Variable Selection using Shrinkage Priors (VsusP)

Nilson Chapagain, Debdeep Pati

2024-06-22

Contents

1 Introduction

VsusP provides robust methods for variable selection within Gaussian linear models, utilizing shrinkage priors to enhance the analysis of high-dimensional data. The main approach involves a two-step process where the number of significant variables is first determined by clustering coefficients, followed by selection based on posterior medians.

1.1 Installation

You can install the most recent version of 'VsusP' package from GitHub using the following commands:

```r
# Plain installation
devtools::install_github("nilson01/VsusP")


# For installation with vignette
devtools::install_github("nilson01/VsusP", build_vignettes = TRUE)


# load the library
library(VsusP)
```

1.2 Methodology

1.2.1 Theoretical Background

Variable selection in high-dimensional models has garnered significant interest, especially for its applications in complex biological and environmental research. Traditional methods using spike-and-slab priors face computational challenges in high dimensions, motivating the use of continuous shrinkage priors. These priors facilitate computation and interpretability by allowing for a mixture of global and local shrinkage effects, thus handling high-dimensional sparse vectors more efficiently.

In the context of Gaussian linear models:

$$Y = X\beta + \epsilon, \quad \epsilon \sim N(0, \sigma^2 I_n)$$

where Y is the response vector, X is the covariate matrix, and β is the coefficient vector. The continuous shrinkage priors, such as the horseshoe prior, have shown promise in variable selection by providing a balance between sparsity and signal retention.

1.2.2 Main Results

The `VsusP` package implements a novel post-MCMC variable selection method using shrinkage priors. This method consists primarily of two approaches: the 2-Means (2-M) and Sequential 2-Means (S2M) variable selection. Both methods are designed to address high-dimensional challenges by efficiently identifying significant variables without the need for extensive tuning parameters.

1.2.3 Sequential 2-Means (S2M) Variable Selection

The S2M method is a refined approach that aims to minimize masking errors, which often occur with heterogeneous signal strengths among variables. It involves iterative clustering of absolute coefficients, adjusting clusters based on a dynamic threshold parameter `b`, which is crucial for distinguishing between signal and noise.

Algorithm

1. Cluster the absolute values of coefficients using the k-means algorithm.
2. Continuously adjust the clustering by recalculating the mean of each cluster and adjusting the membership based on the threshold `b`.
3. Determine the set of significant variables based on the stability of cluster memberships across iterations.

Choosing Tuning Parameter b: The tuning parameter b is chosen to balance the masking and swamping errors. A visual inspection of the plot of b_i vs. the number of significant

variables helps in identifying the optimal value of b.

1.3 Example

This example demonstrates how to use `VsusP` to perform variable selection through simulated data:

```
# Assuming MASS is installed, if not uncomment the next line
# install.packages("MASS")
library(MASS)

# Set seed for reproducibility
set.seed(20221208)

# Simulate data
sim.XY <- function(n, p, beta) {
  X <- matrix(rnorm(n * p), n, p)
  Y <- X %*% beta + rnorm(n)
  return(list(X = X, Y = Y, beta = beta))
}

n <- 100
p <- 20
beta <- exp(rnorm(p))
data <- sim.XY(n, p, beta)
```

1.3.1 Applying Sequential2Means

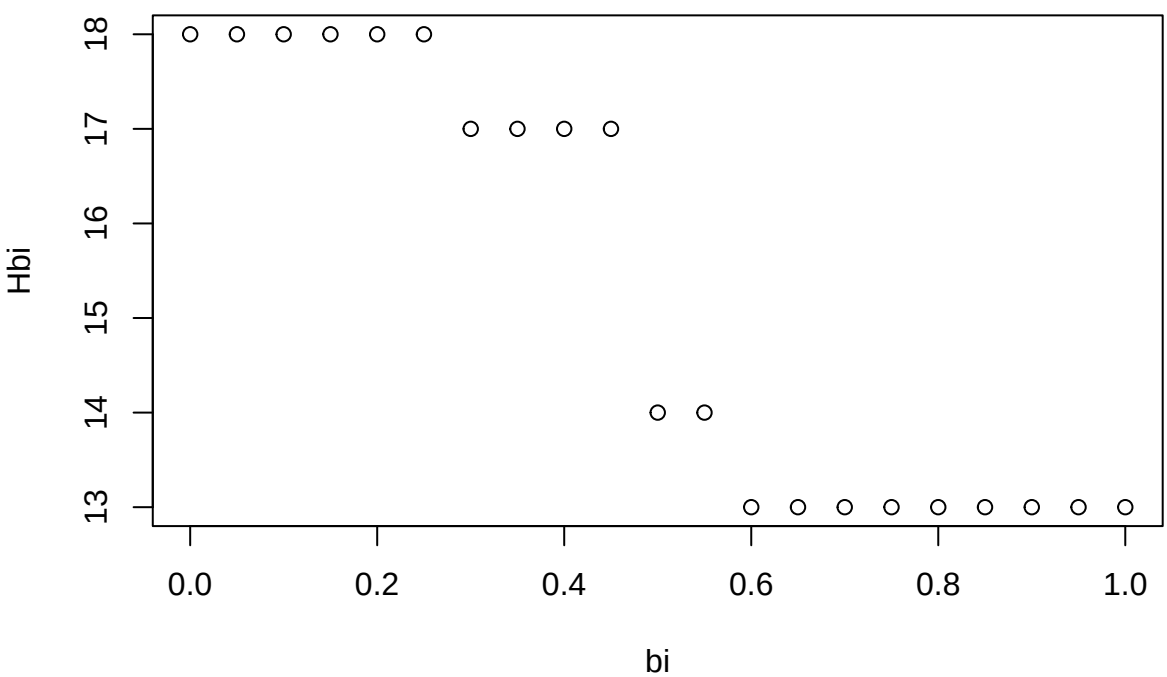
`Sequential2Means` is used to cluster the coefficients and estimate the number of significant variables:

```r
b.i <- seq(0, 1, by = 0.05)
S2M <- VsusP::Sequential2Means(X = data$X, Y = as.vector(data$Y), b.i =
↪  b.i, prior = "horseshoe+", n.samples = 5000, burnin = 2000)
Beta <- S2M$Beta
H.b.i <- S2M$H.b.i
```

1.3.2 Identifying Significant Variables

Using the `OptimalHbi` function, the optimal number of significant variables is determined:

```r
VsusP::OptimalHbi(bi = b.i, Hbi = H.b.i)
```




```r
optimal_b_i <- b.i[which.min(S2M$H.b.i)]
optimal_Hbi <- min(S2M$H.b.i)

cat("Optimal b.i: \n", optimal_b_i, "\n")
```

```
#> Optimal b.i:
#>  0.6
```

```
cat("Optimal H.b.i: \n", optimal_Hbi, "\n")
#> Optimal H.b.i:
#>  13
```

```
H <- 13
# Variable selection
impVariablesGLM <- VsusP::S2MVarSelection(Beta, H)
impVariablesGLM
#>  [1] 17 13 16 15 14 19  3  9 20  1 10 18  5
```

2 Detailed Function Descriptions

2.1 Sequential2Means

Purpose: Implements a sequential two-means clustering to segregate significant variables from noise, crucial for high-dimensional data settings. This function identifies the significant variables from the results obtained from the Sequential2Means function. It uses the posterior distribution of coefficients to determine which variables play a crucial role in explaining the response variable in a Gaussian linear model setting. The selection is based on the highest posterior median of absolute coefficients, ensuring that the most consistently relevant variables are chosen, minimizing both type I and type II errors.

Parameters: - **X**: Design matrix (n x p) where n is the number of observations and p is the number of variables. - **Y**: Response vector (n x 1). - **b.i**: Vector of tuning parameters. - **prior**: Type of shrinkage prior (e.g., "ridge", "lasso", "horseshoe"). - **n.samples**: Number of MCMC samples. - **burnin**: Number of burn-in samples.

Output: Returns a list containing: - **Beta**: Coefficient matrix across all iterations. - **b.i**: Tested values of tuning parameters. - **H.b.i**: Estimated number of significant variables for each tuning parameter value.

Process: Iteratively refines variable classification into signal or noise based on the Bayesian posterior distributions.

```
set.seed(20221208)
n <- 100
p <- 20
X <- matrix(rnorm(n * p), n, p)
Y <- X %*% exp(rnorm(p)) + rnorm(n)
b.i <- seq(0, 1, length.out = 20)
result <- VsusP::Sequential2Means(X, as.vector(Y), b.i, prior =
↪  "horseshoe+", n.samples = 5000, burnin = 2000)

cat("Beta: \n")
#> Beta:
```

```
print(result$Beta[1:5, ])
#>          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]      [,7]
↪  [,8]
#> [1,] 1.657032 5.745516 7.701031 5.371172 2.516629 1.647738 0.5777388
↪  1.1927769
#> [2,] 1.469246 5.747774 7.996900 5.219328 2.593537 1.598362 0.4943291
↪  0.9240306
#> [3,] 1.486050 5.624848 7.745928 5.424644 2.588847 1.504457 0.7120145
↪  1.1200253
#> [4,] 1.595605 5.530576 7.911171 5.323639 2.405887 1.405196 0.3522210
↪  1.4086268
```

```
#> [5,] 1.562255 5.818370 7.785683 5.469377 2.631315 1.572353 0.4054474
↪  1.2112935
#>              [,9]     [,10]     [,11]      [,12]      [,13]     [,14]
↪  [,15]
#> [1,]  0.043974973 2.674129 2.268507 0.7848083 1.0847562 16.38575
↪  14.14053
#> [2,]  0.001202963 2.687869 2.078930 0.7704686 1.0581148 16.40872
↪  14.22505
#> [3,] -0.030921386 2.946734 2.298417 0.3376274 0.8136973 16.21642
↪  14.06580
#> [4,] -0.028563184 2.506436 2.283590 0.3324814 1.1120314 16.40790
↪  14.16857
#> [5,]  0.142924236 2.868259 2.089031 0.7524880 0.8503163 16.47871
↪  14.16593
#>           [,16]     [,17]     [,18]      [,19]      [,20]
#> [1,] 0.02399739 1.063667 1.001282 0.8281522 0.3303679
#> [2,] 0.04255695 1.136850 1.228612 0.8660172 0.3211069
#> [3,] 0.41106684 1.061000 0.946370 1.0617730 0.4043102
#> [4,] 0.21156526 1.381213 1.229646 1.1259068 0.1975442
#> [5,] 0.17997463 1.099806 1.253208 1.0301509 0.3296517
```

```
cat("\n\n")
```

```
cat("H.b.i: \n")
#> H.b.i:
```

```r
print(result$H.b.i)
#>  [1] 18 18 18 18 18 18 18 18 18 18 18 18 18 18 17 15 15 15 15 15
```

2.2 OptimalHbi

Purpose: Determines the optimal number of significant variables by assessing stability across tuning parameters, ensuring balanced model complexity.

This function determines the optimal number of significant variables ('H') by assessing the stability across different values of tuning parameters. It is pivotal for ensuring the model neither overfits nor underfits, providing a robust means to handle model complexity and accuracy efficiently.

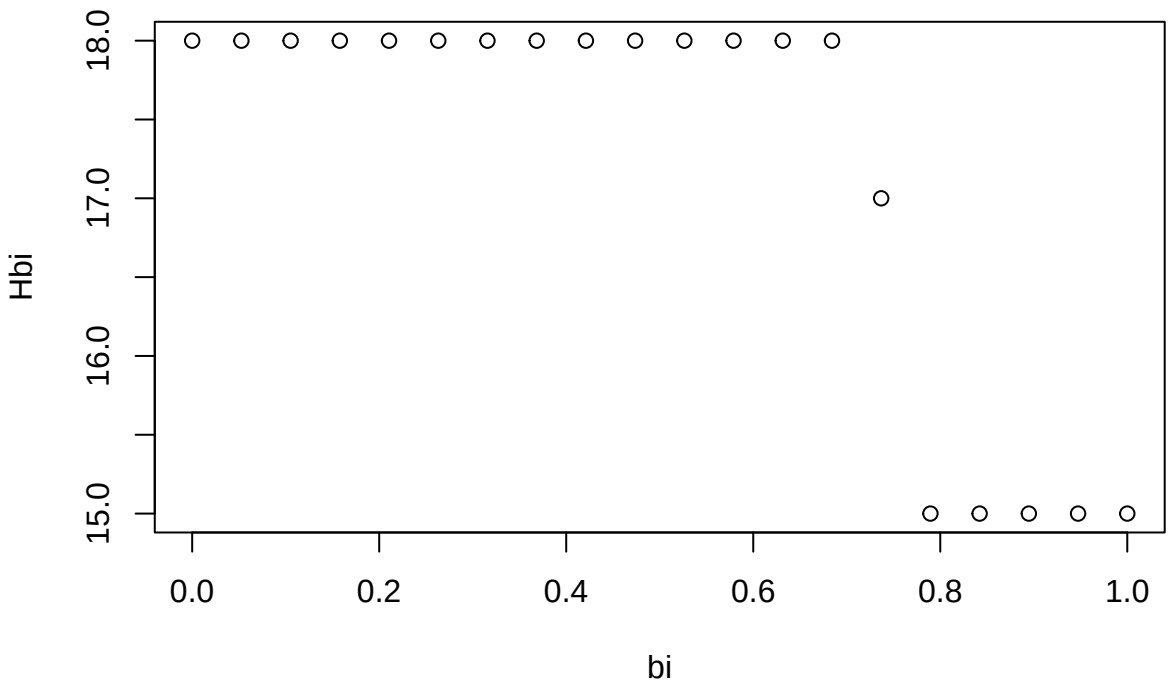
The optimal value of the tuning parameter 'H', representing the best compromise between model complexity and fitting accuracy. This function plots 'bi' vs 'Hbi' and the user should select 'H' at the point of sharpest change in the plot, indicating the most stable variable selection.

Parameters: - **bi**: Vector of tuning parameters. - **Hbi**: Estimated number of signals for each tuning parameter.

Output: Plot to select optimal 'H' value, indicating the most appropriate number of significant variables to prevent overfitting or underfitting.

Process: Select the 'H' at the point of the sharpest change in the plot of 'bi' vs 'Hbi', indicative of stable variable selection.

```r
VsusP::OptimalHbi(b.i, result$H.b.i)
```



2.3 S2MVarSelection

Purpose: Extracts a subset of significant variables based on the highest posterior medians of absolute coefficients from the Sequential2Means output. An alternative to S2MVarSelection that provides a more direct approach to variable selection, using refined criteria for determining significance directly from the clustering outputs of Sequential2Means. It leverages detailed clustering information, specifically the median absolute values of the coefficients, to prioritize variables, providing a method that might be more responsive to subtle variations in data structure and signal intensity.

Parameters: - **Beta**: Matrix of posterior samples. - **H**: Number of significant variables estimated.

Output: Indices of significant variables, directly identifying key predictors.

Process: Selects variables whose coefficients show consistent importance across samples, minimizing error rates.

```r
H <- 17
significantVariables <- VsusP::S2MVarSelection(result$Beta, H)
cat("Significant Variables: \n")
#> Significant Variables:
```

```r
print(significantVariables)
#>  [1] 14 15  3  2  4 10  5 11  1  6 17  8 18 13 19  7 12
```

2.4 Sequential2MeansBeta

Purpose: Extends the functionality of `Sequential2Means` by allowing the specification of a range for the tuning parameters, enabling more flexible model testing. This function extends Sequential2Means by allowing for precise control over the range and density of tuning parameters to be tested, accommodating advanced scenarios where the distribution of signals is hypothesized to be non-uniform. Similar to Sequential2Means, it returns a list with p (number of variables), b.i (tested tuning parameters), and H.b.i (estimated significant variables for each tuning parameter), facilitating detailed analysis of variable significance across a customized range of model complexities.

Parameters: - **Beta**: Matrix of N by p dimensions consisting of N posterior samples of p variables. - **lower**: Lower bound of the tuning parameter values. - **upper**: Upper bound of the tuning parameter values. - **l**: Number of points within the tuning parameter range.

Output: Similar to `Sequential2Means`, it returns a list that includes: - **p**: Number of variables. - **b.i**: Vector of tested tuning parameters. - **H.b.i**: Estimated number of significant variables for each tuning parameter.

Process: Evaluates the significance of coefficients across a user-defined range of tuning parameters, enhancing the ability to discern the underlying signal structure.

```r
Beta <- result$Beta
lower <- 0
upper <- 1
l <- 20

S2MBeta = VsusP::Sequential2MeansBeta(Beta, lower, upper, l)

cat("p : \n")
#> p :
```

```r
print(S2MBeta$p)
#> [1] 20
```

```r
cat("\n\n")
```

```r
cat("bi : \n")
#> bi :
```

```r
print(S2MBeta$b.i)
#>  [1] 0.00000000 0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
#>  [7] 0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
#> [13] 0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684
#> [19] 0.94736842 1.00000000
```

```r
cat("\n\n")
```

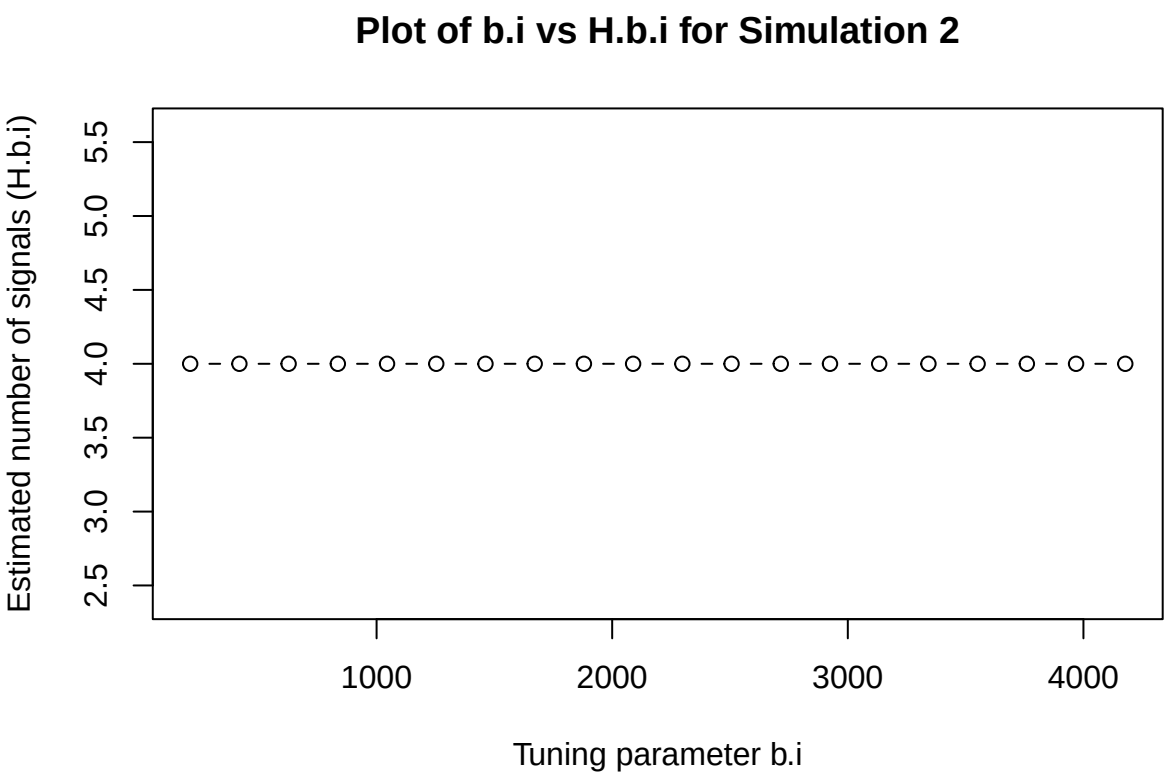
```r
cat("Hbi : \n")
#> Hbi :
```

```r
print(S2MBeta$H.b.i)
#>  [1] 18 18 18 18 18 18 18 18 18 18 18 18 18 18 17 15 15 15 15 15
```

3 Simulation example

```r
library(MASS)
library(VsusP)

set.seed(12345)

n <- 50
p <- 300
r <- 10
rho <- 0.95

Sigma <- diag(1, p)
block_size <- 5
num_blocks <- p / block_size
```

```r
for (b in 0:(num_blocks - 1)) {
  block_start <- b * block_size + 1
  block_end <- min(p, block_start + block_size - 1)
  Sigma[block_start:block_end, block_start:block_end] <- rho
  diag(Sigma[block_start:block_end, block_start:block_end]) <- 1
}

Sigma <- Sigma + diag(0.001, p)  # Ensure positive definiteness

X <- mvrnorm(n, mu = rep(0, p), Sigma = Sigma)
beta_true <- c(6, rep(3, 4), rep(1, 5), rep(0, p - 10))
Y <- as.vector(X %*% beta_true + rnorm(n))

var_y <- var(Y)
b_i_range <- seq(0.5 * var_y, 10 * var_y, length.out = 20)  # Check the
↪  scale of var(Y)

results <- Sequential2Means(X = X, Y = Y, b.i = b_i_range, prior =
↪  "horseshoe+", n.samples = 5000, burnin = 2000)

plot(b_i_range, results$H.b.i, type = 'b', xlab = "Tuning parameter b.i",
↪  ylab = "Estimated number of signals (H.b.i)",
     main = "Plot of b.i vs H.b.i for Simulation 2")
```




```r
optimal_b_i <- b_i_range[which.min(results$H.b.i)]
optimal_Hbi <- min(results$H.b.i)

cat("Optimal b.i:", optimal_b_i, "\n")
#> Optimal b.i: 208.8816
```

```r
cat("Optimal H.b.i:", optimal_Hbi, "\n")
#> Optimal H.b.i: 4
```

```r
H <- optimal_Hbi
# Variable selection
Beta <- results$Beta
```

```r
impVariablesGLM <- S2MVarSelection(Beta, H)

impVariablesGLM

#> [1]  3  1  2 10
```

4 References

- **Li, H., & Pati, D.**: "Variable selection using shrinkage priors." Computational Statistics & Data Analysis, 107, pp.107-119.
- **Makalic, E. & Schmidt, D. F. (2016)**: "High-Dimensional Bayesian Regularised Regression with the BayesReg Package." arXiv:1611.06649.
- **Bhattacharya, A., Pati, D., Pillai, N.S., & Dunson, D.B. (2015)**: "Dirichlet-laplace priors for optimal shrinkage." J. Amer. Statist. Assoc. 110 (512), 1479–1490.
- **Rosenwald, A., et al. (2002)**: "The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma." New Engl. J. Med. 346 (25), 1937–1947.
- **Bhattacharya, A., & Dunson, D. (2011)**: "Sparse Bayesian infinite factor models." Biometrika 98 (2), 291–306.