1. Consider the data \((y_i, x_i)_{i=1}^n\) is following a high dimensional probit regression model

\[
P(y_i = 1|x_i) = \Phi(x_i' \beta),
\]

where \(y_i \in \{0, 1\}\), \(x_i \in \mathbb{R}^p\), and \(\beta = (\beta_1, \ldots, \beta_p)'\). Consider \(p = 400\), \(n = 1000\).

(a) Simulate 5 datasets by drawing \(x_i \sim N(0, I)\) and \(\beta = (1.3, 4, -1, 1.6, 5, -2, 0_{394 \times 1})'\).

(b) Compare parameter estimates from MCMC and Sequential Monte Carlo (SMC).

(c) Provide the coverage and length of 95% credible intervals of \(\beta_1, \ldots, \beta_6\) for the MCMC and SMC.

I create two R lists to store five different simulated datasets for \(y_i \in \{0, 1\}\) and \(x_i, i = 1, \ldots, n\). Seeds for each dataset are fixed at 1000, 2000, 3000, 4000 and 5000. After simulate \(x_i, y_i\) is drawn from \(y_i \sim \text{Bernoulli}(\Phi(x_i' \beta))\).

**MCMC Method Discussion**

**Algorithm summary**

For the MCMC algorithm, I use Gibbs sampling proposed by Albert and Chib (1993) that introduce latent variables \(\{z_i\}_{i=1}^n\) and rewrite the probit model as

\[
z_i = x_i' \beta + \epsilon_i, \quad i = 1, \ldots, n
\]

\[
y_i = \begin{cases} 1, & z_i > 0 \\ 0, & z_i \leq 0 \end{cases}
\]

The full posterior is then

\[
p(\beta, z|y) \propto \prod_{i:y_i=1} N(z_i|x_i' \beta, 1)(z_i > 0) \prod_{i:y_i=0} N(z_i|x_i' \beta, 1)(z_i \leq 0) \pi(\beta),
\]

where \(1()\) is the indicator function and \(\pi(\beta)\) is the prior of \(\beta\).

With a conjugate normal prior \(\pi(\beta) \sim N(\beta|0, vI)\), \(v > 0\), full conditionals for the parameters can be derived:

\[
p(\beta|\cdots) = N(\beta|\mu_\beta, \Sigma_\beta),
\]

\[
p(z|\cdots) \propto \prod_{i:y_i=1} N(z_i|x_i' \beta, 1)(z_i > 0) \prod_{i:y_i=0} N(z_i|x_i' \beta, 1)(z_i \leq 0),
\]

\[
p(\cdots) \propto \prod_{i:y_i=1} N(z_i|x_i' \beta, 1)(z_i > 0) \prod_{i:y_i=0} N(z_i|x_i' \beta, 1)(z_i \leq 0),
\]
where $\mu_\beta = \Sigma_\beta' X z$, $X = (x'_1, \ldots, x'_n)'$, $\Sigma_\beta = (X'X + v^{-1}I)^{-1}$. Notice that the full conditional of $z$ is either one of two Truncated Normal (TN) depending on $y_i = 1$ or 0.

This algorithm works well when the number of parameters $p$ is small, but it may still have some slow mixing issue due to strong posterior correlation between $\beta$ and $z$ and high autocorrelation of draws. The MCMC performance for inferencing $\beta$ is getting worse as $p$ increases. More interestingly, the prior variance $v$ that controls how informative the prior is will act as a tuning parameter that generates very different convergence results for its different values. Hence, the posterior result in a high dimensional setting is very sensitive to the hyperparameters of the prior distribution. I will provide simulation results to justify these findings.

Small $p$ case

First, if $p = 6$ and the coefficients are the first six nonzero ones, the MCMC algorithm performs quite well. With $v = 100$, after burning 1000 samples and thinning the sequences by keeping every 100th draw, 1000 samples are saved for analysis. Table 1 shows MCMC summary about posterior mean, lower and upper bound of the 95% credible interval and its length as well as effective size of the MCMC sample. All of these measures are averaged over the simulation results from the five simulated datasets. All true $\beta$s are in their corresponding interval. In addition, larger values of coefficients tend to have longer length of credible interval. Although not shown here, trace plots, autocorrelation functions (ACFs), and histograms of the coefficients all look fine.

<table>
<thead>
<tr>
<th></th>
<th>Post.mean</th>
<th>Lwr.bound</th>
<th>Upr.bound</th>
<th>95% Length</th>
<th>Eff.size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_1$</td>
<td>1.40</td>
<td>1.09</td>
<td>1.74</td>
<td>0.66</td>
<td>182.91</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>4.26</td>
<td>3.53</td>
<td>5.09</td>
<td>1.55</td>
<td>112.70</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>-1.11</td>
<td>-1.41</td>
<td>-0.84</td>
<td>0.57</td>
<td>221.24</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>1.59</td>
<td>1.24</td>
<td>1.97</td>
<td>0.73</td>
<td>169.75</td>
</tr>
<tr>
<td>$\beta_5$</td>
<td>5.37</td>
<td>4.47</td>
<td>6.41</td>
<td>1.94</td>
<td>110.65</td>
</tr>
<tr>
<td>$\beta_6$</td>
<td>-2.08</td>
<td>-2.53</td>
<td>-1.69</td>
<td>0.85</td>
<td>147.44</td>
</tr>
</tbody>
</table>

Table 1: MCMC summary when $p = 6$.

Large $p$ case

Back to $p = 400$. We check how posterior distributions behave differently from different values of $v$. With the same burn-in and thinning strategy as $p = 6$ case, 1000 draws are kept for analysis. Table 2 and 3 show the MCMC summary of first six nonzero $\beta$s when $v = 1$ and $v = 0.25$, respectively. All numbers are averaged from the five simulated datasets. Their corresponding trace plot, ACF and histograms are shown in Figure 1 and 2. With all initial values and hyperparameters being equal, different $v$s show different convergence results. When $v = 1$, the posterior estimates are away from the true ones, leading to bad inference in this high dimensional setting. When $v = 0.25$, the inference result is much better than the result with $v = 1$, whose posterior means are close to the true values with shorter 95% credible interval and larger effective size. Basically, as $v$ increases, the performance is
Table 2: MCMC summary of first six nonzero $\beta$s when $p = 400$ and $v = 1$.

<table>
<thead>
<tr>
<th>Post.mean</th>
<th>Lwr.bound</th>
<th>Upr.bound</th>
<th>95% Length</th>
<th>Eff.size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_1$</td>
<td>2.70</td>
<td>1.89</td>
<td>3.54</td>
<td>1.65</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>8.77</td>
<td>7.75</td>
<td>9.83</td>
<td>2.09</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>-2.16</td>
<td>-3.08</td>
<td>-1.28</td>
<td>1.80</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>2.93</td>
<td>2.02</td>
<td>3.88</td>
<td>1.86</td>
</tr>
<tr>
<td>$\beta_5$</td>
<td>11.41</td>
<td>10.33</td>
<td>12.49</td>
<td>2.16</td>
</tr>
<tr>
<td>$\beta_6$</td>
<td>-4.44</td>
<td>-5.39</td>
<td>-3.55</td>
<td>1.84</td>
</tr>
</tbody>
</table>

Table 3: MCMC summary of first six nonzero $\beta$s when $p = 400$ and $v = 0.25$.

<table>
<thead>
<tr>
<th>Post.mean</th>
<th>Lwr.bound</th>
<th>Upr.bound</th>
<th>95% Length</th>
<th>Eff.size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_1$</td>
<td>1.35</td>
<td>0.92</td>
<td>1.80</td>
<td>0.88</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>4.35</td>
<td>3.86</td>
<td>4.87</td>
<td>1.02</td>
</tr>
<tr>
<td>$\beta_3$</td>
<td>-1.06</td>
<td>-1.54</td>
<td>-0.60</td>
<td>0.93</td>
</tr>
<tr>
<td>$\beta_4$</td>
<td>1.46</td>
<td>0.99</td>
<td>1.92</td>
<td>0.93</td>
</tr>
<tr>
<td>$\beta_5$</td>
<td>5.66</td>
<td>5.11</td>
<td>6.23</td>
<td>1.12</td>
</tr>
<tr>
<td>$\beta_6$</td>
<td>-2.21</td>
<td>-2.67</td>
<td>-1.76</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Figure 1: Trace, ACF and histograms of first nonzero betas of the third simulated dataset when $p = 400$ and $v = 1$

getting worse. $v$ is like a tuning parameter that we need to guess to get a good inference result.
I am wondering if it is due to high dimension itself or due to its sparseness. Hence, instead of assuming all other 394 coefficients being zero, I now assume they all have value one. Under this setting, Table 4 and 5 tell us that the model with $v = 0.25$ shrinks coefficients too much toward zero. $v = 2$ become more reasonable. If all other 394 coefficients are set to be 3, although its result is not shown here, the story will again be different.\footnote{Using $v = 2$ now shrinks coefficients too much. Larger value of $v$ is used and leads to better posterior estimation, but the credible length is wider.} Therefore, this high dimensional problem is very sensitive to how we specify the prior information. This high sensitivity is problematic because in real applications, we never know what the truth is.

**SMC Method Discussion**

**Algorithm summary**

My SMC algorithm is based on Chopin (2002). It is summarized in Algorithm 1. Here, I assume for $i = 1, \ldots, q,$

\[ p(y_{n+1:n+q}|y_{1:n}, \beta_j) = \prod_{i=1}^{q} p(y_{n+i}|\beta_j) = \Phi(\mathbf{x}_{n+i}'\beta_j)^{y_{n+i}}\Phi(-\mathbf{x}_{n+i}'\beta_j)^{1-y_{n+i}}, \]

Figure 2: Trace, ACF and histograms of first nonzero *betas* of the third simulated dataset when $p = 400$ and $v = 0.25$
<table>
<thead>
<tr>
<th>Post. mean</th>
<th>Lwr. bound</th>
<th>Upr. bound</th>
<th>95% Length</th>
<th>Eff. size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_1 )</td>
<td>0.76</td>
<td>0.26</td>
<td>1.23</td>
<td>0.97</td>
</tr>
<tr>
<td>( \beta_2 )</td>
<td>1.21</td>
<td>0.75</td>
<td>1.69</td>
<td>0.94</td>
</tr>
<tr>
<td>( \beta_3 )</td>
<td>-0.18</td>
<td>-0.67</td>
<td>0.31</td>
<td>0.98</td>
</tr>
<tr>
<td>( \beta_4 )</td>
<td>0.76</td>
<td>0.30</td>
<td>1.22</td>
<td>0.93</td>
</tr>
<tr>
<td>( \beta_5 )</td>
<td>2.00</td>
<td>1.53</td>
<td>2.49</td>
<td>0.96</td>
</tr>
<tr>
<td>( \beta_6 )</td>
<td>-0.46</td>
<td>-0.88</td>
<td>0.02</td>
<td>0.90</td>
</tr>
<tr>
<td>( \beta_7 )</td>
<td>0.29</td>
<td>-0.16</td>
<td>0.75</td>
<td>0.91</td>
</tr>
<tr>
<td>( \beta_8 )</td>
<td>0.29</td>
<td>-0.16</td>
<td>0.73</td>
<td>0.88</td>
</tr>
<tr>
<td>( \beta_9 )</td>
<td>0.45</td>
<td>-0.01</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td>( \beta_{10} )</td>
<td>0.41</td>
<td>-0.07</td>
<td>0.91</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Table 4: MCMC summary of first 10 \( \beta \)s when \( p = 400 \) and \( v = 0.25 \) assuming \( \beta_7 = \cdots = \beta_{400} = 1 \).

<table>
<thead>
<tr>
<th>Post. mean</th>
<th>Lwr. bound</th>
<th>Upr. bound</th>
<th>95% Length</th>
<th>Eff. size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_1 )</td>
<td>2.11</td>
<td>0.75</td>
<td>3.42</td>
<td>2.67</td>
</tr>
<tr>
<td>( \beta_2 )</td>
<td>3.43</td>
<td>2.32</td>
<td>4.63</td>
<td>2.31</td>
</tr>
<tr>
<td>( \beta_3 )</td>
<td>-0.48</td>
<td>-1.80</td>
<td>0.76</td>
<td>2.56</td>
</tr>
<tr>
<td>( \beta_4 )</td>
<td>2.20</td>
<td>0.96</td>
<td>3.30</td>
<td>2.34</td>
</tr>
<tr>
<td>( \beta_5 )</td>
<td>5.60</td>
<td>4.36</td>
<td>6.91</td>
<td>2.54</td>
</tr>
<tr>
<td>( \beta_6 )</td>
<td>-1.36</td>
<td>-2.57</td>
<td>-0.22</td>
<td>2.35</td>
</tr>
<tr>
<td>( \beta_7 )</td>
<td>0.82</td>
<td>-0.38</td>
<td>2.23</td>
<td>2.61</td>
</tr>
<tr>
<td>( \beta_8 )</td>
<td>0.88</td>
<td>-0.30</td>
<td>2.12</td>
<td>2.42</td>
</tr>
<tr>
<td>( \beta_9 )</td>
<td>1.27</td>
<td>0.07</td>
<td>2.53</td>
<td>2.46</td>
</tr>
<tr>
<td>( \beta_{10} )</td>
<td>1.11</td>
<td>-0.12</td>
<td>2.28</td>
<td>2.41</td>
</tr>
</tbody>
</table>

Table 5: MCMC summary of first 10 \( \beta \)s when \( p = 400 \) and \( v = 2 \) assuming \( \beta_7 = \cdots = \beta_{400} = 1 \).

**Algorithm 1:** Iterated Batch Importance Sampling of Probit Regression Model

**Result:** Posterior sample of parameters \( \beta_1, \ldots, \beta_{400} \).

1. **Initialization:** Generate \( H \) particles \( \{ \theta_j \}_{j=1}^H \) from prior distribution \( N(0, \alpha I) \);
2. **Reweighting:** Update the weights, \( w_j \leftarrow w_j \times p(y_{n+1:n+q} | y_{1:n}, \theta_j) \);
3. **Resampling:** Resample \( \{ \theta_j, w_j \}_{j=1}^H \) by \( \{ \theta_j^{\ast}, w_j \}_{j=1}^H \) according to multinomial distribution with equal probability vector \( w_1, \ldots, w_H \);
4. **Move (Rejuvenation):**
   
   **for** \( j = 1, 2, \ldots, H \) **do**
      
   **draw** \( \theta_j^{\ast} \) from \( K_{n+q}(\theta_j^{\ast}, \cdot) \), and set \( \theta_j^m = \theta_j^{\ast} \) if \( U < \rho \), \( \theta_j^m = \theta_j^r \) otherwise, where
   
   \( U \sim \text{unif}(0, 1) \) and \( \rho = \min\{ \frac{\pi_{n+q}(\theta_j^m)}{\pi_{n+q}(\theta_j^r)}, 1 \} \)

**end**

Here, \( K_{n+q} \) is a transition kernel with stationary distribution \( \pi_{n+q} \);
5. **Loop:** \( n \leftarrow n + q, \{ \beta_j, w_j \}_{j=1}^H \leftarrow \{ \beta_j^m, w_j \}_{j=1}^H \) and return to step 2. The algorithm stops when \( n = N \), the sample size of data.

and

\[ K_{n+q} = N(\mu, \Sigma), \]
Figure 3: Histograms of first six $\beta$s using SMC with $H = 200$, $q = 5$, $v = 0.01$.

where

$$\mu = \frac{\sum_{j=1}^{H} w_j \beta_j}{\sum_{j=1}^{H} w_j}$$

$$\Sigma = \frac{\sum_{j=1}^{H} w_j (\beta_j - \mu)(\beta_j - \mu)^t}{\sum_{j=1}^{H} w_j}.$$

The problem I face in this algorithm is that the particles degenerate so quickly that the distribution of $\beta$ cannot be fully described. Even in some cases that the degeneracy is not so severe, the estimation is not good. Figure 3 tells us how poor the performance is.

In fact, how fast particles degenerate depends on how many observations per batch of data, i.e., the vlaues of $q$ and the variance of the prior distribution of $\beta$, which is $v$. From my experience, the larger value of $q$ and $v$, the faster particles will degenerate. It ends up using very strong prior and update with a small bunch of observations at a time in order to make the algorithm work.

```r
smc_probit <- function (Y, X, H, pp, v) {
XtX = t(X) %*% X
p = ncol(X)
N = nrow(X)
w = rep(1, H)
#-------- step 0: Initialization ---------#
# generate totally H particles of beta_1, ..., beta_p from prior
Beta = matrix(rnorm(p*H, 0, sqrt(v)), nr = H, nc = p)
accept = rep(0, N/pp)
```
for(t in 1:(N/pp)) {
  np = t * pp
  #-------- step 1: Reweighting --------#
  # decide which batch of data
  batch.idx = ((t - 1) * pp + 1) : (t * pp)
  idx.1 <- which(Y[batch.idx] == 1)
  idx.0 <- which(Y[batch.idx] == 0)
  sumloglik <- function(x, beta, idx.0, idx.1) {
    # for one beta_j
    sum(pnorm(x[idx.1, ] %*% beta, log.p = TRUE)) +
    sum(pnorm(x[idx.0, ] %*% beta, lower.tail = FALSE, log.p = TRUE))
  }
  # log sum of likelihood for j = 1, ..., H
  sumlik = apply(Beta, 1, sumloglik, x = X, idx.0 = idx.0, idx.1 = idx.1)
  logw = log(w) + sumlik
  w = exp(logw)

  #-------- step 2: Resampling --------#
  Beta.re = Beta[sample(1:H, H, replace = TRUE, prob = w), ]

  #-------- step 3: Move(rejuvenation) --------#
  # Use Normal kernel
  kernel <- function(Beta.re, w) {
    sum_w = sum(w)
    mu = apply(w * Beta.re, 2, sum) / sum_w
    # matrix to list by row
    Betalist = lapply(seq_len(nrow(Beta.re)), function(i) Beta.re[i, ])
    # list of covariance matrix
    siglist = lapply(Betalist, function(Beta.re, mu) (Beta.re - mu) %o% (Beta.re - mu), mu = mu)
    # times weight for each j
    bbb = mapply("*", siglist, w, SIMPLIFY = FALSE)
    # sum of all weighted covariance matrix
    V = Reduce("+", bbb)
    sigma = (V / sum_w) + 0.01 * diag(p)
    chol.sig = t(chol(sigma))
    mu + chol.sig %*% rnorm(p)
  }
  idx.1 <- which(Y[1:np] == 1)
  idx.0 <- which(Y[1:np] == 0)
  Beta.m = matrix(NA, nr = H, nc = p)
  for(j in 1:H) {
    cat("t =", t,"j =", j,"\r")
    beta.star = kernel(Beta.re, w)
    loglik.star = sumloglik(X, beta.star, idx.0, idx.1) +
    sum(dnorm(beta.star, 0, sqrt(v), log = TRUE))
    loglik.old = sumloglik(X, Beta.re[j, ], idx.0, idx.1) +
    sum(dnorm(Beta.re[j, ], 0, sqrt(v), log = TRUE))
    u <- runif(1)
    if (log(runif(1)) < (loglik.star - loglik.old)) {
      Beta.m[j, ] <- beta.star
      accept[t] <- accept[t] + 1
    } else {
      Beta.m[j, ] <- Beta.re[j, ]
    }
  }

  #-------- step 4: Loop --------#
  Beta = Beta.m
}
return(Beta)

2. Suppose

\[ y_i = x_{1i} + (x_{2i} - 0.5)_+ + x_{3i}^2 + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2), \]

for \( i = 1, \ldots, 5000 \). Consider \( \sigma^2 = 0.5 \) and \( x_{1i}, x_{2i}, x_{3i} \sim N(0, 1) \).

(a) Use the above model to generate 5 datasets.
Figure 4: Scatter plot of $y$ versus $x_1$, $x_2$ and $x_3$ of first simulated data for predictive process model

(b) Run a predictive process model with the above data by writing your own code.
(c) Comment on the model fit in terms of estimating the mean function and error variance.

**Data and Predictive Process Model (PPM)**

Five simulated datasets are generated using random seed number $1000 \times k$, $k = 1 \ldots 5$. Our goal is to estimate the mean function $\mu(x_i) = x_{1i} + (x_{2i} - 0.5)_+ + x_{3i}^2$ by $m$ knots, $(x_{1j}^*, x_{2j}^*, x_{3j}^*)_{j=1}^m$, each of which is drawn from $N(0,1)$. Figure 4 gives us marginal relationship between $y$ and $x$, $k = 1, 2, 3$.

Let $s_i = (x_{1i}, x_{2i}, x_{3i})$ and $s_j^* = (x_{1j}^*, x_{2j}^*, x_{3j}^*)$. The PPM is defined as

$$ y = H \mu^* + \epsilon, \quad \epsilon \sim N(0, \sigma^2 I) $$

where $H = C(K^*)^{-1}$, $C_{n \times m} = (C(s_1), \ldots, C(s_m))'$, $C(s_i)_{1 \times m} = \text{Cov}[\mu(s_i), \mu^*] = \text{Cov}[\mu(s_i), \mu(s^*_i), \ldots, \mu(s^*_m)]$, $K = \text{Cov}[\mu^*, \mu^*] = \text{Cov}[(\mu(s^*_1), \ldots, \mu(s^*_m)), (\mu(s^*_1), \ldots, \mu(s^*_m))]$ with some covariance function $\text{Cov}[\mu(s), \mu(s')] = c(s, s'; \tau, \phi)$.

**PPM MCMC algorithm**

After integrating out $\mu^*$, the posterior distribution can be written as

$$ \pi(\sigma^2, \tau^2, \phi | y) \propto N(y|0, \sigma^2 I + HK^*H')\pi(\sigma^2, \tau^2, \phi) $$

With independent prior $\pi(\sigma^2) \sim IG(\alpha_{\sigma}, \beta_{\sigma})$, $\pi(\tau^2) \sim IG(\alpha_{\tau}, \beta_{\tau})$ and $\pi(\phi) \sim U(a_{\phi}, b_{\phi})$, the posterior can be expressed as

$$ \pi(\sigma^2, \tau^2, \phi | y) \propto |\sigma^2 I + HK^*H'|^{-1/2} \exp\{(-0.5)y'(\sigma^2 I + HK^*H')^{-1}y\}(\sigma^2)^{-(\alpha_{\sigma} + 1)}\exp\{-b_{\phi}/\sigma^2\} \exp\{-b_{\tau}/\tau^2\} \frac{1}{b_{\phi} - a_{\phi}} 1(\phi \in (a_{\phi}, b_{\phi})) $$

After change of variables $\eta = \log \sigma^2$, $\theta = \log \tau^2$ and $w = \log \left(\frac{\phi - a_{\phi}}{b_{\phi} - \phi}\right)$, the transformed log-posterior is

$$ \log \pi(\eta, \theta, w | y) \propto (-0.5) \left[ \log |e^y I + HK^*H'| + y'(e^y I + HK^*H')^{-1}y \right] - (\alpha_{\sigma})\eta - b_{\sigma}/e^\eta - (\alpha_{\tau})\theta - b_{\tau}/e^\theta + w - 2 \log(1 + e^w) $$

Let $\xi = (\eta, \theta, w)$. The Metropolis-Hastings step is shown in Algorithm 2.
Algorithm 2: M-H step of PPM

for $m = 1, 2, \ldots$ do

1. Draw $\xi^* \sim N(\xi^{(m-1)}, vI)$ for $m < 500$ and $\xi^* \sim N(\xi^{(m-1)}, S)$ if $m \geq 500$, $S$ is the posterior sample covariance using $\xi^{(1)}, \ldots, \xi^{(m-1)}$;
2. Set $\xi^{(m)} = \xi^*$ if $U \sim U(0, 1) < \rho = \min \left\{ \frac{\log \pi(\xi^* | y)}{\log \pi(\xi^{(m-1)} | y)}, 1 \right\}$; $\xi^{(m)} = \xi^{(m-1)}$ otherwise

end

---

PPM MCMC result

I save time to run my code, I consider $n = 1500$ and $m = 100$. The basic idea should be the same as $n = 5000$ case.  

First, 100 knots $s^*_j = (x^*_1, x^*_2, x^*_3)$ are generated from standard normal with `set.seed(12345)`. I use $\pi(\sigma^2) \sim IG(2, 1)$, $\pi(\tau^2) \sim IG(2, 1)$ and $\pi(\phi) \sim U(0.0001, 10)$ as my priors. For each data set, 11000 MCMC samples are drawn. I burn the first 1000 samples and keep every 10th simulation draw to get the final MCMC posterior sample of size 1000. Initial values of MCMC iterations are all set to be one for $\sigma^2$, $\tau^2$ and $\phi$. The convergence result of the third simulated data is shown in Figure 5. One can see that the variation of $\sigma^2$ and $\phi$ are

---

2I tried $n = 5000$ and $m = 250$ on one dataset and its posterior results look quite good, but I don’t have time to run this high dimensional case on all five datasets since I found my bugs of my code and made my code run correctly so late.
quite small while $\sigma^2$ has larger variation.\footnote{$\sigma^2$ is a bit away from the true value 0.5 in $n = 1500$ and $m = 100$ case, but the posterior mean is very close to 0.5 with small variation when I use $n = 5000$ and $m = 250$ on the first simulated data.} The average of posterior mean of $\sigma^2$ over five simulated datasets is 0.59, and the 95% credible interval is (0.55, 0.63) as shown in Table 6. Different values of $m$ are also used to check the effect on estimation. We can see in Table 6 that the performance of estimation is getting better as $m$ increases, though the price we pay is heavier computational burden.

<table>
<thead>
<tr>
<th>$m$</th>
<th>Post.mean</th>
<th>Lwr.bound</th>
<th>Upr.bound</th>
<th>95% Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.59</td>
<td>0.55</td>
<td>0.63</td>
<td>0.08</td>
</tr>
<tr>
<td>200</td>
<td>0.52</td>
<td>0.48</td>
<td>0.57</td>
<td>0.09</td>
</tr>
<tr>
<td>500</td>
<td>0.50</td>
<td>0.46</td>
<td>0.53</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Table 6: MCMC summary of posterior sample of $\sigma^2$.

Although $n = 5000$ and $m = 250$ case is not full analyzed, I feel that the PPM will estimate error variance pretty well in terms of unbiaesness and/or variation if sample size and knot size are large.

To estimate the mean function $\mu(x_i) = x_{1i} + (x_{2i} - 0.5) + x_{3i}$, for each posterior sample $\xi^{(m)}$, $m = 1, \ldots, M$, I first sample $\mu_{(m)}^* | y$ from $N(0, K(\xi^{(m)}))$ and then use $C(\xi^{(m)})(K(\xi^{(m)}))^{-1} \mu_{(m)}^*$ to recover $\mu(x)^{(m)}$. Figure 6 shows the estimated mean function for all five simulated datasets. The purple one is the 100th realization and gold and green ones are upper and lower bound of 95% interval. The estimated mean function captures the variation of the true mean function quite well. Most values are between the lower and upper bound.
Figure 6: Black: True mean function; Purple: Estimated mean function of the 100th MCMC sample; Gold: 97.5th percentile mean function of the MCMC sample; Green: 2.5th percentile mean function of the MCMC sample