variables (covariates). When provided, it will be used to remove CSs whose minimum correlation among variables is smaller than min_abs_corr. |
| max | When max = FAFLSE, return a matrix of CS correlations. When max = TRUE, return only the maximum absolute correlation among all pairs of correlations. |

## Value

A matrix of correlations between CSs, or the maximum absolute correlation when max = TRUE.

---

| | |
|---|---|
| kriging_rss | *Compute Distribution of z-scores of Variant j Given Other z-scores, and Detect Possible Allele Switch Issue* |

---

## Description

Under the null, the rss model with regularized LD matrix is $z|R, s \ N(0, (1-s)R + sI))$. We use a mixture of normals to model the conditional distribution of z_j given other z scores, $z_j|z_{-j}, R, s \ \sum_{k=1}^{K} \pi_k N(-\Omega_{j,-j}z_{-j}/\Omega_{j}$ $\Omega = ((1-s)R + sI)^{-1}$, $\sigma_1, ..., \sigma_k$ is a grid of fixed positive numbers. We estimate the mixture weights $\pi$ We detect the possible allele switch issue using likelihood ratio for each variant.

## Usage

```
kriging_rss(
  z,
  R,
  n,
  r_tol = 1e-08,
  s = estimate_s_rss(z, R, n, r_tol, method = "null-mle")
)
```

## Arguments

| | |
|---|---|
| z | A p-vector of z scores. |
| R | A p by p symmetric, positive semidefinite correlation matrix. |
| n | The sample size. (Optional, but highly recommended.) |
| r_tol | Tolerance level for eigenvalue check of positive semidefinite matrix of R. |
| s | an estimated s from estimate_s_rss |

## Value

a list containing a ggplot2 plot object and a table. The plot compares observed z score vs the expected value. The possible allele switched variants are labeled as red points (log LR > 2 and abs(z) > 2). The table summarizes the conditional distribution for each variant and the likelihood ratio test. The table has the following columns: the observed z scores, the conditional expectation, the conditional variance, the standardized differences between the observed z score and expected value, the log likelihood ratio statistics.

## Examples

```
# See also the vignette, "Diagnostic for fine-mapping with summary
# statistics."
set.seed(1)
n = 500
p = 1000
beta = rep(0,p)
beta[1:4] = 0.01
```

```
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
ss = univariate_regression(X,y)
R = cor(X)
attr(R,"eigen") = eigen(R,symmetric = TRUE)
zhat = with(ss,betahat/sebetahat)
cond_dist = kriging_rss(zhat,R,n = n)
cond_dist$plot
```

---

N2finemapping                  *Simulated Fine-mapping Data with Two Effect Variables*

---

### Description

This data set contains a genotype matrix for 574 individuals and 1,002 variables. The variables are genotypes after centering and scaling, and therefore retain the correlation structure of the original genotype data. Two of the variables have non-zero effects on the multivariate response. The response data are generated under a multivariate linear regression model. See Wang *et al* (2020) for details.

### Format

N2finemapping is a list with the following elements:

**X** Centered and scaled genotype data.

**chrom** Chromomsome of the original data, in hg38 coordinates.

**pos** Chromomosomal position of the original data, in hg38 coordinates. The information can be used to compare impact of using other genotype references of the same variables in susie_rss application.

**true_coef** Simulated effect sizes.

**residual_variance** Simulated residual covariance matrix.

**Y** Simulated multivariate response.

**allele_freq** Allele frequencies based on the original genotype data.

**V** Suggested prior covariance matrix for effect sizes of the two non-zero effect variables.

### References

G. Wang, A. Sarkar, P. Carbonetto and M. Stephens (2020). A simple new approach to variable selection in regression, with application to genetic fine-mapping. *Journal of the Royal Statistical Society, Series B* doi:10.1101/501114.

### Examples

```
data(N2finemapping)
```

---

| N3finemapping | *Simulated Fine-mapping Data with Three Effect Variables.* |
|---|---|

---

### Description

The data-set contains a matrix of 574 individuals and 1,001 variables. These variables are real-world genotypes centered and scaled, and therefore retains the correlation structure of variables in the original genotype data. 3 out of the variables have non-zero effects. The response data is generated under a multivariate linear regression model. See Wang *et al* (2020) for more details.

### Format

N3finemapping is a list with the following elements:

**X** N by P variable matrix of centered and scaled genotype data.

**chrom** Chromomsome of the original data, in hg38 coordinate.

**pos** Chromomosomal positoin of the original data, in hg38 coordinate. The information can be used to compare impact of using other genotype references of the same variables in susie_rss application.

**true_coef** The simulated effect sizes.

**residual_variance** The simulated residual covariance matrix.

**Y** The simulated response variables.

**allele_freq** Allele frequency of the original genotype data.

**V** Prior covariance matrix for effect size of the three non-zero effect variables.

### References

G. Wang, A. Sarkar, P. Carbonetto and M. Stephens (2020). A simple new approach to variable selection in regression, with application to genetic fine-mapping. *Journal of the Royal Statistical Society, Series B* doi:10.1101/501114.

### Examples

```
data(N3finemapping)
```

---

| `predict.susie` | *Predict outcomes or extract coefficients from susie fit.* |

---

### Description

Predict outcomes or extract coefficients from susie fit.

### Usage

```
## S3 method for class 'susie'
predict(object, newx = NULL, type = c("response", "coefficients"), ...)
```

### Arguments

| | |
|---|---|
| object | A susie fit. |
| newx | A new value for X at which to do predictions. |
| type | The type of output. For type = "response", predicted or fitted outcomes are returned; for type = "coefficients", the estimated coefficients are returned. |
| ... | Other arguments used by generic predict function. These extra arguments are not used here. |

### Value

For type = "response", predicted or fitted outcomes are returned; for type = "coefficients", the estimated coefficients are returned. If the susie fit has intercept = NA (which is common when using susie_suff_stat) then predictions are computed using an intercept of 0, and a warning is emitted.

---

| `summary.susie` | *Summarize Susie Fit.* |

---

### Description

summary method for the "susie" class.

### Usage

```
## S3 method for class 'susie'
summary(object, ...)

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

## Arguments

| | |
|---|---|
| `object` | A susie fit. |
| `...` | Additional arguments passed to the generic `summary` or `print.summary` method. |
| `x` | A susie summary. |

## Value

`summary.susie` returns a list containing a data frame of variables and a data frame of credible sets.

---

SummaryConsistency | *Simulated Fine-mapping Data with LD matrix From Reference Panel.*

---

## Description

Data simulated using real genotypes from 10,000 individuals and 200 SNPs. One SNP have non-zero effect on the multivariate response. The response data are generated under a linear regression model. There is also one SNP with flipped allele between summary statistics and the reference panel.

## Format

SummaryConsistency is a list with the following elements:

**z** z-scores computed by fitting univariate simple regression variable-by-variable.

**ldref** LD matrix estimated from the reference panel.

**flip_id** The index of the SNP with the flipped allele.

**signal_id** The index of the SNP with the non-zero effect.

## See Also

A similar data set with more samples is used in the "Diagnostic for fine-mapping with summary statistics" vignette.

## Examples

```
data(SummaryConsistency)
```

---

susie                              *Sum of Single Effects (SuSiE) Regression*

---

### Description

Performs a sparse Bayesian multiple linear regression of y on X, using the "Sum of Single Effects" model from Wang et al (2020). In brief, this function fits the regression model $y = \mu + Xb + e$, where elements of $e$ are *i.i.d.* normal with zero mean and variance `residual_variance`, $\mu$ is an intercept term and $b$ is a vector of length p representing the effects to be estimated. The "susie assumption" is that $b = \sum_{l=1}^{L} b_l$ where each $b_l$ is a vector of length p with exactly one non-zero element. The prior on the non-zero element is normal with zero mean and variance `var(y) * scaled_prior_variance`. The value of L is fixed, and should be chosen to provide a reasonable upper bound on the number of non-zero effects to be detected. Typically, the hyperparameters `residual_variance` and `scaled_prior_variance` will be estimated during model fitting, although they can also be fixed as specified by the user. See functions `susie_get_cs` and other functions of form `susie_get_*` to extract the most commonly-used results from a susie fit.

### Usage

```
susie(
  X,
  y,
  L = min(10, ncol(X)),
  scaled_prior_variance = 0.2,
  residual_variance = NULL,
  prior_weights = NULL,
  null_weight = 0,
  standardize = TRUE,
  intercept = TRUE,
  estimate_residual_variance = TRUE,
  estimate_prior_variance = TRUE,
  estimate_prior_method = c("optim", "EM", "simple"),
  check_null_threshold = 0,
  prior_tol = 1e-09,
  residual_variance_upperbound = Inf,
  s_init = NULL,
  coverage = 0.95,
  min_abs_corr = 0.5,
  compute_univariate_zscore = FALSE,
  na.rm = FALSE,
  max_iter = 100,
  tol = 0.001,
  verbose = FALSE,
  track_fit = FALSE,
  residual_variance_lowerbound = var(drop(y))/10000,
  refine = FALSE,
  n_purity = 100
```

```
)
susie_suff_stat(
  XtX,
  Xty,
  yty,
  n,
  X_colmeans = NA,
  y_mean = NA,
  maf = NULL,
  maf_thresh = 0,
  L = 10,
  scaled_prior_variance = 0.2,
  residual_variance = NULL,
  estimate_residual_variance = TRUE,
  estimate_prior_variance = TRUE,
  estimate_prior_method = c("optim", "EM", "simple"),
  check_null_threshold = 0,
  prior_tol = 1e-09,
  r_tol = 1e-08,
  prior_weights = NULL,
  null_weight = 0,
  standardize = TRUE,
  max_iter = 100,
  s_init = NULL,
  coverage = 0.95,
  min_abs_corr = 0.5,
  tol = 0.001,
  verbose = FALSE,
  track_fit = FALSE,
  check_input = FALSE,
  refine = FALSE,
  check_prior = FALSE,
  n_purity = 100,
  ...
)
```

**Arguments**

| | |
|---|---|
| X | An n by p matrix of covariates. |
| y | The observed responses, a vector of length n. |
| L | Maximum number of non-zero effects in the susie regression model. If L is larger than the number of covariates, p, L is set to p. |
| scaled_prior_variance | |
| | The prior variance, divided by var(y) (or by (1/(n-1))yty for susie_suff_stat); that is, the prior variance of each non-zero element of b is var(y) * scaled_prior_variance. The value provided should be either a scalar or a vector of length L. If estimate_prior_variance = TRUE, this provides initial estimates of the prior variances. |

residual_variance

>    Variance of the residual. If `estimate_residual_variance = TRUE`, this value
>    provides the initial estimate of the residual variance. By default, it is set to
>    `var(y)` in `susie` and `(1/(n-1))yty` in `susie_suff_stat`.

prior_weights    A vector of length p, in which each entry gives the prior probability that corre-
                 sponding column of X has a nonzero effect on the outcome, y.

null_weight      Prior probability of no effect (a number between 0 and 1, and cannot be exactly
                 1).

standardize      If `standardize = TRUE`, standardize the columns of X to unit variance prior to
                 fitting (or equivalently standardize XtX and Xty to have the same effect). Note
                 that `scaled_prior_variance` specifies the prior on the coefficients of X *after*
                 standardization (if it is performed). If you do not standardize, you may need
                 to think more carefully about specifying `scaled_prior_variance`. Whatever
                 your choice, the coefficients returned by `coef` are given for X on the original
                 input scale. Any column of X that has zero variance is not standardized.

intercept        If `intercept = TRUE`, the intercept is fitted; it `intercept = FALSE`, the intercept
                 is set to zero. Setting `intercept = FALSE` is generally not recommended.

estimate_residual_variance

>    If `estimate_residual_variance = TRUE`, the residual variance is estimated,
>    using `residual_variance` as an initial value. If `estimate_residual_variance`
>    `= FALSE`, the residual variance is fixed to the value supplied by `residual_variance`.

estimate_prior_variance

>    If `estimate_prior_variance = TRUE`, the prior variance is estimated (this is a
>    separate parameter for each of the L effects). If provided, `scaled_prior_variance`
>    is then used as an initial value for the optimization. When `estimate_prior_variance`
>    `= FALSE`, the prior variance for each of the L effects is determined by the value
>    supplied to `scaled_prior_variance`.

estimate_prior_method

>    The method used for estimating prior variance. When `estimate_prior_method`
>    `= "simple"` is used, the likelihood at the specified prior variance is compared to
>    the likelihood at a variance of zero, and the setting with the larger likelihood is
>    retained.

check_null_threshold

>    When the prior variance is estimated, compare the estimate with the null, and set
>    the prior variance to zero unless the log-likelihood using the estimate is larger
>    by this threshold amount. For example, if you set `check_null_threshold =`
>    `0.1`, this will "nudge" the estimate towards zero when the difference in log-
>    likelihoods is small. A note of caution that setting this to a value greater than
>    zero may lead the IBSS fitting procedure to occasionally decrease the ELBO.

prior_tol        When the prior variance is estimated, compare the estimated value to `prior_tol`
                 at the end of the computation, and exclude a single effect from PIP computation
                 if the estimated prior variance is smaller than this tolerance value.

residual_variance_upperbound

>    Upper limit on the estimated residual variance. It is only relevant when `estimate_residual_variance`
>    `= TRUE`.

s_init           A previous susie fit with which to initialize.

| | |
|---|---|
| coverage | A number between 0 and 1 specifying the "coverage" of the estimated confidence sets. |
| min_abs_corr | Minimum absolute correlation allowed in a credible set. The default, 0.5, corresponds to a squared correlation of 0.25, which is a commonly used threshold for genotype data in genetic studies. |
| compute_univariate_zscore | |
| | If compute_univariate_zscore = TRUE, the univariate regression z-scores are outputted for each variable. |
| na.rm | Drop any missing values in y from both X and y. |
| max_iter | Maximum number of IBSS iterations to perform. |
| tol | A small, non-negative number specifying the convergence tolerance for the IBSS fitting procedure. The fitting procedure will halt when the difference in the variational lower bound, or "ELBO" (the objective function to be maximized), is less than tol. |
| verbose | If verbose = TRUE, the algorithm's progress, and a summary of the optimization settings, are printed to the console. |
| track_fit | If track_fit = TRUE, trace is also returned containing detailed information about the estimates at each iteration of the IBSS fitting procedure. |
| residual_variance_lowerbound | |
| | Lower limit on the estimated residual variance. It is only relevant when estimate_residual_variance = TRUE. |
| refine | If refine = TRUE, then an additional iterative refinement procedure is used, after the IBSS algorithm, to check and escape from local optima (see details). |
| n_purity | Passed as argument n_purity to [susie_get_cs](susie_get_cs). |
| XtX | A p by p matrix $X'X$ in which the columns of X are centered to have mean zero. |
| Xty | A p-vector $X'y$ in which y and the columns of X are centered to have mean zero. |
| yty | A scalar $y'y$ in which y is centered to have mean zero. |
| n | The sample size. |
| X_colmeans | A p-vector of column means of X. If both X_colmeans and y_mean are provided, the intercept is estimated; otherwise, the intercept is NA. |
| y_mean | A scalar containing the mean of y. If both X_colmeans and y_mean are provided, the intercept is estimated; otherwise, the intercept is NA. |
| maf | Minor allele frequency; to be used along with maf_thresh to filter input summary statistics. |
| maf_thresh | Variants having a minor allele frequency smaller than this threshold are not used. |
| r_tol | Tolerance level for eigenvalue check of positive semidefinite matrix of R. |
| check_input | If check_input = TRUE, susie_suff_stat performs additional checks on XtX and Xty. The checks are: (1) check that XtX is positive semidefinite; (2) check that Xty is in the space spanned by the non-zero eigenvectors of XtX. |
| check_prior | If check_prior = TRUE, it checks if the estimated prior variance becomes unreasonably large (comparing with 10 * max(abs(z))^2). |
| ... | Additional arguments to provide backward compatibility with earlier versions of susie_suff_stat. |

**Details**

The function `susie` implements the IBSS algorithm from Wang et al (2020). The option `refine = TRUE` implements an additional step to help reduce problems caused by convergence of the IBSS algorithm to poor local optima (which is rare in our experience, but can provide misleading results when it occurs). The refinement step incurs additional computational expense that increases with the number of CSs found in the initial run.

The function `susie_suff_stat` implements essentially the same algorithms, but using sufficient statistics. (The statistics are sufficient for the regression coefficients $b$, but not for the intercept $\mu$; see below for how the intercept is treated.) If the sufficient statistics are computed correctly then the results from `susie_suff_stat` should be the same as (or very similar to) `susie`, although runtimes will differ as discussed below. The sufficient statistics are the sample size n, and then the p by p matrix $X'X$, the p-vector $X'y$, and the sum of squared y values $y'y$, all computed after centering the columns of $X$ and the vector $y$ to have mean 0; these can be computed using `compute_suff_stat`.

The handling of the intercept term in `susie_suff_stat` needs some additional explanation. Computing the summary data after centering `X` and `y` effectively ensures that the resulting posterior quantities for $b$ allow for an intercept in the model; however, the actual value of the intercept cannot be estimated from these centered data. To estimate the intercept term the user must also provide the column means of $X$ and the mean of $y$ (`X_colmeans` and `y_mean`). If these are not provided, they are treated as `NA`, which results in the intercept being `NA`. If for some reason you prefer to have the intercept be 0 instead of `NA` then set `X_colmeans = 0, y_mean = 0`.

For completeness, we note that if `susie_suff_stat` is run on $X'X, X'y, y'y$ computed *without* centering $X$ and $y$, and with `X_colmeans = 0, y_mean = 0`, this is equivalent to `susie` applied to $X, y$ with `intercept = FALSE` (although results may differ due to different initializations of `residual_variance` and `scaled_prior_variance`). However, this usage is not recommended for for most situations.

The computational complexity of `susie` is $O(npL)$ per iteration, whereas `susie_suff_stat` is $O(p^2L)$ per iteration (not including the cost of computing the sufficient statistics, which is dominated by the $O(np^2)$ cost of computing $X'X$). Because of the cost of computing $X'X$, `susie` will usually be faster. However, if $n >> p$, and/or if $X'X$ is already computed, then `susie_suff_stat` may be faster.

**Value**

A `"susie"` object with some or all of the following elements:

| | |
|---|---|
| alpha | An L by p matrix of posterior inclusion probabilites. |
| mu | An L by p matrix of posterior means, conditional on inclusion. |
| mu2 | An L by p matrix of posterior second moments, conditional on inclusion. |
| Xr | A vector of length n, equal to X %*% colSums(alpha * mu). |
| lbf | log-Bayes Factor for each single effect. |
| lbf_variable | log-Bayes Factor for each variable and single effect. |
| intercept | Intercept (fixed or estimated). |
| sigma2 | Residual variance (fixed or estimated). |
| V | Prior variance of the non-zero elements of b, equal to scaled_prior_variance * var(y). |

| | |
|---|---|
| elbo | The value of the variational lower bound, or "ELBO" (objective function to be maximized), achieved at each iteration of the IBSS fitting procedure. |
| fitted | Vector of length n containing the fitted values of the outcome. |
| sets | Credible sets estimated from model fit; see `susie_get_cs` for details. |
| pip | A vector of length p giving the (marginal) posterior inclusion probabilities for all p covariates. |
| z | A vector of univariate z-scores. |
| niter | Number of IBSS iterations that were performed. |
| converged | TRUE or FALSE indicating whether the IBSS converged to a solution within the chosen tolerance level. |

`susie_suff_stat` returns also outputs:

| | |
|---|---|
| XtXr | A p-vector of `t(X)` times the fitted values, X %*% colSums(alpha*mu). |

### References

G. Wang, A. Sarkar, P. Carbonetto and M. Stephens (2020). A simple new approach to variable selection in regression, with application to genetic fine-mapping. *Journal of the Royal Statistical Society, Series B* **82**, 1273-1300 doi:10.1101/501114.

Y. Zou, P. Carbonetto, G. Wang, G and M. Stephens (2022). Fine-mapping from summary data with the "Sum of Single Effects" model. *PLoS Genetics* **18**, e1010299. doi:10.1371/journal.pgen.1010299.

### See Also

`susie_get_cs` and other susie_get_* functions for extracting results; `susie_trendfilter` for applying the SuSiE model to non-parametric regression, particularly changepoint problems, and `susie_rss` for applying the SuSiE model when one only has access to limited summary statistics related to $X$ and $y$ (typically in genetic applications).

### Examples

```
# susie example
set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[1:4] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
res1 = susie(X,y,L = 10)
susie_get_cs(res1) # extract credible sets from fit
plot(beta,coef(res1)[-1])
abline(a = 0,b = 1,col = "skyblue",lty = "dashed")
plot(y,predict(res1))
abline(a = 0,b = 1,col = "skyblue",lty = "dashed")

# susie_suff_stat example
```

```
input_ss = compute_suff_stat(X,y)
res2 = with(input_ss,
            susie_suff_stat(XtX = XtX,Xty = Xty,yty = yty,n = n,
                            X_colmeans = X_colmeans,y_mean = y_mean,L = 10))
plot(coef(res1),coef(res2))
abline(a = 0,b = 1,col = "skyblue",lty = "dashed")
```

---

susie_auto                     *Attempt at Automating SuSiE for Hard Problems*

---

### Description

susie_auto is an attempt to automate reliable running of susie even on hard problems. It implements a three-stage strategy for each L: first, fit susie with very small residual error; next, estimate residual error; finally, estimate the prior variance. If the last step estimates some prior variances to be zero, stop. Otherwise, double L, and repeat. Initial runs are performed with relaxed tolerance; the final run is performed using the default susie tolerance.

### Usage

```
susie_auto(
  X,
  y,
  L_init = 1,
  L_max = 512,
  verbose = FALSE,
  init_tol = 1,
  standardize = TRUE,
  intercept = TRUE,
  max_iter = 100,
  tol = 0.01,
  ...
)
```

### Arguments

| | |
|---|---|
| X | An n by p matrix of covariates. |
| y | The observed responses, a vector of length n. |
| L_init | The initial value of L. |
| L_max | The largest value of L to consider. |
| verbose | If verbose = TRUE, the algorithm's progress, and a summary of the optimization settings, are printed to the console. |
| init_tol | The tolerance to passed to susie during early runs (set large to shorten the initial runs). |

standardize If `standardize` = TRUE, standardize the columns of X to unit variance prior to fitting. Note that `scaled_prior_variance` specifies the prior on the coefficients of X *after* standardization (if it is performed). If you do not standardize, you may need to think more carefully about specifying `scaled_prior_variance`. Whatever your choice, the coefficients returned by `coef` are given for X on the original input scale. Any column of X that has zero variance is not standardized.

intercept If `intercept` = TRUE, the intercept is fitted; it `intercept` = FALSE, the intercept is set to zero. Setting `intercept` = FALSE is generally not recommended.

max_iter Maximum number of IBSS iterations to perform.

tol A small, non-negative number specifying the convergence tolerance for the IBSS fitting procedure. The fitting procedure will halt when the difference in the variational lower bound, or "ELBO" (the objective function to be maximized), is less than `tol`.

... Additional arguments passed to [susie](#).

## Value

See [susie](#) for a description of return values.

## See Also

[susie](#)

## Examples

```
set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[1:4] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
res = susie_auto(X,y)
plot(beta,coef(res)[-1])
abline(a = 0,b = 1,col = "skyblue",lty = "dashed")
plot(y,predict(res))
abline(a = 0,b = 1,col = "skyblue",lty = "dashed")
```

---

susie_get_objective *Inferences From Fitted SuSiE Model*

---

## Description

These functions access basic properties or draw inferences from a fitted susie model.

**Usage**

```
susie_get_objective(res, last_only = TRUE, warning_tol = 1e-06)

susie_get_posterior_mean(res, prior_tol = 1e-09)

susie_get_posterior_sd(res, prior_tol = 1e-09)

susie_get_niter(res)

susie_get_prior_variance(res)

susie_get_residual_variance(res)

susie_get_lfsr(res)

susie_get_posterior_samples(susie_fit, num_samples)

susie_get_cs(
  res,
  X = NULL,
  Xcorr = NULL,
  coverage = 0.95,
  min_abs_corr = 0.5,
  dedup = TRUE,
  squared = FALSE,
  check_symmetric = TRUE,
  n_purity = 100,
  use_rfast
)

susie_get_pip(res, prune_by_cs = FALSE, prior_tol = 1e-09)
```

**Arguments**

| | |
|---|---|
| res | A susie fit, typically an output from [susie](#) or one of its variants. For susie_get_pip and susie_get_cs, this may instead be the posterior inclusion probability matrix, alpha. |
| last_only | If last_only = FALSE, return the ELBO from all iterations; otherwise return the ELBO from the last iteration only. |
| warning_tol | Warn if ELBO is decreasing by this tolerance level. |
| prior_tol | Filter out effects having estimated prior variance smaller than this threshold. |
| susie_fit | A susie fit, an output from [susie](#). |
| num_samples | The number of draws from the posterior distribution. |
| X | n by p matrix of values of the p variables (covariates) in n samples. When provided, correlation between variables will be computed and used to remove CSs whose minimum correlation among variables is smaller than min_abs_corr. |

| | |
|---|---|
| Xcorr | p by p matrix of correlations between variables (covariates). When provided, it will be used to remove CSs whose minimum correlation among variables is smaller than `min_abs_corr`. |
| coverage | A number between 0 and 1 specifying desired coverage of each CS. |
| min_abs_corr | A "purity" threshold for the CS. Any CS that contains a pair of variables with correlation less than this threshold will be filtered out and not reported. |
| dedup | If dedup = TRUE, remove duplicate CSs. |
| squared | If squared = TRUE, report min, mean and median of squared correlation instead of the absolute correlation. |
| check_symmetric | |
| | If check_symmetric = TRUE, perform a check for symmetry of matrix Xcorr when Xcorr is provided (not NULL). |
| n_purity | The maximum number of credible set (CS) variables used in calculating the correlation ("purity") statistics. When the number of variables included in the CS is greater than this number, the CS variables are randomly subsampled. |
| use_rfast | Use the Rfast package for the purity calculations. By default use_rfast = TRUE if the Rfast package is installed. |
| prune_by_cs | Whether or not to ignore single effects not in a reported CS when calculating PIP. |

## Value

susie_get_objective returns the evidence lower bound (ELBO) achieved by the fitted susie model and, optionally, at each iteration of the IBSS fitting procedure.

susie_get_residual_variance returns the (estimated or fixed) residual variance parameter.

susie_get_prior_variance returns the (estimated or fixed) prior variance parameters.

susie_get_posterior_mean returns the posterior mean for the regression coefficients of the fitted susie model.

susie_get_posterior_sd returns the posterior standard deviation for coefficients of the fitted susie model.

susie_get_niter returns the number of model fitting iterations performed.

susie_get_pip returns a vector containing the posterior inclusion probabilities (PIPs) for all variables.

susie_get_lfsr returns a vector containing the average lfsr across variables for each single-effect, weighted by the posterior inclusion probability (alpha).

susie_get_posterior_samples returns a list containing the effect sizes samples and causal status with two components: b, an num_variables x num_samples matrix of effect sizes; gamma, an num_variables x num_samples matrix of causal status random draws.

susie_get_cs returns credible sets (CSs) from a susie fit, as well as summaries of correlation among the variables included in each CS. If desired, one can filter out CSs that do not meet a specified "purity" threshold; to do this, either X or Xcorr must be supplied. It returns a list with the following elements:

| | |
|---|---|
| cs | A list in which each list element is a vector containing the indices of the variables in the CS. |

| coverage | The nominal coverage specified for each CS. |
| purity | If X or Xcorr iis provided), the purity of each CS. |
| cs_index | If X or Xcorr is provided) the index (number between 1 and L) of each reported CS in the supplied susie fit. |

## Examples

```
set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[1:4] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
s = susie(X,y,L = 10)
susie_get_objective(s)
susie_get_objective(s, last_only=FALSE)
susie_get_residual_variance(s)
susie_get_prior_variance(s)
susie_get_posterior_mean(s)
susie_get_posterior_sd(s)
susie_get_niter(s)
susie_get_pip(s)
susie_get_lfsr(s)
```

---

susie_init_coef                    *Initialize a susie object using regression coefficients*

---

## Description

Initialize a susie object using regression coefficients

## Usage

```
susie_init_coef(coef_index, coef_value, p)
```

## Arguments

| coef_index | An L-vector containing the the indices of the nonzero coefficients. |
| coef_value | An L-vector containing initial coefficient estimates. |
| p | A scalar giving the number of variables. |

## Value

A list with elements alpha, mu and mu2 to be used by susie.

## Examples

```
set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[sample(1:1000,4)] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))

# Initialize susie to ground-truth coefficients.
s = susie_init_coef(which(beta != 0),beta[beta != 0],length(beta))
res = susie(X,y,L = 10,s_init=s)
```

---

| susie_plot | *SuSiE Plots.* |
|---|---|

---

## Description

`susie_plot` produces a per-variable summary of the SuSiE credible sets. `susie_plot_iteration` produces a diagnostic plot for the susie model fitting. For `susie_plot_iteration`, several plots will be created if `track_fit = TRUE` when calling susie.

## Usage

```
susie_plot(
  model,
  y,
  add_bar = FALSE,
  pos = NULL,
  b = NULL,
  max_cs = 400,
  add_legend = NULL,
  ...
)

susie_plot_iteration(model, L, file_prefix, pos = NULL)
```

## Arguments

model          A SuSiE fit, typically an output from [susie](#) or one of its variants. For suse_plot, the susie fit must have `model$z`, `model$PIP`, and may include `model$sets`. `model` may also be a vector of z-scores or PIPs.

y              A string indicating what to plot: either `"z_original"` for z-scores, `"z"` for z-score derived p-values on (base-10) log-scale, `"PIP"` for posterior inclusion probabilities, `"log10PIP"` for posterior inclusion probabiliities on the (base-10) log-scale. For any other setting, the data are plotted as is.

| add_bar | If add_bar = TRUE, add horizontal bar to signals in credible interval. |
|---|---|
| pos | Indices of variables to plot. If pos = NULL all variables are plotted. |
| b | For simulated data, set b = TRUE to highlight "true" effects (highlights in red). |
| max_cs | The largest credible set to display, either based on purity (set max_cs between 0 and 1), or based on size (set max_cs > 1). |
| add_legend | If add_legend = TRUE, add a legend to annotate the size and purity of each CS discovered. It can also be specified as location where legends should be added, e.g., add_legend = "bottomright" (default location is "topright"). |
| ... | Additional arguments passed to [plot](plot). |
| L | An integer specifying the number of credible sets to plot. |
| file_prefix | Prefix to path of output plot file. If not specified, the plot, or plots, will be saved to a temporary directory generated using [tempdir](tempdir). |

## Value

Invisibly returns NULL.

## See Also

[susie_plot_changepoint](susie_plot_changepoint)

## Examples

```
set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[sample(1:1000,4)] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
res = susie(X,y,L = 10)
susie_plot(res,"PIP")
susie_plot(res,"PIP",add_bar = TRUE)
susie_plot(res,"PIP",add_legend = TRUE)
susie_plot(res,"PIP", pos=1:500, add_legend = TRUE)
# Plot selected regions with adjusted x-axis position label
res$genomic_position = 1000 + (1:length(res$pip))
susie_plot(res,"PIP",add_legend = TRUE,
           pos = list(attr = "genomic_position",start = 1000,end = 1500))
# True effects are shown in red.
susie_plot(res,"PIP",b = beta,add_legend = TRUE)

set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[sample(1:1000,4)] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
```

```
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
res = susie(X,y,L = 10)
susie_plot_iteration(res, L=10)
```

---

susie_plot_changepoint

*Plot changepoint data and susie fit using ggplot2*

---

### Description

Plots original data, y, overlaid with line showing susie fitted value and shaded rectangles showing credible sets for changepoint locations.

### Usage

```
susie_plot_changepoint(
  s,
  y,
  line_col = "blue",
  line_size = 1.5,
  cs_col = "red"
)
```

### Arguments

| | |
|---|---|
| s | A susie fit generated by susie_trendfilter(y,order = 0). |
| y | An n-vector of observations that are ordered in time or space (assumed equally-spaced). |
| line_col | Color for the line showing fitted values. |
| line_size | Size of the lines showing fitted values |
| cs_col | Color of the shaded rectangles showing credible sets. |

### Value

A ggplot2 plot object.

### Examples

```
set.seed(1)
mu = c(rep(0,50),rep(1,50),rep(3,50),rep(-2,50),rep(0,300))
y = mu + rnorm(500)
# Here we use a less sensitive tolerance so that the example takes
# less time; in practice you will likely want to use a more stringent
# setting such as tol = 0.001.
s = susie_trendfilter(y,tol = 0.1)
```

```
# Produces ggplot with credible sets for changepoints.
susie_plot_changepoint(s,y)
```

---

susie_rss                    *Sum of Single Effects (SuSiE) Regression using Summary Statistics*

---

### Description

susie_rss performs variable selection under a sparse Bayesian multiple linear regression of $Y$ on $X$ using the z-scores from standard univariate regression of $Y$ on each column of $X$, an estimate, $R$, of the correlation matrix for the columns of $X$, and optionally, *but strongly recommended*, the sample size n. See "Details" for other ways to call susie_rss

### Usage

```
susie_rss(
  z,
  R,
  n,
  bhat,
  shat,
  var_y,
  z_ld_weight = 0,
  estimate_residual_variance = FALSE,
  prior_variance = 50,
  check_prior = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| z | p-vector of z-scores. |
| R | p x p correlation matrix. |
| n | The sample size. |
| bhat | Alternative summary data giving the estimated effects (a vector of length p). This, together with shat, may be provided instead of z. |
| shat | Alternative summary data giving the standard errors of the estimated effects (a vector of length p). This, together with bhat, may be provided instead of z. |
| var_y | The sample variance of y, defined as $y'y/(n-1)$. When the sample variance is not provided, the coefficients (returned from coef) are computed on the "standardized" X, y scale. |
| z_ld_weight | This parameter is included for backwards compatibility with previous versions of the function, but it is no longer recommended to set this to a non-zero value. When z_ld_weight > 0, the matrix R is adjusted to be cov2cor((1-w)*R + w*tcrossprod(z)), where w = z_ld_weight. |

estimate_residual_variance

The default is FALSE, the residual variance is fixed to 1 or variance of y. If the in-sample LD matrix is provided, we recommend setting estimate_residual_variance = TRUE.

prior_variance    The prior variance(s) for the non-zero noncentrality parameterss $\tilde{b}_l$. It is either a scalar, or a vector of length L. When the susie_suff_stat option estimate_prior_variance is set to TRUE (which is highly recommended) this simply provides an initial value for the prior variance. The default value of 50 is simply intended to be a large initial value. Note this setting is only relevant when n is unknown. If n is known, the relevant option is scaled_prior_variance in susie_suff_stat.

check_prior       When check_prior = TRUE, it checks if the estimated prior variance becomes unreasonably large (comparing with 100 * max(abs(z))^2).

...               Other parameters to be passed to susie_suff_stat.

## Details

In some applications, particularly genetic applications, it is desired to fit a regression model ($Y = Xb + E$ say, which we refer to as "the original regression model" or ORM) without access to the actual values of $Y$ and $X$, but given only some summary statistics. susie_rss assumes availability of z-scores from standard univariate regression of $Y$ on each column of $X$, and an estimate, $R$, of the correlation matrix for the columns of $X$ (in genetic applications $R$ is sometimes called the "LD matrix").

With the inputs z, R and sample size n, susie_rss computes PVE-adjusted z-scores z_tilde, and calls susie_suff_stat with XtX = (n-1)R, Xty = $\sqrt{n-1}z_tilde$, yty = n-1, n = n. The output effect estimates are on the scale of $b$ in the ORM with *standardized* $X$ and $y$. When the LD matrix R and the z-scores z are computed using the same matrix $X$, the results from susie_rss are same as, or very similar to, susie with *standardized* $X$ and $y$.

Alternatively, if the user provides n, bhat (the univariate OLS estimates from regressing $y$ on each column of $X$), shat (the standard errors from these OLS regressions), the in-sample correlation matrix $R = cov2cor(crossprod(X))$, and the variance of $y$, the results from susie_rss are same as susie with $X$ and $y$. The effect estimates are on the same scale as the coefficients $b$ in the ORM with $X$ and $y$.

In rare cases in which the sample size, $n$, is unknown, susie_rss calls susie_suff_stat with XtX = R and Xty = z, and with residual_variance = 1. The underlying assumption of performing the analysis in this way is that the sample size is large (*i.e.*, infinity), and/or the effects are small. More formally, this combines the log-likelihood for the noncentrality parameters, $\tilde{b} = \sqrt{n}b$,

$$L(\tilde{b}; z, R) = -(\tilde{b}'R\tilde{b} - 2z'\tilde{b})/2,$$

with the "susie prior" on $\tilde{b}$; see susie and Wang *et al* (2020) for details. In this case, the effect estimates returned by susie_rss are on the noncentrality parameter scale.

The estimate_residual_variance setting is FALSE by default, which is recommended when the LD matrix is estimated from a reference panel. When the LD matrix R and the summary statistics z (or bhat, shat) are computed using the same matrix $X$, we recommend setting estimate_residual_variance = TRUE.

## Value

A "susie" object with the following elements:

| | |
|---|---|
| alpha | An L by p matrix of posterior inclusion probabilites. |
| mu | An L by p matrix of posterior means, conditional on inclusion. |
| mu2 | An L by p matrix of posterior second moments, conditional on inclusion. |
| lbf | log-Bayes Factor for each single effect. |
| lbf_variable | log-Bayes Factor for each variable and single effect. |
| V | Prior variance of the non-zero elements of b. |
| elbo | The value of the variational lower bound, or "ELBO" (objective function to be maximized), achieved at each iteration of the IBSS fitting procedure. |
| sets | Credible sets estimated from model fit; see [susie_get_cs](susie_get_cs) for details. |
| pip | A vector of length p giving the (marginal) posterior inclusion probabilities for all p covariates. |
| niter | Number of IBSS iterations that were performed. |
| converged | TRUE or FALSE indicating whether the IBSS converged to a solution within the chosen tolerance level. |

## References

G. Wang, A. Sarkar, P. Carbonetto and M. Stephens (2020). A simple new approach to variable selection in regression, with application to genetic fine-mapping. *Journal of the Royal Statistical Society, Series B* **82**, 1273-1300 doi:10.1101/501114.

Y. Zou, P. Carbonetto, G. Wang, G and M. Stephens (2022). Fine-mapping from summary data with the "Sum of Single Effects" model. *PLoS Genetics* **18**, e1010299. doi:10.1371/journal.pgen.1010299.

## Examples

```
set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[1:4] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))

input_ss = compute_suff_stat(X,y,standardize = TRUE)
ss   = univariate_regression(X,y)
R    = with(input_ss,cov2cor(XtX))
zhat = with(ss,betahat/sebetahat)
res  = susie_rss(zhat,R, n=n)

# Toy example illustrating behaviour susie_rss when the z-scores
# are mostly consistent with a non-invertible correlation matrix.
# Here the CS should contain both variables, and two PIPs should
# be nearly the same.
```

```
z = c(6,6.01)
R = matrix(1,2,2)
fit = susie_rss(z,R)
print(fit$sets$cs)
print(fit$pip)

# In this second toy example, the only difference is that one
# z-score is much larger than the other. Here we expect that the
# second PIP will be much larger than the first.
z = c(6,7)
R = matrix(1,2,2)
fit = susie_rss(z,R)
print(fit$sets$cs)
print(fit$pip)
```

---

| susie_trendfilter | *Apply susie to trend filtering (especially changepoint problems), a type of non-parametric regression.* |
|---|---|

---

### Description

Fits the non-parametric Gaussian regression model $y = mu + e$, where the mean $mu$ is modelled as $mu = Xb$, X is a matrix with columns containing an appropriate basis, and b is vector with a (sparse) SuSiE prior. In particular, when order = 0, the jth column of X is a vector with the first j elements equal to zero, and the remaining elements equal to 1, so that $b_j$ corresponds to the change in the mean of y between indices j and j+1. For background on trend filtering, see Tibshirani (2014). See also the "Trend filtering" vignette, vignette("trend_filtering").

### Usage

```
susie_trendfilter(y, order = 0, standardize = FALSE, use_mad = TRUE, ...)
```

### Arguments

| | |
|---|---|
| y | An n-vector of observations ordered in time or space (assumed to be equally spaced). |
| order | An integer specifying the order of trend filtering. The default, order = 0, corresponds to "changepoint" problems (*i.e.*, piecewise constant $mu$). Although order > 0 is implemented, we do not recommend its use; in practice, we have found problems with convergence of the algorithm to poor local optima, producing unreliable inferences. |
| standardize | Logical indicating whether to standardize the X variables ("basis functions"); standardize = FALSE is recommended as these basis functions already have a natural scale. |

use_mad          Logical indicating whether to use the "median absolute deviation" (MAD) method
                 to the estimate residual variance. If use_mad = TRUE, susie is run twice, first by
                 fixing the residual variance to the MAD value, then a second time, initialized
                 to the first fit, but with residual variance estimated the usual way (by maximiz-
                 ing the ELBO). We have found this strategy typically improves reliability of the
                 results by reducing a tendency to converge to poor local optima of the ELBO.
...              Other arguments passed to susie.

### Details

This implementation exploits the special structure of X, which means that the matrix-vector product
$X^T y$ is fast to compute; in particular, the computation time is $O(n)$ rather than $O(n^2)$ if X were
formed explicitly. For implementation details, see the "Implementation of SuSiE trend filtering"
vignette by running vignette("trendfiltering_derivations").

### Value

A "susie" fit; see susie for details.

### References

R. J. Tibshirani (2014). Adaptive piecewise polynomial estimation via trend filtering. *Annals of
Statistics* **42**, 285-323.

### Examples

```
set.seed(1)
mu = c(rep(0,50),rep(1,50),rep(3,50),rep(-2,50),rep(0,200))
y = mu + rnorm(400)
s = susie_trendfilter(y)
plot(y)
lines(mu,col = 1,lwd = 3)
lines(predict(s),col = 2,lwd = 2)

# Calculate credible sets (indices of y that occur just before
# changepoints).
susie_get_cs(s)

# Plot with credible sets for changepoints.
susie_plot_changepoint(s,y)
```

---

univariate_regression   *Perform Univariate Linear Regression Separately for Columns of X*

---

### Description

This function performs the univariate linear regression y ~ x separately for each column x of X.
Each regression is implemented using .lm.fit(). The estimated effect size and stardard error for
each variable are outputted.

## Usage

```
univariate_regression(
  X,
  y,
  Z = NULL,
  center = TRUE,
  scale = FALSE,
  return_residuals = FALSE
)
```

## Arguments

| | |
|---|---|
| X | n by p matrix of regressors. |
| y | n-vector of response variables. |
| Z | Optional n by k matrix of covariates to be included in all regresions. If Z is not NULL, the linear effects of covariates are removed from y first, and the resulting residuals are used in place of y. |
| center | If center = TRUE, center X, y and Z. |
| scale | If scale = TRUE, scale X, y and Z. |
| return_residuals | |
| | Whether or not to output the residuals if Z is not NULL. |

## Value

A list with two vectors containing the least-squares estimates of the coefficients (betahat) and their standard errors (sebetahat). Optionally, and only when a matrix of covariates Z is provided, a third vector residuals containing the residuals is returned.

## Examples

```
set.seed(1)
n = 1000
p = 1000
beta = rep(0,p)
beta[1:4] = 1
X = matrix(rnorm(n*p),nrow = n,ncol = p)
X = scale(X,center = TRUE,scale = TRUE)
y = drop(X %*% beta + rnorm(n))
res = univariate_regression(X,y)
plot(res$betahat/res$sebetahat)
```

# Index