LS3MU: Laplacian Spike-Slab Selector with Matrix Uncertainty

Shuyu Guo

2023/12/8

`LS3MU` is developed for high-dimensional sparse regression models where variables of design matrix contain additive measurement errors. More specifically, we assume that the measurement errors are Normally distributed with mean 0, and errors in each variable have a specific variance. Inspired by SSLASSO(Ročková and George 2018), in this package spike and slab priors are also assigned to regression parameter $\boldsymbol{\beta}$. The main function works under EM framework, where the unknown true design matrix and indicators of spike-slab priors are treated as latent variables. To avoid the effect of inappropriate choices of spike and slab scale parameters on variable selection, the final output regression coefficients are obtained following a decreasing sequence of spike parameters, and the path can be displayed by function from `LS3MU` package.

Notice: According to numerical experiments, this method works fine when all the variables include additive errors and when the variances are within a reasonable range. Also, satisfying selection result is not guaranteed if inappropriate initial values are chosen.

Model

Additive Measurement Error

Suppose we have n observations and p potential predictors (possibly $p > n$ in high-dimensional case), then the ordinary linear regression model is

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where X is the $n \times p$ design matrix and $\boldsymbol{\varepsilon}$ is the $i.i.d$ Gaussian model error with mean 0 and variance σ^2.

Assume each column of X is hidden behind independent additive measurement errors. That is, what we actually observe is the matrix Z:

$$Z = X + \Xi$$

where the elements of Ξ, measurement errors, are assumed to be independent Normal with mean 0.

We use $\boldsymbol{z_i}$ and $\boldsymbol{x_i}$ to denote the row vectors of Z and X. Specifically, each column $\Xi_{(,j)}$ has its corresponding variance τ_j. Then the covariance matrix of each row $\Xi_{(i,)}$ is $\Lambda = diag(\tau_1, ..., \tau_d)$. Thus given the ith error-in-variable observation $\boldsymbol{z_i}$, $\boldsymbol{x_i} = \boldsymbol{z_i} - \Xi_{(i,)}$ has a multivariate normal distribution::

$$\boldsymbol{x_i}|\boldsymbol{z_i}, \Lambda \sim MultiN(\boldsymbol{z_i}, \Lambda)$$

Parameter Priors

Directly following SSLASSO(Ročková and George 2018), Laplacian spike and slab priors are assumed for $\boldsymbol{\beta}$ to do variable selection.

γ_j is a binary indicator implying whether β_j has a spike prior($\gamma_j = 0$) or a slab($\gamma_j = 1$) prior. For Laplacian priors with $\lambda_0 < \lambda_1$,

$$\pi(\beta_j|\gamma_j = 1) = \frac{1}{2\lambda_1} e^{-\frac{|\beta_j|}{\lambda_1}} \ (Slab) \quad and \quad \pi(\beta_j|\gamma_j = 0) = \frac{1}{2\lambda_0} e^{-\frac{|\beta_j|}{\lambda_0}} \ (Spike)$$

Furthermore, it is assumed the prior of γ_j is a Bernoulli distribution dependent on θ:

$$\pi(\gamma_j|\theta) = \theta^{\gamma_j}(1-\theta)^{(1-\gamma_j)}$$

And θ has a Beta prior parameter (a, b):

$$\pi(\theta) \propto \theta^{(a-1)}(1-\theta)^{(b-1)}$$

When σ^2 is unknown, an Inverse Gamma prior is set:

$$\sigma^2 \sim IG(\omega/2, \omega\kappa/2) \propto (\sigma^2)^{-\frac{\omega}{2}-1} e^{-\frac{\omega\kappa}{2\sigma^2}}$$

EM Algorithm

According to the construction in the previous part and with the likelihood $y_i|\boldsymbol{x}_i, \boldsymbol{\beta}, \sigma^2 \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2)$, the goal clearly is to find maximizers $\boldsymbol{\beta}$, θ,σ for

$$\log f(\boldsymbol{y}, \boldsymbol{\beta}, \theta, \sigma^2|Z, \Lambda) = \sum_{i=1}^{n} \log \int f(y_i|\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) f(\boldsymbol{x}_i|\boldsymbol{z}_i, \Lambda) dx_i \sum_{j=1}^{p} \log \sum_{\gamma_j=0}^{1} (\pi(\beta_j|\gamma_j)\pi(\gamma_j|\theta))\pi(\theta)\pi(\sigma^2),$$

where EM algorithm can be applied, where X and $\boldsymbol{\gamma}$ are latent variables.

For **E-step**, Jensen's Inequality gives

$$\begin{aligned}
\log f(\boldsymbol{y}, \boldsymbol{\beta}, \theta, \sigma^2|Z, \Lambda) &\geq g(\boldsymbol{\beta}, \theta, \sigma^2|\boldsymbol{\beta^{(t)}}, \theta^{(t)}, (\sigma^2)^{(t)}) \\
&= -\frac{1}{2\sigma^2}(\boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{y}^\top \widetilde{X}^{(t)} \boldsymbol{\beta} + \boldsymbol{\beta}^\top \left[(\widetilde{X}^{(t)})^\top \widetilde{X}^{(t)} + n\widetilde{\Sigma}^{(t)} \right] \boldsymbol{\beta}) + Pen^{(t)}(\boldsymbol{\beta}) \\
&- \frac{n+\omega+2}{2} \log(\sigma^2) - \frac{\omega\kappa}{2\sigma^2} \\
&+ (p - \sum_{j=1}^{p} p_j^{(t)} + b - 1) \log(1-\theta) + (a - 1 + \sum_{j=1}^{p} p_j^{(t)}) \log\theta \\
&+ Const^{(t)}.
\end{aligned}$$

where

$$\widetilde{X}^{(t)} = \mathbb{E}_{X|y,Z,\Lambda,\boldsymbol{\beta}^{(t)},\sigma^{(t)}}(X)^{(t)} \ and \ \ \widetilde{\Sigma}^{(t)} = Var(\boldsymbol{x}_i|\Lambda, \boldsymbol{\beta}^{(t)}, \sigma^{(t)})$$

.

$Pen^{(t)}(\boldsymbol{\beta}) = -\sum_{j=1}^{p} Pen_j^{(t)} |\beta_j|$, where $Pen_j^{(t)} = \frac{1}{\lambda_0}(1 - p_j^{(t)}) + \frac{1}{\lambda_1} p_j^{(t)}$ and $p_j^{(t)} = \mathbb{P}(\gamma_j = 1|\boldsymbol{\beta}^{(t)}, \theta^{(t)})$.

The **M-step** is to find the maximizers of g function at each iteration. Maximizing g in terms of $\boldsymbol{\beta}$ is similar to Lasso regression.

Example

```
library(LS3MU)
library(ggplot2)
library(MASS)
```

Here we construct a high dimensional regression model $\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ with $n = 200$, $p = 500$ and $\sigma^2_{true} = var(\varepsilon_i) = 1$. Nonzero $\boldsymbol{\beta}_j$ are set to be `c(-3, -2, 1.5, 2, 3)`, which are located at `c(1, 101, 201, 301, 401)`. The elements of the true design matrix X are generated from $i.i.d$ standard Normal distribution. In this example, the variance of measurement errors of each column is arbitrarily chosen from the `seq(0.5,0.9,by = 0.1)`. With Ξ generated, we can obtain the error-in-variable matrix Z.

```
set.seed(1234)

# model dimension
n = 200
p = 500

# true beta
beta_true = rep(0,p)
true_indices = 1+100*(0:4)
true_values = c(-3, -2, 1.5, 2, 3)
beta_true[true_indices] = true_values

# construct X, y
cov = diag(p)
mu = rep(0,p)
X = mvrnorm(n,mu,cov)
epsilon = rnorm(n,0,1)
y = X %*% beta_true+epsilon

set.seed(1234)
# tauList: variances of error in variables
tau = sample(seq(0.5,0.9,by = 0.1), size = p, replace = TRUE)

# construct Z
Xi = mvrnorm(n, rep(0,p), diag(tau))
Z = X +Xi
```

Users can set the initial values for $\boldsymbol{\beta}$, σ and θ. If they are not specified by user, then the function will set `beta_init = rep(0,p)` where p is the model dimension, `theta_init = 0.5`. `sigma_init` will be determined from the sample variance of $\boldsymbol{y}$.

Since the value of the scale parameter of spike and slab priors can affect the result of variable selection, here path following method("warm starter") is used following a decreasing sequence of spike parameters. Users can customize values of scale parameters of spike and slab priors, where the spike parameters should be a decreasing sequence less than slab parameter. If not specified, then default values will be set.

```
# set initial values
beta_0 = rep(0,p)
sigma_0 = 1
theta_0 = 0.5

# slab and spike parameters
slab = 1  # slab parameter
spikes = exp(seq(log(slab),by=-0.3,length.out=20))[-1] # spike parameters
```

We use `lsum` to select variables. `sigma_update = TRUE` by default. `return_g` means whether to return the maximized value of g at each iteration, and it is `FALSE` by default.

```
# Fitting with main function lsum
lsum_fit = LS3MU::lsum(Z, y, tauList = tau, spike_params = spikes,
                slab_param = slab, beta_init = beta_0, sigma_update = TRUE,
                sigma_init = sigma_0, theta_init = theta_0, return_g = TRUE)
```

The indices of nonzero coefficients are obtained. Also, the values of selected nonzero coefficients can be obtained by calling `$beta_values`.

```
print(lsum_fit$beta_indices)
```

```
## [1]   1 101 201 301 401
```

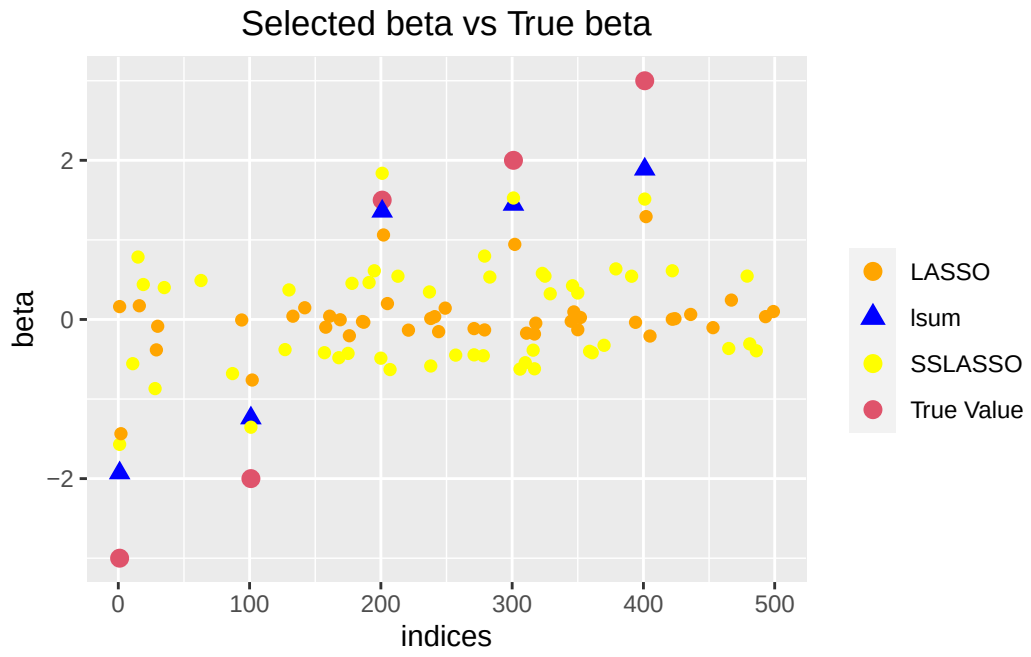
To make comparison, we also apply the original `Lasso` and `SSLASSO` to this regression model with measurement error.

```
library(glmnet)
library(SSLASSO)

# SSLASSO
sslassofit = SSLASSO(Z, y,variance = "unknown",sigma = 1)
sslasso_indices = sslassofit$model
sslasso_value = sslassofit$beta[,100][sslasso_indices]

# Lasso
set.seed(1234)
cv.fit = cv.glmnet(Z,y,alpha=1)
beta_lasso = coef(cv.fit,s = "lambda.min")
lasso_indices = which(beta_lasso!=0)
lasso_value = beta_lasso[lasso_indices]
```

The plot below shows nonzero coefficients from different methods. Particularly, Different colors represents different methods. In this error-in-variable example, we can see that `LASSO` and `SSLASSO` have high false discovery rates, while `lsum` selects nonzero variables more precisely. Though the fitted coefficients from `lsum` are not exactly equal to their true values, users can further apply ordinary linear regression where only selected variables are included and get more precise coefficients.

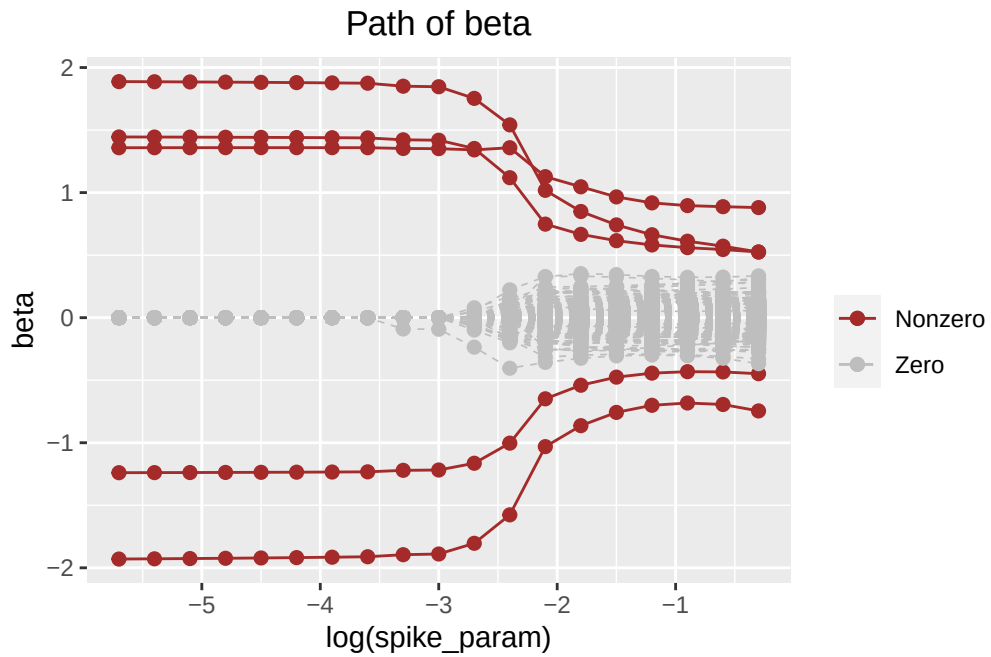
```
Plot = ggplot()+
  geom_point(aes(x = true_indices, y = true_values, color="True Value",
                 shape="True Value"),size = 3)+
  geom_point(aes(x = lsum_fit$beta_indices, y = lsum_fit$beta_values,
                 color="lsum",shape="lsum"),size = 2.5)+
  geom_point(aes(x = sslasso_indices, y = sslasso_value,
                 color="SSLASSO",shape="SSLASSO"),size = 2)+
  geom_point(aes(x = lasso_indices, y = lasso_value,
                 color="LASSO",shape="LASSO"),size = 2)+
  scale_color_manual("", values = c("True Value" = "10A19D", "lsum" = "blue",
                                    "SSLASSO" = "yellow", "LASSO" = "orange"))+
```

```
  scale_shape_manual("",values = c("True Value" = 16, "lsum" = 17,
                                   "SSLASSO" = 16, "LASSO" = 16))+
  xlab("indices")+ylab("beta")+
  labs(title ="Selected beta vs True beta")+
  theme(plot.title = element_text(size=13,hjust=0.5))

Plot
```



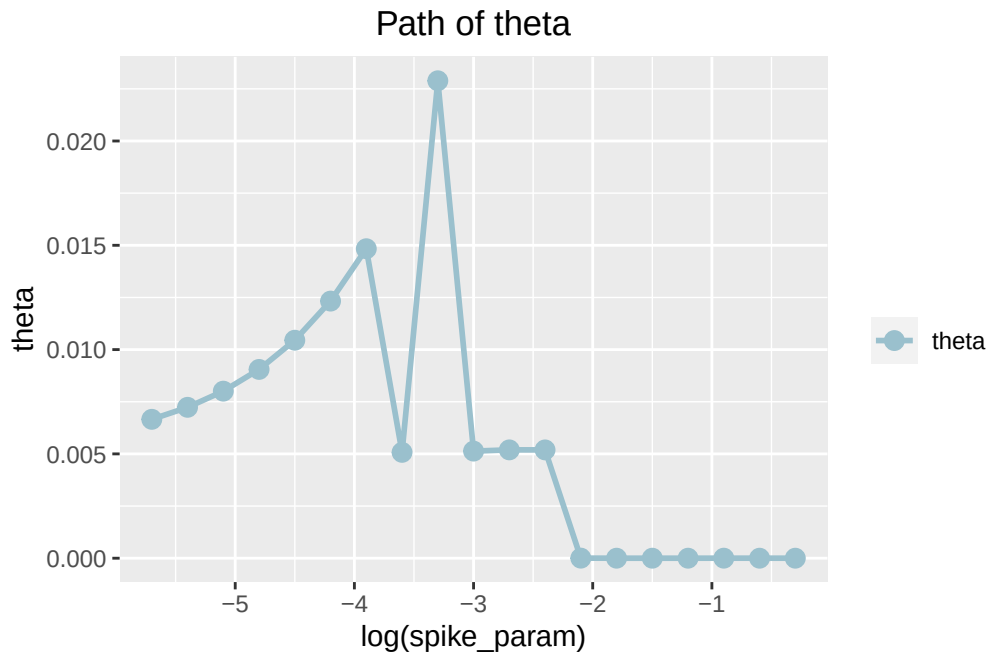
Paths of parameters can be obtained from the `pathPlot`. The default choice is `beta`. The brown line stands for paths of finally selected regression parameters. From right to left, with the decreasing of spike parameter, values of coefficients tend to be stabilized.

```
pathPlot(lsum_fit)
```

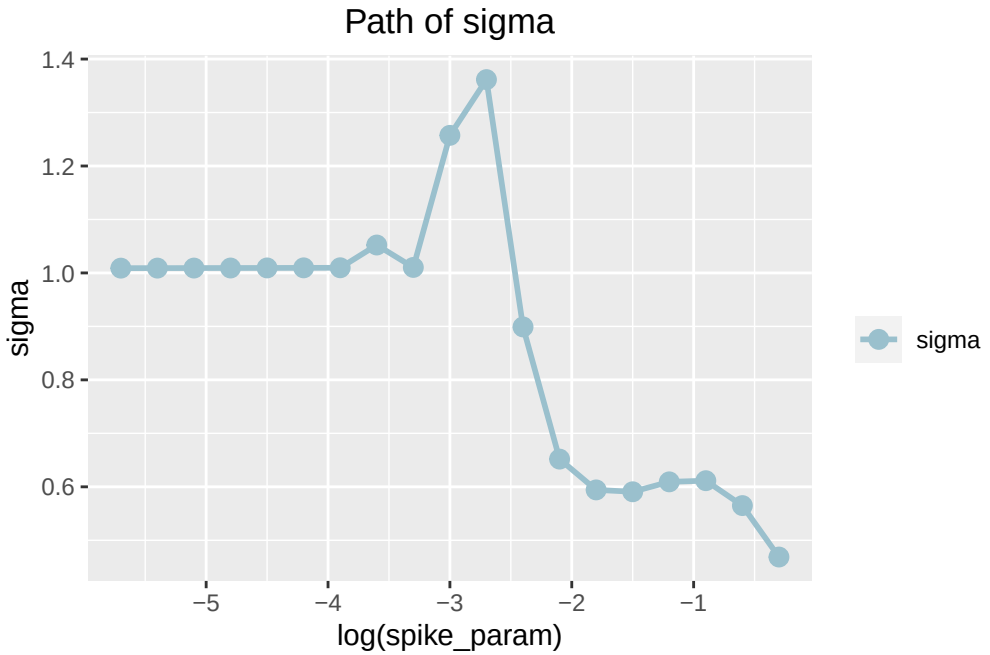


And the paths of θ and σ can also be obtained if specified.

```
pathPlot(lsum_fit,path_of = "theta")
```



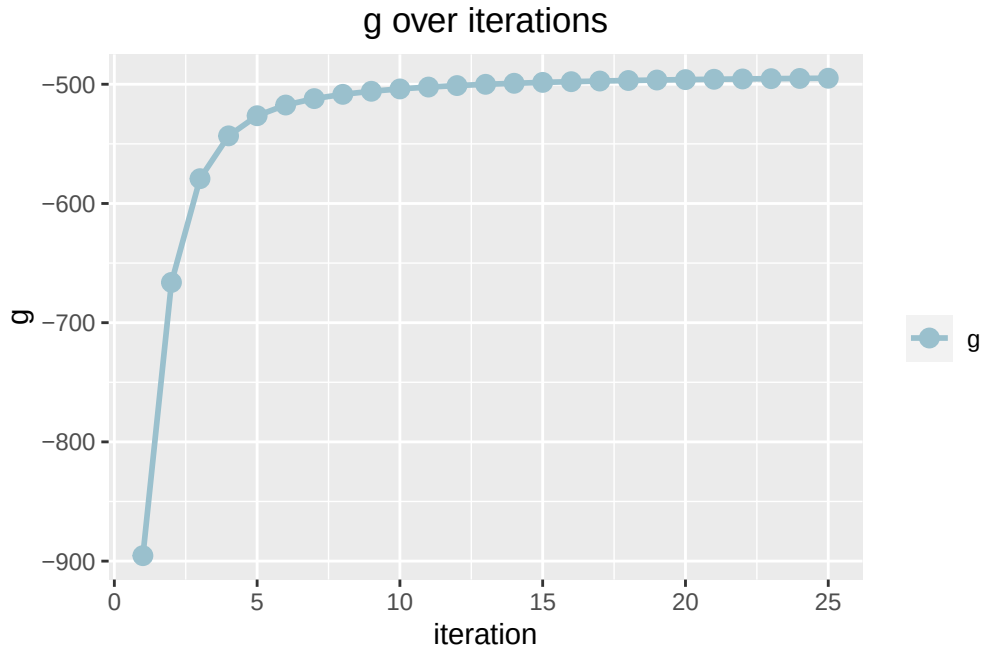

```
pathPlot(lsum_fit,path_of = "sigma")
```



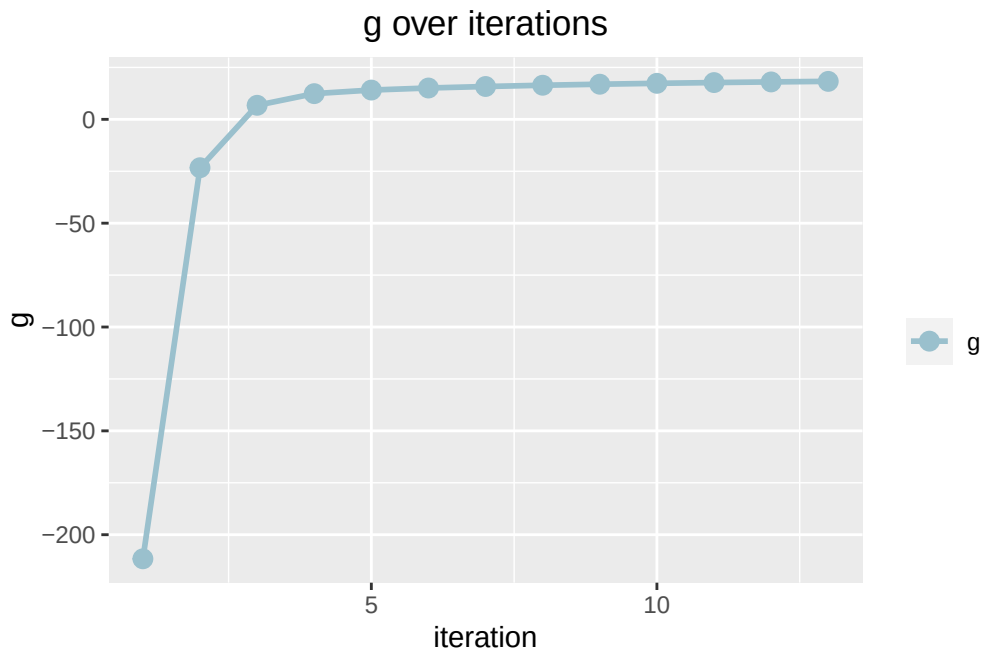
If `return_g = TRUE`, then according to the returned `g_List`, we can draw plots for `g` over iterations for each specific `spike_param` with function `gPlot`. Users must specify the value of `spike_param` (a single value) in order to obtain the plot and it must be from `spike_params` used in `lsum`. If default `spike_params` is used in `lsum`, users can check its value by calling `$spike_params` from the returned object.

From the plot below, we can check that the value of g is increasing over iterations.

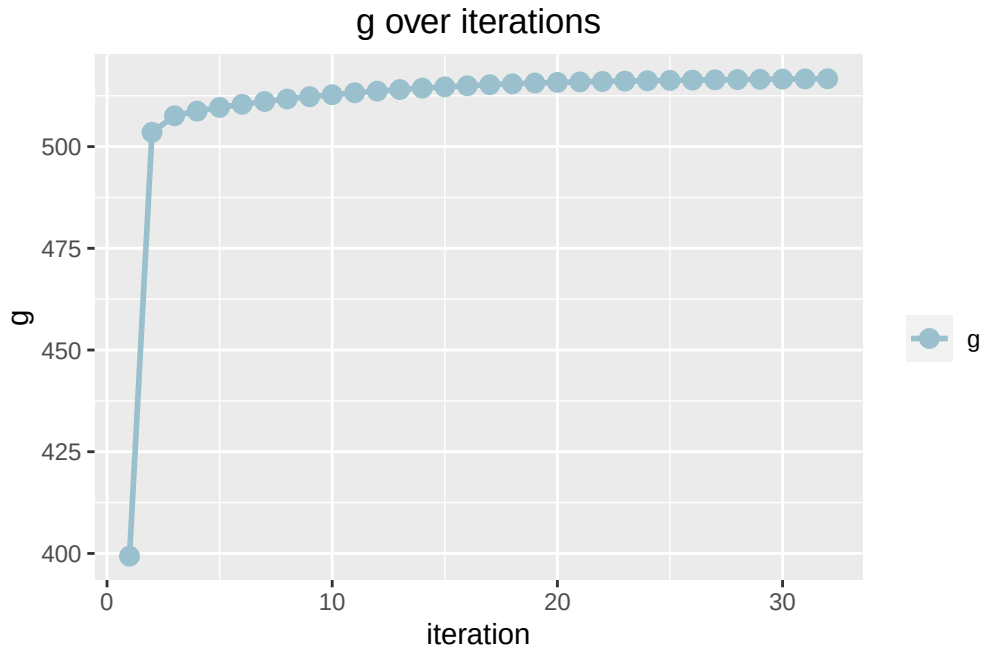
```
gPlot(lsum_fit, spike_param = lsum_fit$spike_params[1])
```



```
gPlot(lsum_fit, spike_param = lsum_fit$spike_params[5])
```




```
gPlot(lsum_fit, spike_param = lsum_fit$spike_params[9])
```



References

Ročková, Veronika, and Edward I. George. 2018. "The Spike-and-Slab LASSO." *Journal of the American Statistical Association* 113 (521): 431–44. https://doi.org/10.1080/01621459.2016.1260469.