HDS 8. Bayesian variable selection

Matti Pirinen, University of Helsinki

27.11.2021

In HDS6, we saw that LASSO has a property of setting many of the coefficients to exactly zero at the optimum of the objective function and hence we said that LASSO does *variable selection*. A technical reason for this behavior was the LASSO's prior distribution for coefficients that had a sharp peak near zero. In HDS7, we embedded the LASSO model into a fully Bayesian model `blasso` that allowed us to estimate the full posterior distribution of the coefficients but that was computationally more heavy to apply. Next we will consider conceptually simpler Bayesian models for variable selection, where the prior distribution of a coefficient states that coefficient is non-zero with probability π_1 and has a positive probability mass $1-\pi_1$ of being exactly zero.

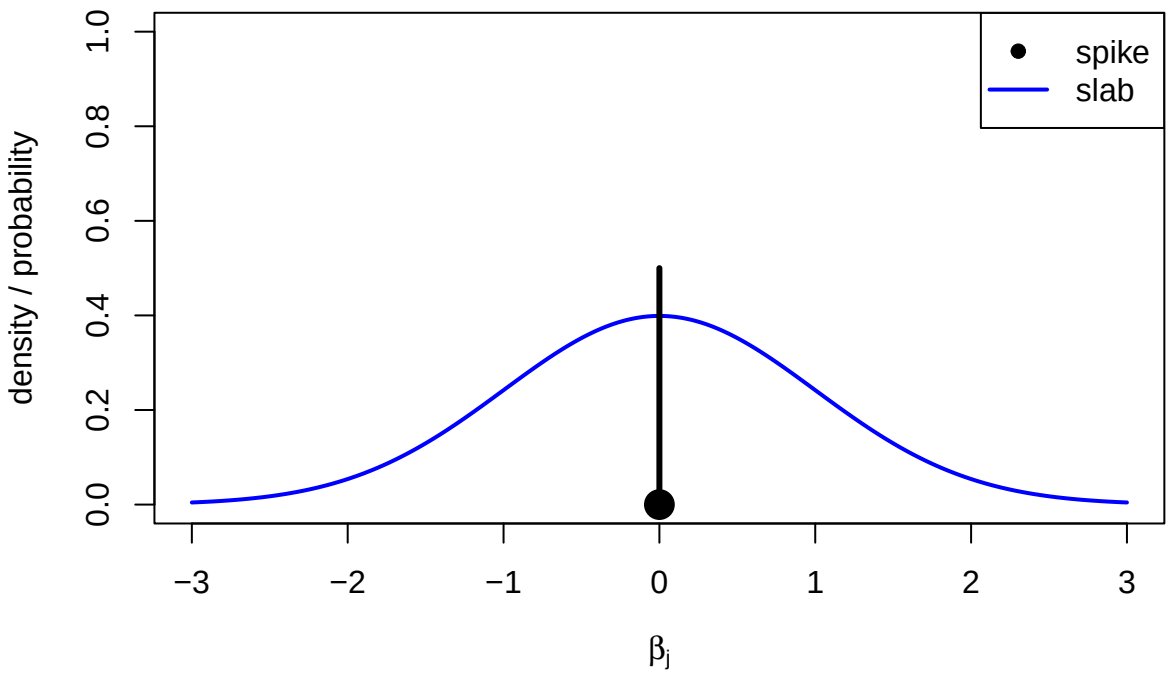
Spike and slab prior (SSP)

Under SSP, we model individual coefficient β_j using a mixture distribution

$$\beta_j \,|\, \pi_1, \tau^2 \sim (1-\pi_1)\delta_0 + \pi_1 \mathcal{N}(0, \tau^2)$$

where δ_0 is point mass at 0. In other words,

$$\begin{cases} \beta_j = 0, & \text{with probability } 1-\pi_1, \\ \beta_j \sim \mathcal{N}(0, \tau^2), & \text{with probability } \pi_1. \end{cases}$$



Suppose we have n samples and p predictors (columns of $\boldsymbol{X}$) to model outcome vector $\boldsymbol{y}$. Let $\boldsymbol{\gamma} = (\gamma_j)_{j=1}^p$ be vector of binary indicators indicating which variables are non-zero, that is,

$$\gamma_j = \begin{cases} 1, & \text{if } \beta_j \neq 0, \\ 0, & \text{if } \beta_j = 0. \end{cases}$$

We can assign to each predictor, independently, an $\text{SSP}(\pi_1, \tau^2)$ prior where the values π_1 and τ^2 are shared between the predictors. Value of π_1 could be determined based on which proportion of predictors we expect to be non-zero, and τ^2 could be determined based on what magnitude of coefficient values we are expecting. Such model is implemented in `BoomSpikeSlab` package. (Later we will discuss extensions that estimate π_1 and τ^2 from data.)

Let's try it on the same data we analyzed using Bayesian LASSO in HDS 7 that had $n = 250$, $p = 30$ and first 3 variables had an effect size of 0.229 while the remaining 27 were exactly 0. We use prior $\pi_1 = 5/30 \approx 0.167$ (by setting `expected.model.size = 5`) and $\tau^2 = 1$ (by setting `prior.beta.sd = rep(1,p)`).

```
set.seed(122)
p = 30
n = 250
phi = 0.05 #variance explained by x_1, should be 0 < phi < 1.
b = rep(c(sqrt(phi / (1-phi)), 0), c(3, p-3)) #effects 1,2,3 are non-zero, see Lect. 0.1 for "phi"
X = scale(matrix(rnorm(n*p), nrow = n)) #cols 1,2,3 of X have effects, other cols are noise
eps = scale(rnorm(n, 0, 1)) #epsilon, error term
y = scale(X%*%b + eps, scale = FALSE) #makes y have mean = 0
library(BoomSpikeSlab)
prior = IndependentSpikeSlabPrior(X, y,
                                  expected.model.size = 5,
                                  prior.beta.sd = rep(1,p))
lm.ss = lm.spike(y ~ X - 1, niter = 1000, prior = prior)
```

```
summary(lm.ss)
```

```
## coefficients:
##          mean      sd mean.inc sd.inc inc.prob
## X3   3.23e-01 0.06400  0.32300 0.0640    1.000
## X1   2.67e-01 0.07150  0.27000 0.0666    0.990
## X2   2.36e-01 0.08200  0.24800 0.0645    0.953
## X6   9.01e-03 0.03640  0.12500 0.0622    0.072
## X27 -4.77e-03 0.02620 -0.10400 0.0686    0.046
## X11  4.90e-03 0.02680  0.10700 0.0699    0.046
## X23 -3.68e-03 0.02180 -0.11200 0.0494    0.033
## X30 -1.26e-03 0.01280 -0.04650 0.0640    0.027
## X24 -1.81e-03 0.01430 -0.06940 0.0578    0.026
## X15 -2.48e-03 0.01910 -0.09940 0.0722    0.025
## X9  -2.09e-03 0.01720 -0.08700 0.0716    0.024
## X12 -1.56e-03 0.01400 -0.07100 0.0648    0.022
## X7  -1.15e-03 0.01100 -0.05200 0.0545    0.022
## X4  -4.15e-04 0.00790 -0.01980 0.0521    0.021
## X20  1.52e-03 0.01350  0.07610 0.0600    0.020
## X17 -3.05e-04 0.00713 -0.01700 0.0518    0.018
## X10 -1.11e-03 0.01210 -0.06170 0.0682    0.018
## X8   5.69e-05 0.00767  0.00335 0.0605    0.017
## X18  8.86e-04 0.00959  0.05540 0.0540    0.016
## X13 -1.49e-04 0.00675 -0.00933 0.0542    0.016
## X26  5.54e-04 0.00767  0.03950 0.0535    0.014
## X21 -2.10e-05 0.00567 -0.00150 0.0497    0.014
## X5  -7.90e-04 0.00820 -0.05640 0.0423    0.014
## X22  4.16e-04 0.00686  0.03200 0.0531    0.013
## X16  7.18e-04 0.00890  0.05520 0.0578    0.013
## X14 -6.55e-04 0.00960 -0.05040 0.0704    0.013
## X29  4.95e-04 0.00831  0.04120 0.0667    0.012
## X28 -2.64e-05 0.00607 -0.00220 0.0578    0.012
## X25  6.34e-04 0.00673  0.05760 0.0302    0.011
## X19 -1.65e-04 0.00523 -0.01830 0.0551    0.009
##
## residual.sd =
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8482  0.9661  0.9950  0.9963  1.0262  1.1503
##
## r-square    =
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.0691  0.1492  0.2001  0.1963  0.2459  0.4187
```

We see that the 3 variables with true effects have posterior inclusion probabilities (PIP) > 0.90 whereas all others have PIPs < 0.10. In the results, we have two sets of posterior means and sds: first from the full posterior distribution and second conditional on variable being included in the model. When the variable has a large PIP, then the two are similar, but for small PIPs, the two differ and the full unconditional mean is a shrunk version of the conditional mean where the shrinkage factor is the PIP. For example, for variable `X6` the posterior mean is 0.125136 conditional on the variable being non-zero, and `X6` is non-zero with a posterior probability of 0.072 Thus, the (unconditional) posterior mean is

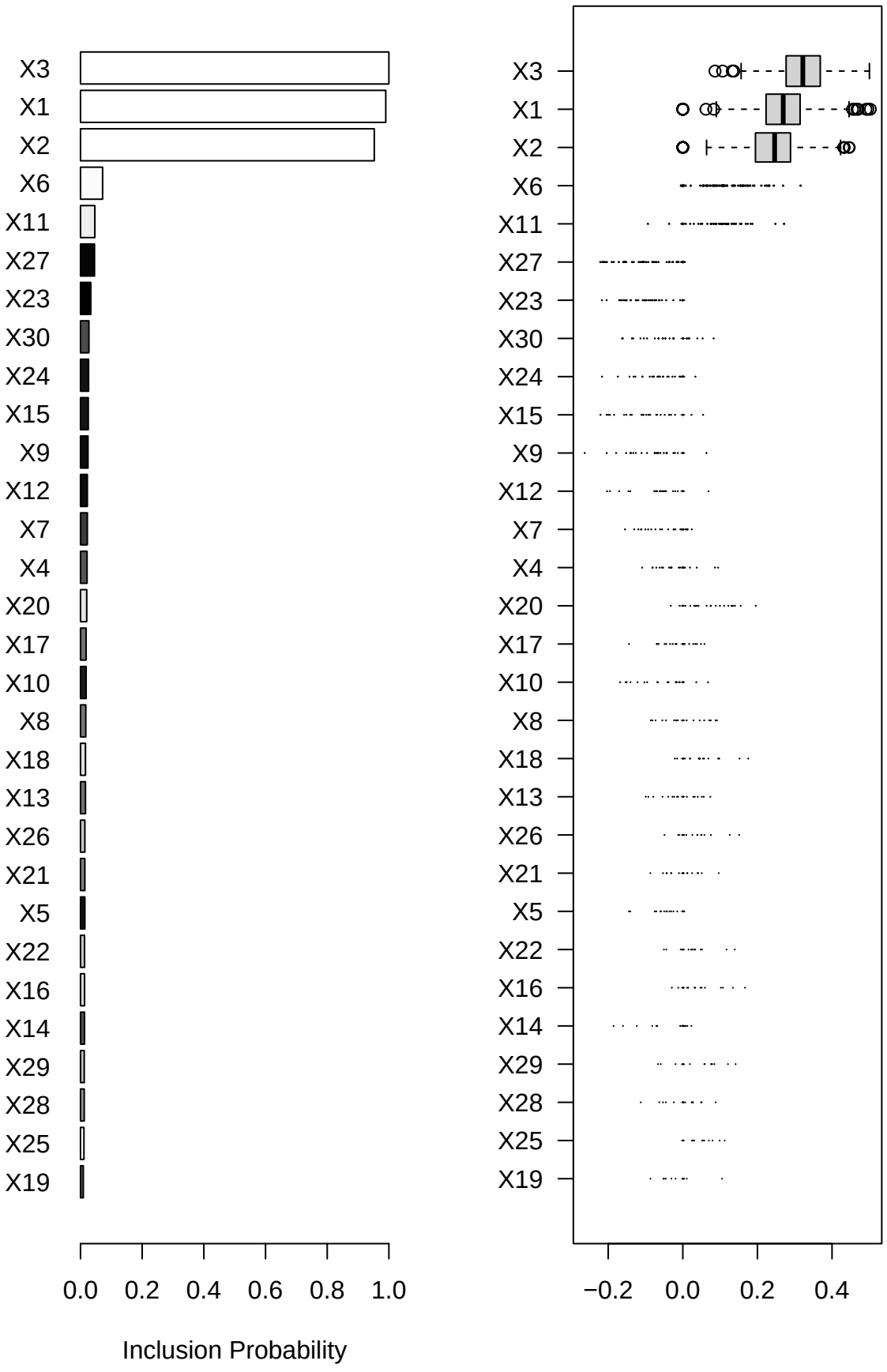
$0.072 \cdot 0.125136 + 0.928 \cdot 0 = 0.0090098.$

Note that in similar spirit, one could extend the LASSO method to **relaxed LASSO** where one first runs

LASSO to choose the non-zero variables (corresponding to unconditional analysis) and then fits an unpenalized model using only the chosen variables (corresponding to conditional analysis). Statistical inference for such a stepwise procedure is, however, complicated whereas in the Bayesian approach the posterior distribution is directly available and a natural basis for inference.

Let's visualize the PIPs and posterior distributions of the coefficients.

```
par(mfrow = c(1,2))
plot(lm.ss) #plots PIPs
plot(lm.ss,"coefficients")#plots coeffs
```


X3
X1
X2
X6
X11
X27
X23
X30
X24
X15
X9
X12
X7
X4
X20
X17
X10
X8
X18
X13
X26
X21
X5
X22
X16
X14
X29
X28
X25
X19
0.0
0.2
0.4
0.6
0.8
1.0
Inclusion Probability
−0.2
0.0
0.2
0.4

Posterior on the model size can also be plotted.

```
plot(lm.ss,"size", freq = F)
```

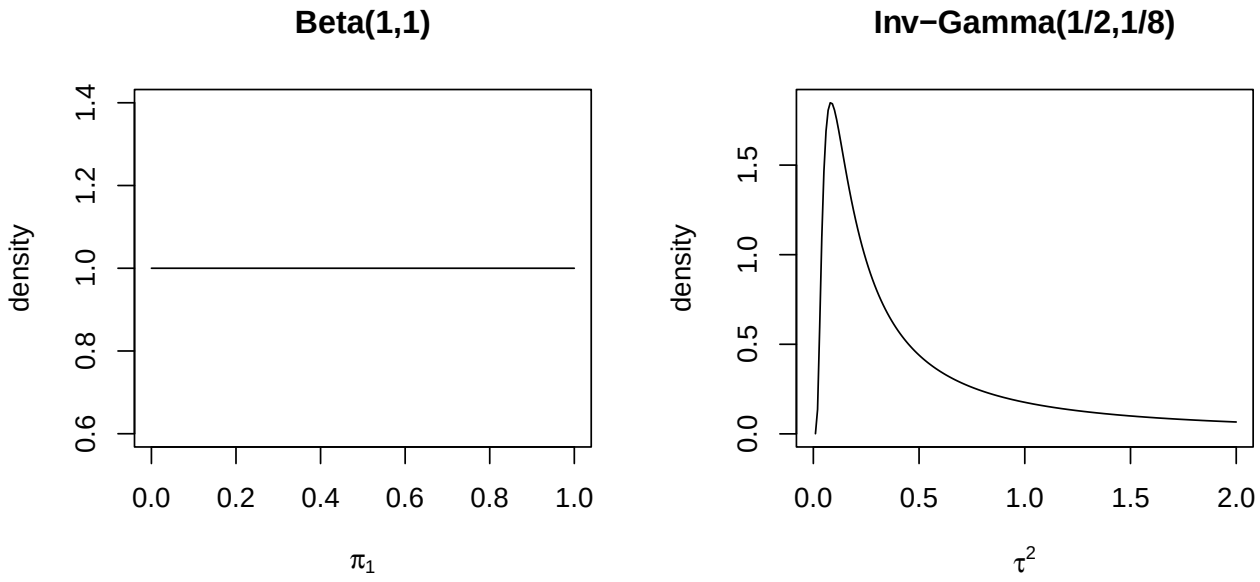
Extension to estimating π_1 and τ^2

To make the model fully Bayesian, we would like to estimate also parameters π_1 and τ^2. For that, we will consider τ^2 as being specified relative to the error variance σ^2, i.e., we use prior

$$\beta_j \,|\, \pi_1, \tau^2, \sigma^2 \sim (1-\pi_1)\delta_0 + \pi_1 \mathcal{N}(0, \sigma^2\tau^2).$$

For example, these parameters can be given priors $\pi_1 \sim \text{Beta}(b_0, b_1)$ and $\tau^2 \sim \text{Inverse-Gamma}(1/2, s^2/2)$ where the hyper-parameters b_0, b_1 and s are given such fixed values that the priors cover all reasonable values of the parameters in a fairly uniform manner. Additionally, a flat prior on σ^2 could be an Inverse-Gamma(a, a) where a is a small positive value, such as 0.01, as such a distribution approximates a uniform prior on $\log(\sigma^2)$ scale (so called Jeffrey's prior for the variance parameter). By using such "flat priors" we hope to achieve a posterior distribution that has been guided by information from the data while the effect of the prior distribution on the posterior remains small.

In the following analysis, we will use hyper-parameters $b_0 = b_1 = 1$, $s = 0.5$ and $a = 0.01$ that lead to the following prior densities on π_1 and τ^2.



The following Gibbs sampler code to implement this model is adapted from Fabian Dablander https://fabiandablander.com/r/Spike-and-Slab.html

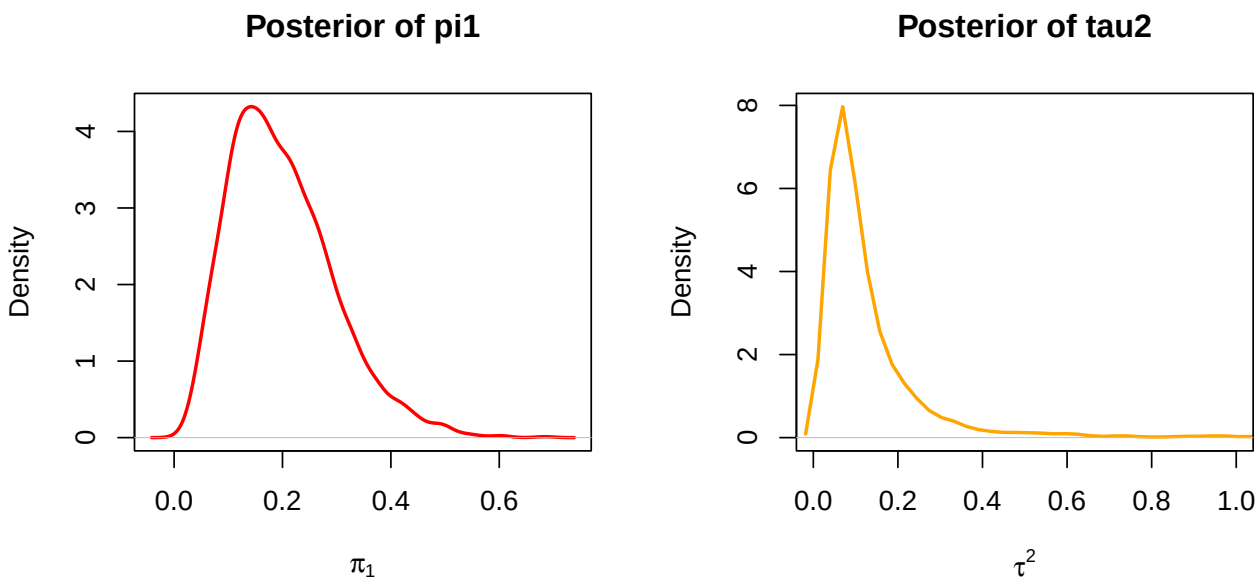
```
#' Spike-and-Slab Regression using Gibbs Sampling for p > 1 predictors
#'
#' @param y: vector of responses
#' @param X: matrix of predictor values
#' @param nr_samples: indicates number of samples drawn
#' @param a1: parameter a1 of Gamma prior on variance sigma2e
#' @param a2: parameter a2 of Gamma prior on variance sigma2e
#' @param pi1: parameter of prior over mixture weight
#' @param a: 1st parameter of Beta prior for pi1
#' @param b: 2nd parameter of Beta prior for pi1
#' @param burnin: number of samples we discard ('burnin samples')
#'
#' @returns matrix of posterior samples from parameters gamma, beta, tau2, sigma2e, pi1
ss_regress <- function(
  y, X, a1 = .01, a2 = .01, pi1 = .5,
  a = 1, b = 1, s = 1/2, nr_samples = 6000, nr_burnin = round(nr_samples / 4, 2)
  ) {

  p <- ncol(X)
  n <- nrow(X)

  # res is where we store the posterior samples
  res <- matrix(NA, nrow = nr_samples, ncol = 2*p + 1 + 1 + 1)

  colnames(res) <- c(
    paste0('gamma', seq(p)),
    paste0('beta', seq(p)),
    'sigma2', 'tau2', 'pi1'
  )

  # take the MLE estimate as the values for the first sample
  m <- lm(y ~ X - 1)
  res[1, ] <- c(rep(0, p), coef(m), var(predict(m) - y), 1, .5)
```

```r
# compute only once
XtX <- t(X) %*% X
Xty <- t(X) %*% y

# we start running the Gibbs sampler
for (i in seq(2, nr_samples)) {

  # first, get all the values of the previous time point
  gamma_prev <- res[i-1, seq(p)]
  beta_prev <- res[i-1, seq(p + 1, 2*p)]
  sigma2_prev <- res[i-1, ncol(res) - 2]
  tau2_prev <- res[i-1, ncol(res) - 1]
  pi1_prev <- res[i-1, ncol(res)]

  ## Start sampling from the conditional posterior distributions
  ##############################################################

  # sample pi1 from a Beta
  pi1_new <- rbeta(1, a + sum(gamma_prev), b + sum(1 - gamma_prev))

  # sample sigma2e from an Inverse-Gamma
  err <- y - X %*% beta_prev
  sigma2_new <- 1 / rgamma(1, a1 + n/2, a2 + t(err) %*% err / 2)

  # sample tau2 from an Inverse Gamma
  tau2_new <- 1 / rgamma(
    1, 1/2 + 1/2 * sum(gamma_prev),
    s^2/2 + t(beta_prev) %*% beta_prev / (2*sigma2_new)
  )

  # sample beta from multivariate Gaussian
  beta_cov <- qr.solve((1/sigma2_new) * XtX + diag(1/(tau2_new*sigma2_new), p))
  beta_mean <- beta_cov %*% Xty * (1/sigma2_new)
  beta_new <- mvtnorm::rmvnorm(1, beta_mean, beta_cov)

  # sample each gamma_j in random order
  for (j in sample(seq(p))) {

    # get the betas for which beta_j is zero
    gamma0 <- gamma_prev
    gamma0[j] <- 0
    bp0 <- t(beta_new * gamma0)

    # compute the z variables and the conditional variance
    xj <- X[, j]
    z <- y - X %*% bp0
    cond_var <- sum(xj^2) + 1/tau2_new

    # compute chance parameter of the conditional posterior of gamma_j (Bernoulli)
    l0 <- log(1 - pi1_new)
    l1 <- (
      log(pi1_new) - .5 * log(tau2_new*sigma2_new) +
      sum(xj*z)^2 / (2*sigma2_new*cond_var) + .5 * log(sigma2_new / cond_var)
```

```
      )

      # sample gamma_j from a Bernoulli
      gamma_prev[j] <- rbinom(1, 1, exp(l1) / (exp(l1) + exp(l0)))
    }

    gamma_new <- gamma_prev

    # add new samples
    res[i, ] <- c(gamma_new, beta_new*gamma_new, sigma2_new, tau2_new, pi1_new)
  }

  # remove the first nr_burnin number of samples
  res[-seq(nr_burnin), ]
}
```

Let's fit the model using this Gibbs sampler and see posteriors of π_1 and τ^2.

```
set.seed(18)
ss = ss_regress(
  y, X, a1 = .01, a2 = .01, pi1 = .5,
  a = 1, b = 1, s = 1/2, nr_samples = 5000
  )
par(mfrow = c(1,2))
plot(density(ss[,"pi1"]), xlab = expression(pi[1]), lwd = 2,
     main = "Posterior of pi1", col = "red")
plot(density(ss[,"tau2"]), xlab = expression(tau^2), lwd = 2,
     main = "Posterior of tau2", col = "orange", xlim = c(0,1))
```




```
summary(ss[,"pi1"])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.008434 0.126940 0.184983 0.199423 0.258069 0.688194
```

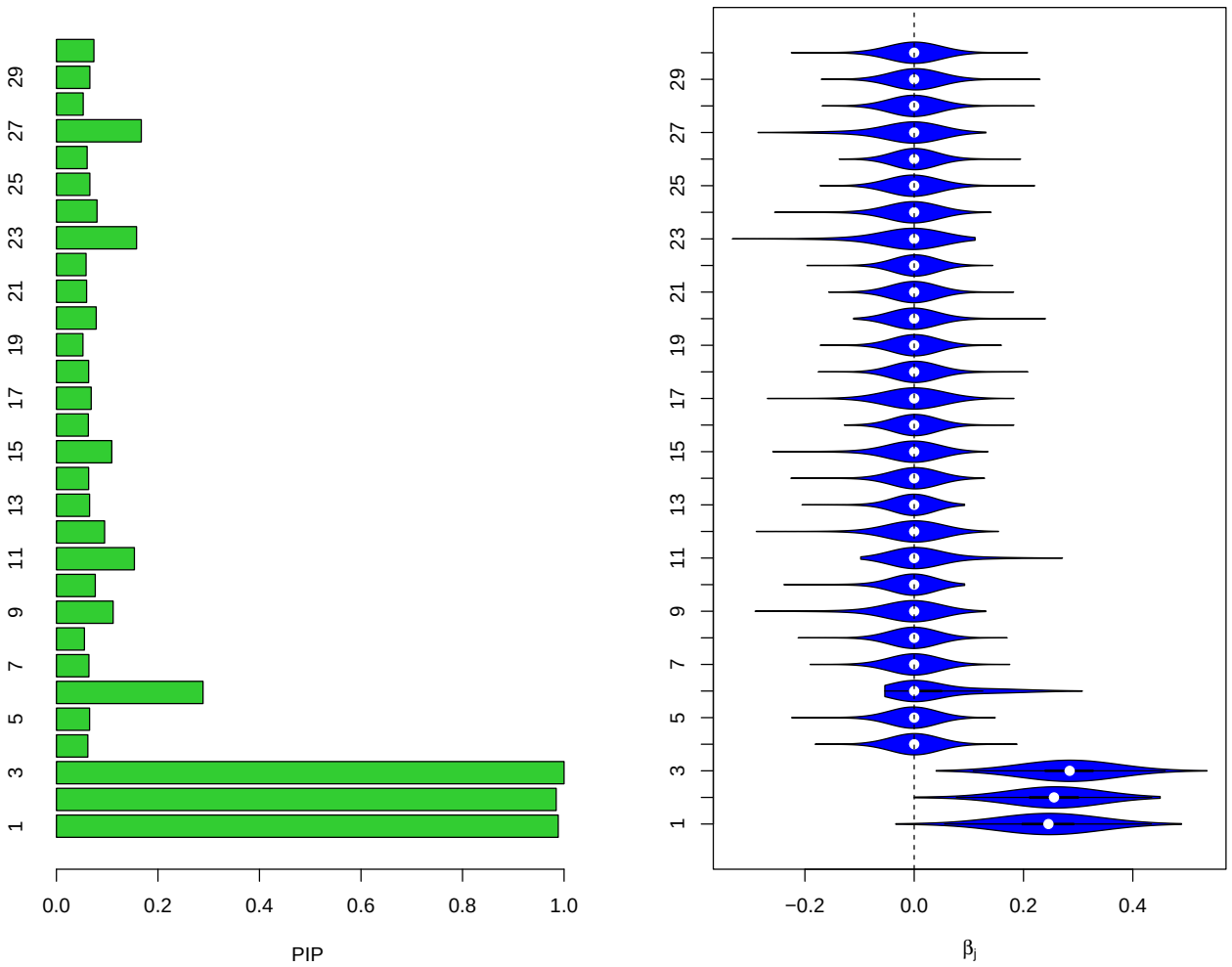
```
summary(ss[,"tau2"])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##   0.01785   0.05738   0.09008   0.14707   0.15038 14.88438
```

The posterior of π_1 is quite wide and suggest that π_1 is somewhere between 0.05 and 0.4. For τ^2, the bulk of the posterior is below 0.20.

We can also see the PIPs and coefficient estimates.

```
par(mfrow = c(1,2))
barplot(colMeans(ss[,1:p]), names.arg = 1:30,
        col = "limegreen", horiz = TRUE, xlab = "PIP")
library(vioplot)
vioplot( ss[,(p+1):(2*p)], names = 1:30,
         col = "blue", horizontal = TRUE, xlab = expression(beta[j]))
abline(v = 0, lty = 2)
```



Unfortunately, these implementation of Bayesian variable selection using Spike and Slab prior through Gibbs sampler do not scale to large data sets. First problem is that at each iteration of the Gibbs sampler matrix decomposition on $p \times p$ matrix related to $\boldsymbol{X}^T\boldsymbol{X}$ are done. Second problem is that at each iteration every

predictor is updated even when the model might be very sparse, i.e., only a few coefficients are non-zero. These issues become too demanding when p becomes large.

Let's next discuss some recent work that has made it possible to apply these kinds of models to very large p.

Efficient computation of Bayes factors under SSP

Let's continue with the assumption that each predictor has an independent SSP(π_1, τ^2) prior. It has been shown by Benner et al. that the Bayes factor (BF) comparing a model where the non-zero coefficients are specified by a configuration $\boldsymbol{\gamma}$ and the null model $\boldsymbol{\gamma}_0 = (0, \ldots, 0)$ where all coefficients are zero can be computed using data on only the non-zero variables (Details in Slides). This allows computing BF$(\boldsymbol{\gamma} : \boldsymbol{\gamma}_0)$ between $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}_0$ in $\mathcal{O}(k^3)$ operations, where $k = \sum_{j=1}^{p} \gamma_j$ is the number of non-zero coefficients in $\boldsymbol{\gamma}$, compared to $\mathcal{O}(p^3)$ operations that are required when the computation is done in the standard way. When we consider sparse models where $k \sim 10$ and $p \sim 10^5$ this can make all the difference.

Note that under SSP(π_1, τ^2) prior, the prior probability of a configuration $\boldsymbol{\gamma}$ is $\pi_1^{k_\gamma}(1-\pi_1)^{p-k_\gamma}$ and only depends on k_γ. Thus, by knowing the BFs for all possible configurations, we can compute an (unnormalized) posterior probabilities for the configurations as

$$\Pr(\boldsymbol{\gamma} \,|\, \text{Data}) \propto \text{BF}(\boldsymbol{\gamma} : \boldsymbol{\gamma}_0)\pi_1^{k_\gamma}(1-\pi_1)^{p-k_\gamma}.$$

By normalizing these quantities with respect to their sum over $\boldsymbol{\gamma}$, we have computed the posteriors for every configuration.

However, if we allowed, say, 10 non-zero variables out of 10^5 candidates, we would still have $\binom{10^5}{10} \approx 10^{43}$ configurations to evaluate, and that is far too many even when we can do every one of them quickly. For this reason, stochastic search algorithms have been introduced to approximate the posterior by finding the most relevant configurations from the whole space of configurations and then normalizing their un-normalized posteriors with respect to the relevant subset of configurations. (See slides for an example.)

Sum of single effect model (SUSIE)

Another recent approach to analyze Bayesian variable selection model is using the 'single-effect regression' (SER) model as a building block (Wang et al.). SER model is a multiple-regression model in which exactly one of the p predictors has a non-zero regression coefficient. The idea is to combine some number K of these SERs together to get a model where there are K non-zero predictors, or K *effects*. The key is that fitting each SER model conditional on the other SER models is very quick.

SUSIE model with K effects Let's assume that the number of non-zero effects K, the error variance σ^2 and the prior variance on the non-zero effect τ^2 are fixed. For each non-zero effect $k = 1, \ldots, K$, define $\boldsymbol{\gamma}^{(k)} \in \{0,1\}^p$ as the indicator vector that shows which is the kth non-zero coefficient, that is, $\sum_{j=1}^{p} \gamma_j^{(k)} = 1$. The model is

$$\begin{aligned}
\boldsymbol{y} &\sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{I}) \\
\boldsymbol{\beta} &= \sum_{k=1}^{K} \boldsymbol{\beta}^{(k)} \\
\boldsymbol{\beta}^{(k)} &= \beta^{(k)}\boldsymbol{\gamma}^{(k)}, \text{ for each } k = 1, \ldots, K, \\
\beta^{(k)} &\sim \mathcal{N}(0, \tau^2), \text{ for each } k = 1, \ldots, K, \\
\boldsymbol{\gamma}^{(k)} &\sim \text{Multinomial}(1, (1/p, \ldots, 1/p)), \text{ for each } k = 1, \ldots, K.
\end{aligned}$$

The last line means that the exactly one of the elements of $\boldsymbol{\gamma}^{(k)}$ is non-zero and, *a priori*, it is equally likely to be any one of the p elements.

Note that the model does not rule out the possibility that $\boldsymbol{\gamma}^{(k)} = \boldsymbol{\gamma}^{(\ell)}$ for some $k \neq \ell$ even though conceptually it would be natural to avoid such overlaps. However, by leaving a possibility for an overlap, we can treat each effect independently and we get much simpler computations. Additionally, in practice, we won't observe overlapping effects in the posterior distribution estimated by the following algorithm.

Iterative Bayesian stepwise selection (IBSS) A key motivation for the SUSIE model is that, given $(\boldsymbol{\beta}^{(\ell)})_{\ell \neq k}$ fitting $\boldsymbol{\beta}^{(k)}$ involves only fitting an SER model whose posterior distribution on the indicator $\boldsymbol{\gamma}^{(k)}$ and the parameter $\beta^{(k)}$ is simple to compute. This suggests an iterative algorithm (called IBSS) to fit the model. Below we apply it to our existing data set by allowing $K = 5$ effects and setting prior variance $\tau^2 = 1$.

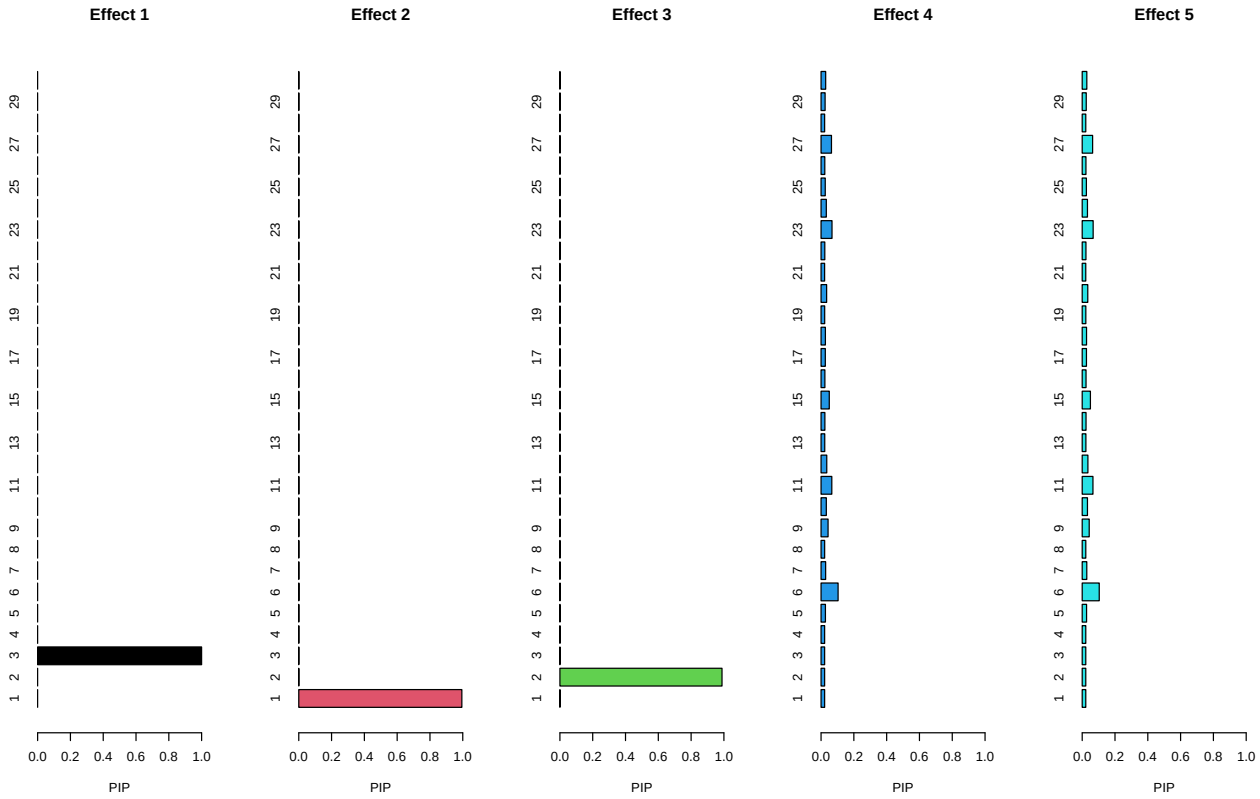
```
p = ncol(X)
n = nrow(X)
K = 5
tau2 = 1
xx = colSums(X^2) # = diagonal of t(X) %*% X. Will be needed later.
b = matrix(0, nrow = K, ncol = p) #initialize each effect beta_k = 0
res.b = res.pip = res.v = b #initialize results to 0
diff.in.post = rep(0,p) #initialize to 0
converged = FALSE
tol = 1e-6 # stop when difference between iterations < tol
iter = 0
while(!converged){
  iter = iter + 1
  for (k in 1:K){    #Fitting SER model for effect k

    r = as.vector(y - rowSums(X%*%t(b[-k,]))) #residuals without effect k

    #Use Exercise 1.5 results here to get MLE of linear model
    b.k = t(X) %*% r / xx #univariate OLS estimate
    B = matrix(b.k, byrow = T, nrow = n, ncol = p)
    M = ( X*B - r )^2
    v.k = colSums(M) / xx / (n-1)  #MLE for variance of b.k

    #Combine MLEs with prior distributions to get posteriors
    post.v.k = 1/(1/v.k + 1/tau2)
    post.b.k = post.v.k/v.k * b.k
    #Conditional posterior for beta_k is N(post.b.k, post.v.k)
    # conditional on the effect being at that position

    #PIPs for effect being at position 1,...,p:
    log.bf = 0.5*log(v.k/(v.k + tau2)) + 0.5*b.k^2/v.k*tau2/(tau2 + v.k)
    tmp = exp(log.bf - max(log.bf))
    post.ip = tmp / sum(tmp)

    b[k,] = post.b.k*post.ip #save new effect k estimate for next iteration
    #compare to previous iteration to observe convergence:
    diff.in.post[k] = max(abs(b[k,] - res.b[k,]*res.pip[k,]))
    #save results
    res.b[k,] = post.b.k
    res.pip[k,] = post.ip
    res.v[k,] = post.v.k
  }
```

```
  converged = (all(diff.in.post < tol))
}
print(paste("Ended in",iter,"iterations"))
```

```
## [1] "Ended in 8 iterations"
```

Let's examine the output of this algorithm. We allowed 5 non-zero effects. Let's see the distributions of PIPs for each effect $k = 1, \ldots, 5$.

```
par(mfrow = c(1,5))
for(ii in 1:K){
barplot(res.pip[ii,], names.arg = 1:p, horiz = TRUE, xlim = c(0,1),
        main = paste("Effect",ii), col = ii, xlab = "PIP")
}
```



We see that for the first 3 effects, there is a clear difference between predictors in their PIPs. For the last two, none of the coefficients has a large PIP and therefore we may want to ignore those effects. In other words, likely there are only 3 clear non-zero effects there and the remaining effects, if included in SUSIE model, wouldn't provide any information about which of the predictors would have a non-zero effect. Note, however, that despite this reasonable ad hoc criterion to determine the number of effects, the SUSIE model cannot make any probabilistic assessment of the number of non-zero effects whereas SSP model directly gave us the posterior on the number of non-zero coefficients.

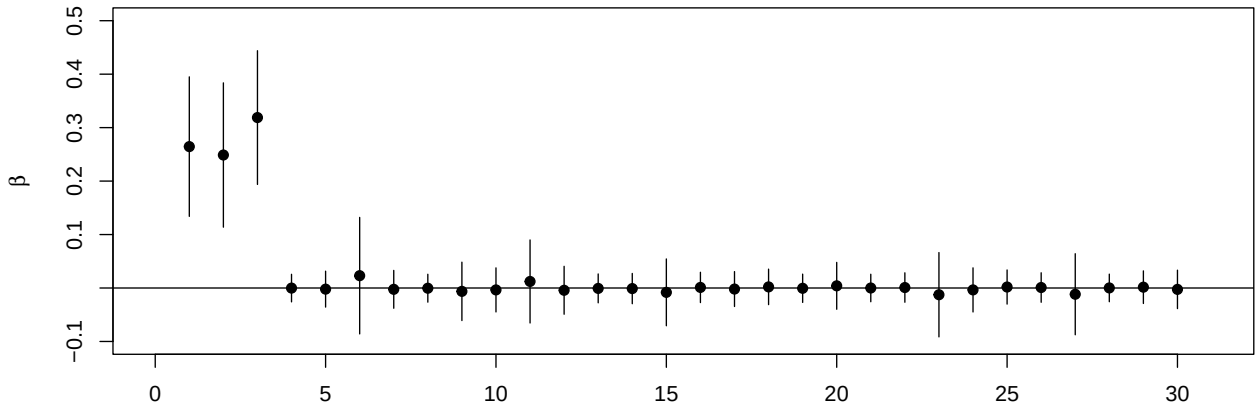
We can next estimate the coefficients by summing over K SER models. Let's denote the posterior mean and variance of the SER model k conditional on the effect being at predictor j by $\mu_j^{(k)}$ and $\eta_j^{(k)}$, respectively. Then we can compute the mean and variance for each coefficient (marginalized over all K SER models) as

follows.

$$
\begin{aligned}
\mathrm{E}(\beta_j) &= \mathrm{E}\left(\sum_{k=1}^{K}\gamma_j^{(k)}\beta_j^{(k)}\right) = \sum_{k=1}^{K}\pi_j^{(k)}\mu_j^{(k)} \\
\mathrm{Var}(\beta_j) &= \mathrm{Var}\left(\sum_{k=1}^{K}\gamma_j^{(k)}\beta_j^{(k)}\right) = \sum_{k=1}^{K}\mathrm{Var}\left(\gamma_j^{(k)}\beta_j^{(k)}\right) \\
&= \sum_{k=1}^{K}\left(\mathrm{E}\left(\gamma_j^{(k)^2}\beta_j^{(k)^2}\right) - \mathrm{E}\left(\gamma_j^{(k)}\beta_j^{(k)}\right)^2\right) \\
&= \sum_{k=1}^{K}\left(\pi_j^{(k)}\left(\eta_j^{(k)^2} + \mu_j^{(k)^2}\right) - \pi_j^{(k)^2}\mu_j^{(k)^2}\right) \\
&= \sum_{k=1}^{K}\left(\pi_j^{(k)}\left(\eta_j^{(k)^2}\right) + \pi_j^{(k)}(1-\pi_j^{(k)})\mu_j^{(k)^2}\right)
\end{aligned}
$$

Let's find the posterior means and variances and then show the 95% credible intervals assuming that posteriors are Gaussian.

```
b.mean = colSums(res.pip*res.b)
b.var = colSums(res.pip*res.v + res.pip*(1-res.pip)*res.b^2)
plot(NULL, xlim = c(0,p+1), ylim = c(-0.1, 0.5),
     ylab = expression(beta), xlab = "")
points(1:p, b.mean, pch = 19)
arrows(1:p, b.mean-1.96*sqrt(b.var), 1:p, b.mean+1.96*sqrt(b.var),
       code = 3, length = 0)
abline(h=0)
```



Credible sets of non-zero variables A nice property of SUSIE model is that it allows a straightforward computation of **credible set** of the non-zero variables.

Definition. A level ρ credible set (CS) is a subset of variables that has probability $\geq \rho$ of containing at least one effect variable (i.e. a variable with non-zero regression coefficient).

For each effect in SUSIE model, we can compute credible set by sorting the PIPs in decreasing order and by forming the CS by inlcuding the smallest possible number of variables whose PIPs sum to a value $\geq \rho$.

In our current example, 95% CS are below.

```
rho = 0.95
cs = list()
for(ii in 1:K){
  ord = order(res.pip[ii,], decreasing = T)
  cs[[ii]] = ord[1:min(which(cumsum(res.pip[ii,ord]) >= rho))]
}
cs
```

```
## [[1]]
## [1] 3
##
## [[2]]
## [1] 1
##
## [[3]]
## [1] 2
##
## [[4]]
##  [1]  6 23 11 27 15  9 12 20 24 10 30  7  5 18 17 25 29 16 14 26 22 13 19  8  1
## [26]  4 28  3
##
## [[5]]
##  [1]  6 23 11 27 15  9 12 20 24 10 30  7  5 18 17 25 29 16 14 26 22 13 19  8  1
## [26]  4 28  3
```

For the first three effects, the CSs only contain one variable each while the CSs for the last two effects contain almost all variables. This again shows that the first three effects are clearly localized each to one variable whereas the two additional effects are not informative and we might want to just ignore them.

Note that the posterior distribution provided by the SSP model does not directly contain information by which one could build credible sets for variables. Credible sets are useful when one needs to list the possible candidate variables that could replace each other in the model.

Question. Assume that there are three highly correlated variables of which one truly affects the outcome but no other variables associated with the outcome. What kind of PIPs SUSIE would give and how would the credible sets look like?