Robust Statistical Modeling for Quantifying Periodontal Disease: A Single Index Mixed-Effects Approach with Skewed Random Effects and Heavy-Tailed Residuals

Contents

1 Introduction

1.1 The ST-GP Model

We propose a single index model with skewed random effects and heavy-tailed residuals. We refer to this model as the ST-GP model because the random effects and residuals jointly follow the ST distribution, and we apply the constrained Gaussian process (GP) prior from Maatouk and Bay (2017) on the index function.

Let $\mathbf{Y}_i^P = \left(Y_{i,1}^P, Y_{i,2}^P, \ldots, Y_{i,n_i}^P\right)^\top$ and $\mathbf{Y}_i^C = \left(Y_{i,1}^C, Y_{i,2}^C, \ldots, Y_{i,n_i}^C\right)^\top$ be the measurements of PD and CAL (in millimeter) for subject $i = 1, \ldots, N$. Here n_i denotes the number of teeth accounted for within the mouth for i-th subject. At the subject level, we propose a single index model with skewed random effects and heavy-tailed residuals as:

$$\mathbf{Y}_i = \begin{pmatrix} \mathbf{Y}_i^P \\ \mathbf{Y}_i^C \end{pmatrix} = \begin{pmatrix} g\left(\mathbf{X}_i\boldsymbol{\beta}\right) \\ a \times g\left(\mathbf{X}_i\boldsymbol{\beta}\right) \end{pmatrix} + \begin{pmatrix} \mathbf{1}_{n_i} \\ \mathbf{1}_{n_i} \end{pmatrix} b_i + \begin{pmatrix} \boldsymbol{\epsilon}_i^P \\ \boldsymbol{\epsilon}_i^C \end{pmatrix}, \tag{1}$$

with

$$g\left(\mathbf{X}_i\boldsymbol{\beta}\right) = \begin{pmatrix} g^\star\left(\mathbf{X}_i^{(1)}\boldsymbol{\beta}\right) \\ \vdots \\ g^\star\left(\mathbf{X}_i^{(n_i)}\boldsymbol{\beta}\right) \end{pmatrix},$$

where $\mathbf{X}_i^{(1)}$ and $\mathbf{X}_i^{(n_i)}$ represent the first and last row of $\mathbf{X}_i$, respectively. The slope parameter $a \in (-\infty, \infty)$ differentiates the fixed effects between PD and CAL, motivated by their observed correlation. The function $g^\star(\cdot)$ is assumed to be a continuous monotonic increasing function on its support $[-1, 1]$, with the constraint that $g^\star(-1) = 0$. For the identifiability concern, the L_2 norm of $\boldsymbol{\beta}$ must be 1. As the support of $g^\star(\cdot)$ is defined $[-1, 1]$, one need to scale $\mathbf{X}_i$ such that each row of $\mathbf{X}_i$ has L_2 norm no larger than 1.

The distributional assumption for the random effects and errors is expressed as follows:

$$\begin{pmatrix} b_i \\ \boldsymbol{\epsilon}_i \end{pmatrix} \sim \mathrm{ST}_{2n_i+1}\left[\begin{pmatrix} h(\nu)\delta \\ \mathbf{0}_{2n_i} \end{pmatrix}, \begin{pmatrix} d^2 & \mathbf{0}_{2n_i}^\top \\ \mathbf{0}_{2n_i} & \sigma^2\mathbf{I}_{2n_i} \end{pmatrix}, \begin{pmatrix} \delta \\ \mathbf{0}_{2n_i} \end{pmatrix}, \nu\right], \tag{2}$$

where

$$\boldsymbol{\epsilon}_i = \begin{pmatrix} \boldsymbol{\epsilon}_i^P \\ \boldsymbol{\epsilon}_i^C \end{pmatrix},$$

$$h(\nu) = -\sqrt{\nu/\pi}\,\Gamma\left(0.5\nu - 0.5\right)/\Gamma\left(0.5\nu\right),$$

$\Gamma(\cdot)$ represents the Gamma function, d^2 and σ^2 represent the conditional variance of the random effects and residuals, respectively, $\delta \in (-\infty, \infty)$ is the skewness parameter, and ν is the degree of freedom. Definitions and properties of the ST distribution are discussed in Appendix A. If one applies the constrained GP prior on the index function g, then the model described in (1) and (2) is referred to as the ST-GP model.

Additionally, using Proposition 5 of Schumacher et al. (2021), we have

$$\mathbf{Y}_i \sim \mathrm{ST}_{2n_i}\left(\boldsymbol{\theta}_i + h(\nu)\delta\mathbf{1}_{2n_i}, \boldsymbol{\Psi}_i, \delta\mathbf{1}_{2n_i}, \nu\right), \tag{3}$$

where

$$\boldsymbol{\theta}_i = \begin{pmatrix} g\left(\mathbf{X}_i\boldsymbol{\beta}\right) \\ a \times g\left(\mathbf{X}_i\boldsymbol{\beta}\right) \end{pmatrix}$$

and

$$\boldsymbol{\Psi}_i = d^2\mathbf{1}_{2n_i}\mathbf{1}_{2n_i}^\top + \sigma^2\mathbf{I}_{2n_i \times 2n_i}$$

represents a covariance matrix characterized by a compound symmetry structure, with readily available closed-form expressions for its inverse and determinant.

Last, the model in (1) and (2) has a hierarchical representation that facilities an associated Gibbs sampler. The details of the hierarchical representation and the associated Gibbs sampler can be found in Appendix B and C, respectively.

1.2 The Constrained GP Prior

To apply the constrained GP prior, we need to add one more assumption on $g^\star(\cdot)$. The support of $g^\star(\cdot)$ is restricted to $[-1, 1]$. Furthermore, grounded in our observation that utilizing a random intercept alone is adequate for the analysis of the real data, we add one more condition: $g^\star(-1) = 0$. This condition aligns with the reality of periodontal disease research, where the readings of PD and CAL must be non-negative. Therefore, assuming that the single index function is non-negative is reasonable. We summarize the assumption imposed on $g^\star(x)$ as follows: $g^\star(x)$ is defined as a continuous, monotonic increasing function on its support $[-1, 1]$, with the minimal value defined as $g^\star(-1) = 0$.

With these assumptions in place, we can now proceed to introduce the associated basis functions, $h_k(\cdot)$ and $\phi_k(\cdot)$, associated with the constrained GP prior. For given knots $-1 = u_0 < u_1 < \cdots < u_L = 1$, continuous piecewise linear functions are defined as, for $k = 1, \ldots, L$,

$$h_k(x) = \begin{cases} 0 & \text{if } x > u_{k+1} \text{ or } x < u_{k-1} \\ 1 & \text{if } x = u_k \\ \text{linear} & \text{otherwise} \end{cases}.$$

Taking integration of $h_k(x)$ on $(-1, x)$, we define $\psi_k(\cdot)$ as

$$\psi_k(x) = \int_{-1}^{x} h_k(t)dt.$$

Next, we define $\phi_k(\mathbf{X}_i\boldsymbol{\beta})$ as a vector-valued function consisting of n_i continuous piecewise linear functions:

$$\phi_k(\mathbf{X}_i\boldsymbol{\beta}) = \begin{pmatrix} \psi_k\left(\mathbf{X}_i^{(1)}\boldsymbol{\beta}\right) \\ \vdots \\ \psi_k\left(\mathbf{X}_i^{(n_i)}\boldsymbol{\beta}\right) \end{pmatrix}.$$

Finally, we can define the constrained GP prior and the index function as follows:

$$g(\mathbf{X}_i\boldsymbol{\beta}) = \boldsymbol{\Phi}\boldsymbol{\xi}, \tag{4}$$

where $\boldsymbol{\Phi}$ is a $n_i \times (L+1)$ matrix:

$$\begin{aligned} \boldsymbol{\Phi} &= \begin{pmatrix} \phi_0(\mathbf{X}_i\boldsymbol{\beta}) & \cdots & \phi_L(\mathbf{X}_i\boldsymbol{\beta}) \end{pmatrix} \\ &= \begin{pmatrix} \psi_0\left(\mathbf{X}_i^{(1)}\boldsymbol{\beta}\right) & \cdots & \psi_L\left(\mathbf{X}_i^{(1)}\boldsymbol{\beta}\right) \\ \vdots & \ddots & \vdots \\ \psi_0\left(\mathbf{X}_i^{(n_i)}\boldsymbol{\beta}\right) & \cdots & \psi_L\left(\mathbf{X}_i^{(n_i)}\boldsymbol{\beta}\right) \end{pmatrix}, \end{aligned}$$

and the random vector $\boldsymbol{\xi} = [\xi_0, \ldots, \xi_L]^\top$ is positive and follows a truncated multivariate normal distribution:

$$\boldsymbol{\xi} \sim \mathcal{N}_{L+1}^{+}(\mathbf{0}_{L+1}, \boldsymbol{K}),$$

representing the constrained GP prior on $\boldsymbol{\xi}$.

With the vector-valued input, $\mathbf{X}_i\boldsymbol{\beta}$, the index function $g(\cdot)$ is a function with vector-valued output. It is a collection of scalar-valued monotonic increasing functions:

$$g(\mathbf{X}_i\boldsymbol{\beta}) = \begin{pmatrix} g^\star\left(\mathbf{X}_i^{(1)}\boldsymbol{\beta}\right) \\ \vdots \\ g^\star\left(\mathbf{X}_i^{(n_i)}\boldsymbol{\beta}\right) \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Phi}^{(1)}\boldsymbol{\xi} \\ \vdots \\ \boldsymbol{\Phi}^{(n_i)}\boldsymbol{\xi} \end{pmatrix},$$

where $g^{\star}(\cdot)$ is a function with both scalar-valued input and output, and $\mathbf{\Phi}^{(1)}$ and $\mathbf{\Phi}^{(n_i)}$ represent the first and last row of $\mathbf{\Phi}$, respectively.

By Proposition 2 of Maatouk and Bay (2017), setting $\boldsymbol{\xi}$ as a positive random vector is both a *necessary and sufficient* condition for $g^{\star}(\cdot)$ to be a monotonic increasing function and for the index function $g(\cdot)$ in (4) to be coordinate-wise monotonic increasing.

The covariance matrix $\boldsymbol{K}$ is characterized by the Matérn kernel (Rasmussen and Williams, 2005), consisting of a scale parameter ρ_1, a range parameter ρ_2, and a smoothness parameter ρ_3, defined as follows:

$$C(r) = \rho_1^2 \frac{2^{1-\rho_3}}{\Gamma(\rho_3)} \left(\sqrt{2\rho_3} \frac{r}{\rho_2} \right)^{\rho_3} B_{\rho_3} \left(\sqrt{2\rho_3} \frac{r}{\rho_2} \right),$$

where r represents the distance between two measurements, $\Gamma(\cdot)$ denotes the gamma function, and $B_{\rho_3}(\cdot)$ is the modified Bessel function of the second kind. Inference about the smoothness parameter ρ_3 is challenging both theoretically and empirically (Zhang, 2004). Additionally, ρ_3 is set to 3/2 due to the simplified analytic form of the modified Bessel function of the second kind(Chen et al., 2024). Furthermore, following the suggestion by Ray et al. (2020), we assume that the covariance matrix $\boldsymbol{K}$ is obtained from a regular grid in the interval $[-1, 1]$, which matches the support of the index function. This results in $\boldsymbol{K}$ having a Toeplitz structure, for which there exists an associated efficient sampling algorithm.

1.3 Variable Selection and Prior Elicitation

In this section, we aim to address one challenging aspect in analyzing the NHANES data: the relatively high number of risk factors. We also want to discuss the prior elicitation for unknown parameters $\left(a, \boldsymbol{\beta}, \delta, d^2, \sigma^2, \nu, \rho_1^2, \rho_2\right)$ in the ST-GP model. We suggest the following list of priors:

1. We put a non-informative prior, a normal distribution with mean 0 and variance 1000, on the slope parameter a:
$$a \sim \mathcal{N}(0, 1000).$$

2. Recall that there is an identifiability restriction such that $||\boldsymbol{\beta}|| = 1$. To satisfy this restriction, the following transformation can be applied:
$$\boldsymbol{\beta} = \frac{\tilde{\boldsymbol{\beta}}}{||\tilde{\boldsymbol{\beta}}||}.$$
This transformation addresses the identifiability concern and allows for the use of the elliptical slice sampler (Murray et al., 2010), which is a tuning-free sampler. Traditional samplers used in existing Bayesian single index models often require careful tuning. In contrast, a tuning-free sampler simplifies the tuning process and enhances computational stability compared to samplers that require careful tuning.

   When the number of covariates is small, we suggest placing independent normal priors with mean 0 and variance 10 on each of $\tilde{\boldsymbol{\beta}}$. Because, for any $c > 0$, $\tilde{\boldsymbol{\beta}}/||\tilde{\boldsymbol{\beta}}|| = c\tilde{\boldsymbol{\beta}}/||c\tilde{\boldsymbol{\beta}}||$, scaling the variance of the prior on $\tilde{\boldsymbol{\beta}}$ does not alter the prior distribution of $\boldsymbol{\beta}$.

3. We put the grouped horseshoe prior on $\tilde{\boldsymbol{\beta}}^{\star} = \left\{ \tilde{\beta}^{\star}_{j,k} : j \geq 1, k \geq 1 \right\}$, such that for the j-th group and the k-th level,
$$\begin{aligned} \tilde{\beta}^{\star}_{j,k} \mid \lambda_j, \tau &\sim \mathcal{N}\left(0, \lambda_j^2 \tau^2\right), \\ \lambda_j &\sim \mathcal{C}^{0,\infty}(0, 1), \\ \tau &\sim \mathcal{C}^{0,1}(0, 1), \end{aligned} \tag{5}$$
   where $\mathcal{C}^{0,\infty}(0, 1)$ and $\mathcal{C}^{0,1}(0, 1)$ represent the standard Cauchy distribution truncated to $(0, \infty)$ and the standard Cauchy distribution truncated to $(0, 1)$ respectively.

   Last, for other data sets or for researchers who want to investigate different questions, it is advisable for researchers to determine the usage of the normal prior and the (grouped) horseshoe prior based on the specific requirements and characteristics of their data and research objectives.

4. We assign a non-informative prior to the skewness parameter δ, allowing the data to fully determine both the direction and magnitude of the skewness of the random effects:

$$\delta \sim \mathcal{N}(0, 1000).$$

5. We assign a commonly used non-informative and conjugate prior, a inverse Gamma distribution, on the variance of random effects, d^2:

$$d^2 \sim \mathcal{IG}(5, 5),$$

where $\mathcal{IG}(5, 5)$ denotes the inverse Gamma distribution with shape and scale parameters set to 5, characterized by the probability density function proportional to $x^{-5-1}\exp(-5/x)$.

6. We assign the same non-informative and conjugate prior, $\mathcal{IG}(5, 5)$, on the variance of the residual term, σ^2:

$$\sigma^2 \sim \mathcal{IG}(5, 5).$$

7. To utilize the elliptical slice sampler, we place a log-normal prior on the degree of freedom:

$$\log(\nu - 2) \sim \mathcal{N}(0, 1).$$

This prior implies a lower bound such that $\nu > 2$, ensuring the existence of the first and second moments of the random effects and residuals.

8. Similarly, for convenient use of the elliptical slice sampler, we assign the same log-normal prior on ρ_1^2 and ρ_2, two hyperparameters of the Matérn kernel:

$$\log\left(\rho_1^2\right) \sim \mathcal{N}(0, 1),$$

and

$$\log\left(\rho_2\right) \sim \mathcal{N}(0, 1).$$

2 First Simulation Study

In this simulation study, we aim to demonstrate the data generation function `reg_simulation1` and the Gibbs sampling function `Gibbs_Sampler`.

The data generation stage of the simulation study involves several key steps. The function `reg_simulation1` iterates over N subjects to generate design matrices $\mathbf{X}_i$ and outcomes $\mathbf{Y}_i$. For each subject, we replicate the non-uniform number of measurements observed in real data by setting $n_i = T + 2$, where T follows a Poisson distribution with a mean of 8. Each subject's data includes an associated $n_i \times 10$ design matrix $\mathbf{X}_i$. It then creates the design matrix $\mathbf{X}_i$ with two continuous predictors (`X1` and `X2`) and one binary predictor (`Zi`).

To generate `X1`, it draws values from a standard normal distribution. For `X2`, it first determines the value of the binary predictor `Zi` by drawing from a binomial distribution with a probability of 0.5. If `Zi` is 1, it draws values for `X2` from a normal distribution with a mean of 1; if `Zi` is 0, it draws values from a normal distribution with a mean of -1. The resulting design matrix $\mathbf{X}_i$ is standardized and scaled.

Next, the function calculates the linear predictor `etai` using the standardized $\mathbf{X}_i$. Then, it transforms `etai` using the `true.g` function, which is defined mathematically as:

$$g(x) = 5\left(\Phi\left(\frac{x+1}{2}; 0.5, 0.1\right) - \Phi\left(0; 0.5, 0.1\right)\right).$$

where $\Phi(\cdot; \mu, \sigma)$ is the cumulative distribution function of the normal distribution with mean μ and standard deviation σ. This transformation provides the outcome g_i.

Finally, the function generates the outcome $\mathbf{Y}_i$ from (3). The standardized design matrices and outcomes are then stored.

2.1 Simulate Data

```
# load the package
library(MSIMST)
```

```
set.seed(100)

simulated_data <- reg_simulation1(N = 50,
                                  ni_lambda = 8,
                                  beta = c(0.5,0.5,0.5),
                                  beta_b = 1.5,
                                  dsq = 0.1,
                                  sigmasq = 0.5,
                                  delta = 0.6,
                                  nu = 5.89)

y <- simulated_data$y
X <- simulated_data$X
```

2.2 Use the Gibbs Sampler

Users can use the `Gibbs_Sampler` function to draw samples from the posterior distribution.

```
group_info <- c(0,0,0)
L <- 50
N <- length(y)

GP_MCMC_output <- Gibbs_Sampler(X = X,
                                y = y,
                                group_info = group_info,
                                beta_value = c(0.5,0.5,0.5),
                                beta_prior_variance = 10,
                                beta_b_value = 1.5,
                                beta_lambdasq_value = 1,
                                beta_tausq_value = 1,
                                xi_value = abs(rnorm(n = L + 1)),
                                xi_lengthscale_value = 1.0,
                                xi_tausq_value = 1.0,
                                g_func_type = "GP",
                                dsq_value = 1,
                                sigmasq_value = 1,
                                delta_value = 0.6,
                                nu_value = 5.89,
                                U_value = abs(rnorm(N)),
                                S_value = abs(rnorm(N)),
                                loglik_type = "skewT",
                                gof_K = 10,
                                gof_L = 5,
                                iter_warmup = 2000,
                                iter_sampling = 5000,
                                verbatim = TRUE,
```

```
                                update = 1000,
                                incremental_output = FALSE,
                                incremental_output_filename = NULL,
                                incremental_output_update = 1e6,
                                n_core = 1)
#> [1] "i:1000"
#> [1] "i:2000"
#> [1] "i:3000"
#> [1] "i:4000"
#> [1] "i:5000"
#> [1] "i:6000"
#> [1] "i:7000"
```

2.3 Posterior Inference

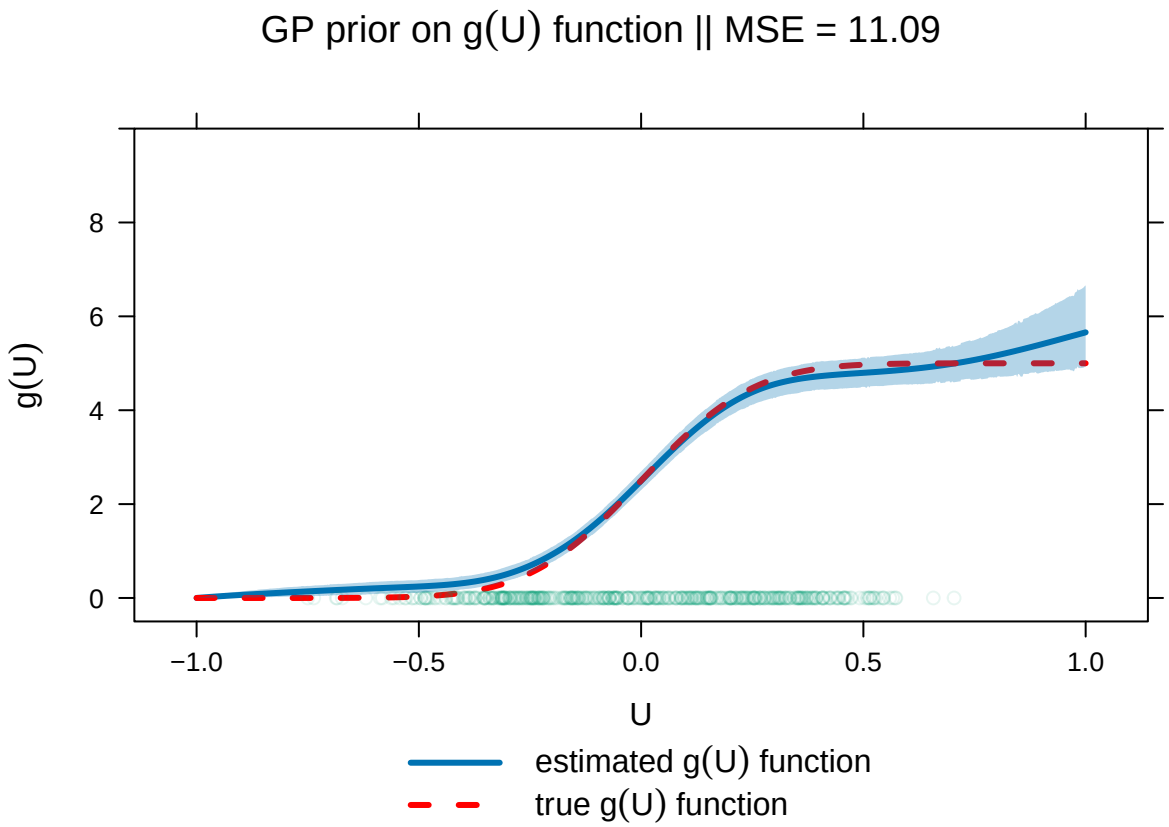
```
if (require(posterior)) {
  require(posterior)
  df_beta_GP <- GP_MCMC_output$beta_output
  colnames(df_beta_GP) <- c("beta1","beta2","beta3")
  summary(as_draws(df_beta_GP))
}
#> # A tibble: 3 x 10
#>   variable  mean median     sd    mad    q5   q95  rhat ess_bulk ess_tail
#>   <chr>    <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
#> 1 beta1    0.597  0.597 0.0129 0.0134 0.576 0.618  1.00     907.    1601.
#> 2 beta2    0.577  0.577 0.0205 0.0202 0.543 0.610  1.00     693.    1033.
#> 3 beta3    0.556  0.557 0.0247 0.0239 0.515 0.597  1.00     513.     824.
```

2.4 Plot the Estimated Single Index Function

```
L <- ncol(GP_MCMC_output$xi_output) - 1
u <- seq(-1,1,length.out = L + 1)
x.grid <- seq(-1,1,length.out = 1000)
df_g_GP <- data.frame(x = x.grid,
                      ymean = 0,
                      ylb = 0,
                      yub = 0)

# true index values
true_beta <- c(0.5,0.5,0.5) / norm(c(0.5,0.5,0.5), "2")
index_values <- unlist(lapply(X, function(x){as.numeric(x %*% true_beta)}))

# the true link function used in the simulation study
true.g <- function(x){
  y <- (x+1)/2
  5*(pnorm(y, mean=0.5, sd=0.1) - pnorm(0, mean = 0.5, sd = 0.1))
}

# true values of the g function
```

```r
df_g_GP$true_y <- true.g(x.grid)

GP_MSE <- mean((df_g_GP$ymean - df_g_GP$true_y)^2)

if (require(lattice) & require(HDInterval) & require(latex2exp)) {
  require(lattice)
  require(HDInterval)
  require(latex2exp)

  # the estimated g function - GP
  for (i in 1:length(x.grid)) {
    # use phiX_c function to generate the Phi matrix defined in Equation (3)
    phiX <- phiX_c(x.grid[i],u,L)
    value <- as.numeric(phiX%*%t(GP_MCMC_output$xi_output))
    value.mean <- mean(value)
    value.ci <- hdi(value)
    value.ub <- value.ci[2]
    value.lb <- value.ci[1]
    df_g_GP[i,"ymean"] <- value.mean
    df_g_GP[i,"ylb"] <- value.lb
    df_g_GP[i,"yub"] <- value.ub
  }

  p1 <- xyplot(ymean + true_y ~ x, data = df_g_GP,
               type = "l",
               lty = c(1,2),
               lwd = c(3,3),
               col = c("#0072B2","red"),
               panel = function(...){
                 panel.xyplot(...)
                 panel.xyplot(x = index_values,
                              y = rep(0,length(index_values)),
                              col = rgb(0,158,115,255*0.1,maxColorValue = 255))
                 panel.polygon(c(df_g_GP$x, rev(df_g_GP$x)),
                               c(df_g_GP$yub, rev(df_g_GP$ylb)),
                               col = rgb(0,114,178,255*0.3,maxColorValue = 255),
                               border = FALSE)
               },
               ylab = TeX("$g(U)$"),
               xlab = TeX("$U$"),
               key = list(space = "bottom",
                          lines = list(lty=1:2,
                                       lwd=3,
                                       col=c("#0072B2","red"),
                                       points = FALSE),
                          text = list(c(TeX("estimated $g(U)$ function"),
                                        TeX("true $g(U)$ function")))
               ),
               main = TeX(paste0("GP prior on $g(U)$ function || ",
                                 "MSE = ",
                                 round(GP_MSE,2))),
               ylim = c(-0.5,10))
  plot(p1)
```

```
}
```



3 Second Simulation Study

In the second simulation study, we aim to show that the grouped horseshoe prior in (5) efficiently separates noise from signals. We increase the number of covariates, with first covariate conforms to a categorical distribution with two levels, designated as A and B, each assigned a probability of 0.5. To emulate the prevalence of diabetes observed in actual datasets, the second covariate follows a categorical distribution with two levels: diabetes and non-diabetes, assigned probabilities of 0.13 and 0.87, respectively. To investigate the performance of the grouped horseshoe prior, it is essential to include a categorical covariate with more than two levels. Thus, the third covariate is generated from a categorical distribution with three levels, each having an equal probability of 1/3. The fourth covariate also adheres to a categorical distribution with two levels, C and D, each with a probability of 0.5. In mirroring potential correlations present in real data, if the fourth covariate assumes level C, the fifth covariate follows a normal distribution with a mean of 1 and a variance of 1; otherwise, it follows a normal distribution with a mean of -1 and the same variance. The first five covariates are associated with non-zero coefficients, whereas the remaining three covariates have coefficients assigned values of zero. The sixth covariate follows a categorical distribution with three levels, each with an equal probability of 1/3. Similarly, the seventh covariate follows a categorical distribution with two levels, each with an equal probability of 0.5. Analogous to the fifth covariate, the eighth covariate follows a normal distribution with a mean of 1 if the seventh covariate assumes the first level and a mean of -1 if it assumes the second level, both with a variance of 1. Lastly, we standardized the design matrix to ensure that the L_2 norm of each row of all $\mathbf{X}_i$ is less than 1.

3.1 Simulate Data

```
set.seed(200)
simulated_data <- reg_simulation2(N = 50,
                                  ni_lambda = 8,
                                  beta = c(rep(1,6),rep(0,4)),
                                  beta_b = 1.5,
                                  dsq = 0.1,
                                  sigmasq = 0.5,
                                  delta = 0.6,
                                  nu = 5.89)

y <- simulated_data$y
X <- simulated_data$X
```

3.2 Use the Gibbs Sampler

```
group_info <- c(rep(0,2),
                rep(1,2),
                2,3,
                rep(4,2),
                5,6)
L <- 50
N <- length(y)

GP_MCMC_output <- Gibbs_Sampler(X = X,
                                y = y,
                                group_info = group_info,
                                beta_value = c(rep(1,6),rep(0,4)),
                                beta_prior_variance = 10,
                                beta_b_value = 1.5,
                                beta_lambdasq_value = rep(1.0,max(group_info)),
                                beta_tausq_value = 1,
                                xi_value = abs(rnorm(n = L + 1)),
                                xi_lengthscale_value = 1.0,
                                xi_tausq_value = 1.0,
                                g_func_type = "GP",
                                dsq_value = 0.1,
                                sigmasq_value = 0.5,
                                delta_value = 0.6,
                                nu_value = 5.89,
                                U_value = abs(rnorm(N)),
                                S_value = abs(rnorm(N)),
                                loglik_type = "skewT",
                                gof_K = 10,
                                gof_L = 5,
                                iter_warmup = 10000,
                                iter_sampling = 10000,
                                verbatim = TRUE,
                                update = 5000,
                                incremental_output = FALSE,
```

```
                                incremental_output_filename = NULL,
                                incremental_output_update = 1e6,
                                n_core = 1)
#> [1] "i:5000"
#> [1] "i:10000"
#> [1] "i:15000"
#> [1] "i:20000"
```

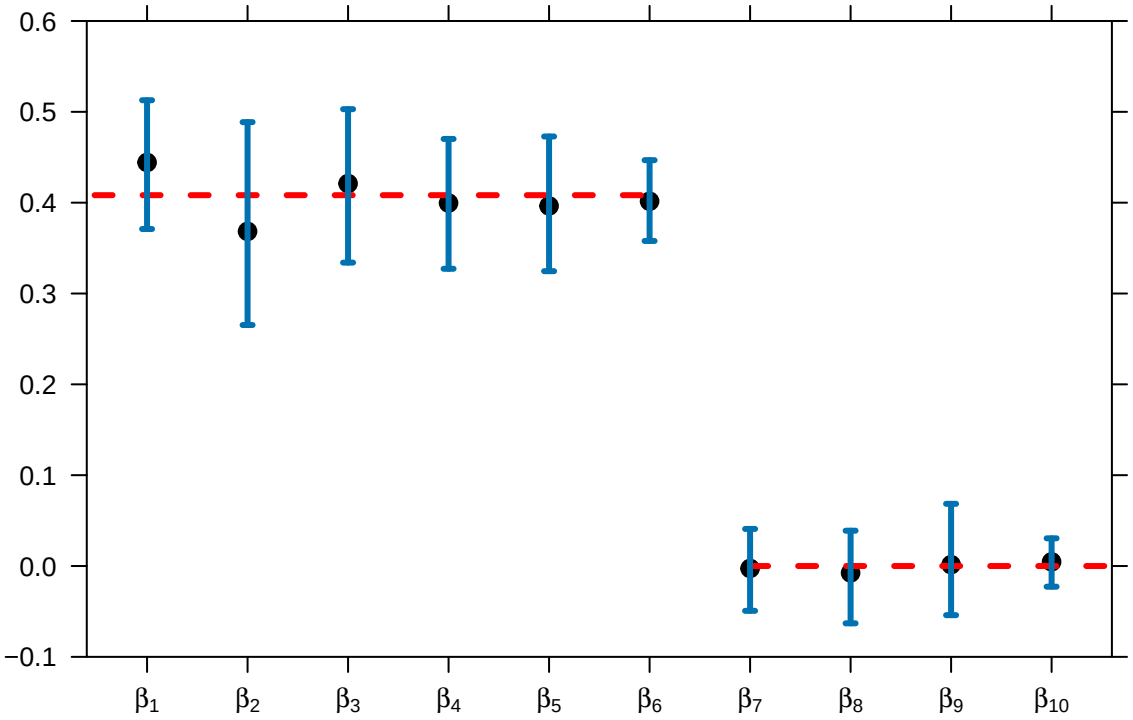
3.3 Posterior Inference

```
if (require(posterior)) {
  df_beta_GP <- GP_MCMC_output$beta_output
  colnames(df_beta_GP) <- paste0("beta",1:10)
  summary(as_draws(df_beta_GP))
}
#> # A tibble: 10 x 10
#>    variable     mean    median     sd    mad       q5    q95  rhat ess_bulk ess_tail
#>    <chr>       <dbl>     <dbl>  <dbl>  <dbl>    <dbl>  <dbl> <dbl>    <dbl>    <dbl>
#>  1 beta1     0.444    0.445    0.0358 0.0353  0.385  0.502   1.00    919.     1654.
#>  2 beta2     0.368    0.365    0.0567 0.0527  0.281  0.466   1.00    363.      750.
#>  3 beta3     0.421    0.422    0.0422 0.0415  0.349  0.490   1.00    308.      581.
#>  4 beta4     0.400    0.400    0.0365 0.0356  0.336  0.458   1.00    425.      694.
#>  5 beta5     0.396    0.396    0.0383 0.0389  0.335  0.460   1.00    362.      809.
#>  6 beta6     0.402    0.401    0.0228 0.0227  0.365  0.440   1.00    651.     1331.
#>  7 beta7    -0.00262 -0.000921 0.0209 0.0175 -0.0398 0.0327  1.01     31.0     140.
#>  8 beta8    -0.00755 -0.00461  0.0241 0.0200 -0.0522 0.0290  1.04     25.6      81.2
#>  9 beta9     0.00151  0.000291 0.0266 0.0174 -0.0435 0.0533  1.02     85.6      76.8
#> 10 beta10    0.00461  0.00409  0.0128 0.0113 -0.0167 0.0248  1.01    198.      297.
```

3.4 Variable Selection Plot

```
if (require(posterior)) {
  beta_draws <- as_draws(df_beta_GP)
  df_plot_vs <- summarise_draws(beta_draws,
                                mean,
                                ~quantile(.x, probs = c(0.025, 1-0.025)))
  df_plot_vs$variable <- factor(df_plot_vs$variable,
                                levels = paste0("beta",1:10))
}

if (require(lattice) & require(latex2exp)) {
  true_beta <- c(rep(1,6),rep(0,4)) / norm(c(rep(1,6),rep(0,4)), "2")
  beta_label <- TeX(paste0("$\\beta_{",1:10,"}$"))
  VS_plot <- xyplot(mean ~ variable,
                    data = df_plot_vs,
                    panel = function(x,y,...) {
                      panel.xyplot(x = x, y = y,...)
                      panel.xyplot(x = 0:6,
                                   y = rep(true_beta[1],7),
```

```
                                type = "l",
                                col = "red",
                                lty = 2,
                                lwd = 3)
                  panel.xyplot(x = 7:11,
                                y = rep(0,5),
                                type = "l",
                                col = "red",
                                lty = 2,
                                lwd = 3)
                  for (j in 1:10) {
                    panel.arrows(x0 = j,
                                 y0 = df_plot_vs$`2.5%`[j],
                                 x1 = j,
                                 y1 = df_plot_vs$`97.5%`[j],
                                 length = 0.05, unit = "native",
                                 angle = 90,
                                 code = 3,
                                 lwd = 3,
                                 col = "#0072B2")
                  }
                },
                col = "black",
                pch = 19,
                cex = 1.1,
                ylab = NULL,
                xlab = NULL,
                scales = list(x = list(labels = beta_label),
                              y = list(at = seq(-0.1,0.6,0.1),
                                       limits = c(-0.1,0.6))))
  plot(VS_plot)
}
```



4 Third Simulation Study

The goal of the third simulation study is to demonstrate the effectiveness of the Bayesian method to adjust the survey weights (Gunawan et al., 2020), implemented in the function `WFPBB()`.

In this simulation study, we introduce a selection variable Z. When a sample is taken from the population, the Z-value for a subject in the population determines the probability of selecting that subject into the sample. Specifically, we assume that for each subject, the joint distribution of the selection variable Z and the response variable $\mathbf{Y}$ is

$$\begin{pmatrix} \mathbf{Y}_i \\ Z_i \end{pmatrix} \sim ST_{2n_i+1}\left[\begin{pmatrix} \boldsymbol{\theta}_i + b\delta\mathbf{1}_{2n_i} \\ \mu_z \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Psi}_i & \rho \times \mathbf{1}_{2n_i} \\ \rho \times \mathbf{1}_{2n_i}^\top & \sigma_z^2 \end{pmatrix}, \begin{pmatrix} \delta\mathbf{1}_{2n_i} \\ 0 \end{pmatrix}, \nu\right], \tag{6}$$

where

$$\boldsymbol{\theta}_i = \begin{pmatrix} g\left(\mathbf{X}_i\boldsymbol{\beta}\right) \\ a \times g\left(\mathbf{X}_i\boldsymbol{\beta}\right) \end{pmatrix},$$

$g(\cdot)$ is the same index function introduced in the first simulation study, and

$$\boldsymbol{\Psi}_i = d^2\mathbf{1}_{2n_i}\mathbf{1}_{2n_i}^\top + \sigma^2\mathbf{I}_{2n_i}.$$

Marginally, $\mathbf{Y}_i$ comes from the ST-GP model defined in (1) and (2). Additionally, to replicate the intrinsic sampling mechanism in real data, we introduce a sampling mechanism here. We assume that $\mathbf{Y}_i$ is selected into the sample if and only if $I_i = 1$, where

$$\mathbb{P}\left(I_i = 1 \mid \mathbf{Y}_i, Z_i\right) = \mathbb{P}\left(I_i = 1 \mid Z_i\right) = \pi_i = \text{logistic}\left(\zeta_0 + \zeta_1 Z_i\right), \tag{7}$$

with logistic $(\cdot)$ denoting the standard logistic function.

4.1 Simulate Data

```
set.seed(100)
# set the population size
population_N <- 1000
# group information and the number of nodes

group_info <- c(rep(0,2),
                rep(1,2),
                2,3,
                rep(4,2),
                5,6)

L <- 50
output_data <- reg_simulation3(N = population_N,
                               ni_lambda= 8,
                               beta = c(rep(1,6),rep(0,4)),
                               beta_b = 1.5,
                               dsq = 0.1,
                               sigmasq = 0.5,
                               delta = 0.6,
                               nu = 5.89,
                               muz = 0,
                               rho = 36.0,
                               sigmasq_z = 0.6,
                               zeta0 = -1.8,
                               zeta1 = 0.1)
```

4.2 Use the Gibbs Sampler and the `WFPBB()` Function

Calling `WFPBB()` and `Gibbs_Sampler()` takes around 12 hours in a personal laptop. Users can test the following codes by themselves.

```
N <- length(output_data$y)
survey_weight <- output_data$survey_weight

# the size of bootstrap is 50
size_bootstrap <- 50
output_MCMC <- list(beta = vector("list",size_bootstrap),
                    beta_b = vector("list",size_bootstrap),
                    delta = vector("list",size_bootstrap),
                    dsq = vector("list",size_bootstrap),
                    sigmasq = vector("list",size_bootstrap),
                    nu = vector("list",size_bootstrap))

for (j in 1:size_bootstrap) {

  print(paste0("bootstrap iteration: ", j, "of ", size_bootstrap, "."))

  condition <- TRUE
  while (condition) {
    index_WFPBB <- WFPBB(y = 1:N,
```

```
                        w = survey_weight,
                        N = population_N,
                        n = N)

  y <- output_data$y
  y <- y[index_WFPBB]
  X <- output_data$X
  X <- X[index_WFPBB]

  # check full rank condition
  X_merged <- do.call("rbind",X)
  X_rank <- Matrix::rankMatrix(X_merged)[1]
  if (ncol(X_merged) == X_rank) {
    print("WFPBB procedure is successful.")
    condition <- FALSE
  } else {
    print("WFPBB procedure is unsuccessful (design matrix is not full ranked). Will try again.")
  }
  rm(X_merged)
  rm(X_rank)
}

GP_MCMC_output <- Gibbs_Sampler(X = X,
                                y = y,
                                group_info = group_info,
                                beta_value = c(rep(1,6),rep(0,4)),
                                beta_prior_variance = 10,
                                beta_b_value = 1.5,
                                beta_lambdasq_value = rep(1.0,max(group_info)),
                                beta_tausq_value = 1,
                                xi_value = abs(rnorm(n = L + 1)),
                                xi_lengthscale_value = 1.0,
                                xi_tausq_value = 1.0,
                                g_func_type = "GP",
                                dsq_value = 0.1,
                                sigmasq_value = 0.5,
                                delta_value = 0.6,
                                nu_value = 5.89,
                                U_value = abs(rnorm(N)),
                                S_value = abs(rnorm(N)),
                                loglik_type = "skewT",
                                gof_K = 10,
                                gof_L = 5,
                                iter_warmup = 5000,
                                iter_sampling = 10000,
                                verbatim = TRUE,
                                update = 5000,
                                incremental_output = FALSE,
                                incremental_output_filename = NULL,
                                incremental_output_update = 1e6,
                                n_core = 1)

output_MCMC$beta[[j]] <- GP_MCMC_output$beta_output
```

```
  output_MCMC$beta_b[[j]] <- GP_MCMC_output$beta_b_output
  output_MCMC$delta[[j]] <- GP_MCMC_output$delta_output
  output_MCMC$dsq[[j]] <- GP_MCMC_output$dsq_output
  output_MCMC$sigmasq[[j]] <- GP_MCMC_output$sigmasq_output
  output_MCMC$nu[[j]] <- GP_MCMC_output$nu_output
}
```

4.3 Posterior Inference

```
# chunk not evaluated.
if (require(posterior)) {
  df_beta <- do.call("rbind",output_MCMC$beta)
  colnames(df_beta) <- paste0("beta",1:10)
  df_beta_summary <- summarise_draws(as_draws(df_beta),
                                     mean,
                                     ~quantile(.x, probs = c(0.025, 1-0.025)))
  print(df_beta_summary)
}
```

A Skewed Distributions

We introduce the definition of the ST distribution by first explaining the construction of the SN distribution. The construction of the SN distribution begins with a linear combination of two *independent* normal distributions. A random vector $\mathbf{Y}$ follows a skew-normal (SN) distribution with a $p \times 1$ location vector $\boldsymbol{\mu}$, a $p \times p$ scale matrix $\boldsymbol{\Omega}$, and a $p \times 1$ skewness vector $\boldsymbol{\delta}$, denoted as $\mathbf{Y} \sim \mathrm{SN}_p(\boldsymbol{\mu}, \boldsymbol{\Omega}, \boldsymbol{\delta})$, if it can be expressed as:

$$\mathbf{Y} = \boldsymbol{\mu} + \boldsymbol{\delta}|X_0| + \mathbf{X}_1, \tag{8}$$

where X_0 follows a univariate standard normal distribution, and $\mathbf{X}_1$ follows a multivariate normal distribution with zero mean and a covariance matrix $\boldsymbol{\Omega}$. The random variable that follows the truncated normal distribution, $|X_0|$, along with the skewness vector $\boldsymbol{\delta}$, brings skewness into the SN distribution.

By introducing one more latent variable, denoted as U, which is independent of X_0 and $\mathbf{X}_1$ and follows a Gamma distribution with shape and rate parameters both equal to $\nu/2$, i.e., $U \sim \mathrm{Gamma}(\nu/2, \nu/2)$, where its density function is proportional to $u^{0.5\nu-1}\exp(-0.5\nu u)$, we can construct the ST distribution as follows:

$$\mathbf{Y} = \boldsymbol{\mu} + U^{-1/2}\left(\boldsymbol{\delta}|X_0| + \mathbf{X}_1\right), \tag{9}$$

which is denoted as $\mathbf{Y} \sim \mathrm{ST}_p(\boldsymbol{\mu}, \boldsymbol{\Omega}, \boldsymbol{\delta}, \nu)$. Adding the new latent variable U introduces heavy tail and high kurtosis features into the ST distribution.

The stochastic representations of the ST and SN distributions in (8) and (9) are not only useful for sampling from the ST/SN distributions but also imply the relationship between the ST distribution and the SN distribution. As the degree of freedom parameter ν approaches infinity, U converges to 1 in probability, and therefore, the ST distribution converges to the SN distribution. Additionally, (8) and (9) imply that the normal distribution is a special case of the SN distribution, and that both the normal distribution and the Student-t distribution are special cases of the ST distribution. With the shape vector $\boldsymbol{\delta}$ set as a vector of zeros, the random vector defined in (9) follows a multivariate Student-t distribution with the degree of freedom parameter ν, while the random vector defined in (8) follows a multivariate normal distribution.

Finally, the stochastic representation of the ST distribution in (9) also implies an equivalent hierarchical representation:

$$\begin{aligned} \mathbf{Y} \mid S, U &\sim \mathcal{N}_p\left(\boldsymbol{\mu} + u^{-1/2} s\boldsymbol{\delta}, u^{-1}\boldsymbol{\Omega}\right), \\ S &\sim \mathcal{N}^+(0,1), \\ U &\sim \text{Gamma}(\nu/2, \nu/2). \end{aligned} \tag{10}$$

Integrating out two latent variables, S and U, we obtain the density function of the ST distribution as:

$$f_{\mathbf{Y}}(\mathbf{Y}) = 2t_p(\mathbf{Y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu) T\left(\boldsymbol{\delta}^\top \boldsymbol{\Sigma}^{-1}(\mathbf{Y} - \boldsymbol{\mu}) \sqrt{\frac{\nu + p}{\nu + d(\mathbf{Y})}} \mid 0, \Lambda, \nu + p\right), \tag{11}$$

where t_p represents the density function of a p-dimensional multivariate Student-t distribution, T represents the cumulative distribution function of a univariate Student-t distribution. Additional $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}$ are given as follows:

$$\begin{aligned} \boldsymbol{\Sigma} &= \boldsymbol{\Omega} + \boldsymbol{\delta}\boldsymbol{\delta}^\top, \\ \Lambda &= 1 - \boldsymbol{\delta}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\delta}. \end{aligned}$$

Furthermore, $d(\mathbf{Y})$ is defined as:

$$d(\mathbf{Y}) = (\mathbf{Y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{Y} - \boldsymbol{\mu}).$$

Both representations of the ST distribution in (9) and (10), along with its density function in (11), are utilized in the tailored Gibbs sampler.

A notable feature of the ST distribution is its closure under linear transformation, as demonstrated in Proposition 5 of Schumacher et al. (2021). That is, if $\mathbf{Y} \sim \text{ST}_p(\boldsymbol{\mu}, \boldsymbol{\Omega}, \boldsymbol{\delta}, \nu)$ then

$$\mathbf{A}\mathbf{Y} + \mathbf{b} \sim \text{ST}_m\left(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Omega}\mathbf{A}^\top, \mathbf{A}\boldsymbol{\delta}, \nu\right), \tag{12}$$

where $\mathbf{A}$ is a $m \times p$ matrix and $\mathbf{b}$ is a vector of length m.

B The Hierarchical Representation

Using Proposition 5 of Schumacher et al. (2021), we have

$$\mathbf{Y}_i \sim \text{ST}_{2n_i}\left(\boldsymbol{\theta}_i + h(\nu)\delta \mathbf{1}_{2n_i}, \boldsymbol{\Psi}_i, \delta \mathbf{1}_{2n_i}, \nu\right), \tag{13}$$

where

$$\boldsymbol{\theta}_i = \begin{pmatrix} g(\mathbf{X}_i\boldsymbol{\beta}) \\ a \times g(\mathbf{X}_i\boldsymbol{\beta}) \end{pmatrix}$$

and

$$\boldsymbol{\Psi}_i = d^2 \mathbf{1}_{2n_i} \mathbf{1}_{2n_i}^\top + \sigma^2 \mathbf{I}_{2n_i \times 2n_i}$$

represents a covariance matrix characterized by a compound symmetry structure, with readily available closed-form expressions for its inverse and determinant.

From Proposition 6 of Schumacher et al. (2021), we have the following stochastic representation,

$$\begin{aligned} \mathbf{Y}_i \mid \cdot &\sim \mathcal{N}_{2n_i}\left(\boldsymbol{\theta}_i + h(\nu)\delta \mathbf{1}_{2n_i} + \delta s_i \mathbf{1}_{2n_i}, u_i^{-1}\boldsymbol{\Psi}_i\right) \\ S_i \mid \cdot &\sim \mathcal{N}^+\left(0, u_i^{-1}\right) \\ U_i &\sim \text{Gamma}(\nu/2, \nu/2), \quad i = 1, \ldots, N. \end{aligned} \tag{14}$$

Here, $\mathcal{N}^+\left(0, u_i^{-1}\right)$ represents the half-normal distribution with a location parameter of 0 and a scale parameter of $\sqrt{u_i^{-1}}$.

Last, (14) can also be represented in the following way.

$$\begin{aligned}
\mathbf{Y}_i \mid \cdot &\sim \mathcal{N}_{2n_i}\left(\boldsymbol{\theta}_i + \mathbf{1}_{2n_i} b_i, u_i^{-1}\sigma^2 \mathbf{I}_{2n_i}\right) \\
b_i \mid \cdot &\sim \mathcal{N}\left(\delta\left(h(\nu) + s_i\right), u_i^{-1} d^2\right) \\
S_i \mid \cdot &\sim \mathcal{N}^{+}\left(0, u_i^{-1}\right) \\
U_i &\sim \text{Gamma}\left(\nu/2, \nu/2\right), \quad i = 1, \ldots, N.
\end{aligned} \tag{15}$$

With conjugate priors in Section 1.3, utilizing (14), we can derive the updating equations for β_0, δ, S_i, and U_i. To update $\boldsymbol{\xi}$, we can employ the exact Hamiltonian algorithm, as described by Pakman and Paninski (2014). Leveraging Equation (15), we can establish the updating equations for b_i, σ^2, and d^2. The remaining parameters, which include $\boldsymbol{\beta}$, ν, as well as the hyperparameters associated with the constrained GP prior, will be updated using the elliptical slice sampler algorithm.

C The Gibbs Sampler

To simplify notation, let

$$g\left(\mathbf{X}_i\boldsymbol{\beta}\right) = g_i.$$

Update a

The prior for a is

$$a \sim \mathcal{N}\left(0, \sigma_a^2\right).$$

Let

$$\begin{aligned}
\boldsymbol{\Omega}_{i,a} &= u_i^{-1}\sigma^2 \mathbf{I}_{n_i}, \\
\mathbf{Y}_{i,a}^{\star} &= \mathbf{Y}_i^C - \mathbf{1}_{n_i} b_i.
\end{aligned} \tag{16}$$

After simple algebra,

$$a \mid \cdot \sim \mathcal{N}\left(\frac{\sum_{i=1}^N g_i^\top \boldsymbol{\Omega}_{i,a}^{-1}\mathbf{Y}_{i,a}^{\star}}{\sigma_a^{-2} + \sum_{i=1}^N g_i^\top \boldsymbol{\Omega}_{i,a}^{-1} g_i}, \frac{1}{\sigma_a^{-2} + \sum_{i=1}^N g_i^\top \boldsymbol{\Omega}_{i,a}^{-1} g_i}\right). \tag{17}$$

Update $\boldsymbol{\xi}$

Given hyperparameters ρ_1^2 and ρ_2, the distribution for $\boldsymbol{\xi} = (\xi_0, \ldots, \xi_L)^\top$ is

$$\boldsymbol{\xi} \mid \cdot \sim \mathcal{N}_{L+1}^{+}\left(\mathbf{0}_{L+1}, \boldsymbol{K}\right).$$

Let

$$\begin{aligned}
\mathbf{Y}_{i,\boldsymbol{\xi}}^{\star} &= \begin{pmatrix} \mathbf{Y}_i^P - \mathbf{1}_{n_i \times 1}\left(h(\nu)\delta + s_i\delta\right) \\ \left(\mathbf{Y}_i^C - \mathbf{1}_{n_i \times 1}\left(h(\nu)\delta + s_i\delta\right)\right)/a \end{pmatrix}, \\
\boldsymbol{\Omega}_{i,\boldsymbol{\xi}} &= \begin{pmatrix} \mathbf{I}_{n_i} & \mathbf{0}_{n_i} \\ \mathbf{0}_{n_i} & a^{-1}\mathbf{I}_{n_i} \end{pmatrix} u_i^{-1}\boldsymbol{\Psi}_i \begin{pmatrix} \mathbf{I}_{n_i} & \mathbf{0}_{n_i} \\ \mathbf{0}_{n_i} & a^{-1}\mathbf{I}_{n_i} \end{pmatrix},
\end{aligned}$$

and

$$\boldsymbol{\Phi}_i = \begin{pmatrix} \phi_0\left(\mathbf{X}_i\boldsymbol{\beta}\right), \ldots, \phi_L\left(\mathbf{X}_i\boldsymbol{\beta}\right) \\ \phi_0\left(\mathbf{X}_i\boldsymbol{\beta}\right), \ldots, \phi_L\left(\mathbf{X}_i\boldsymbol{\beta}\right) \end{pmatrix}.$$

After simple algebra,

$$\boldsymbol{\xi} \mid \cdot \sim \mathcal{N}_{L+1}^{+}\left(\left(\boldsymbol{K}^{-1} + \sum_{i=1}^N \boldsymbol{\Phi}_i^\top \boldsymbol{\Omega}_{i,\boldsymbol{\xi}}^{-1}\boldsymbol{\Phi}_i\right)^{-1}\left(\sum_{i=1}^N \boldsymbol{\Phi}_i^\top \boldsymbol{\Omega}_{i,\boldsymbol{\xi}}^{-1}\mathbf{Y}_{i,\boldsymbol{\xi}}^{\star}\right), \left(\boldsymbol{K}^{-1} + \sum_{i=1}^N \boldsymbol{\Phi}_i^\top \boldsymbol{\Omega}_{i,\boldsymbol{\xi}}^{-1}\boldsymbol{\Phi}_i\right)^{-1}\right).$$

Update δ

The prior for δ is

$$\delta \sim \mathcal{N}\left(0, \sigma_\delta^2\right).$$

Let

$$\mathbf{Y}^\star_{i,\delta} = \begin{pmatrix} \mathbf{Y}_i^P \\ \mathbf{Y}_i^C \end{pmatrix} - \begin{pmatrix} g_i \\ ag_i \end{pmatrix},$$

and

$$\mathbf{\Omega}_{i,\delta} = u_i^{-1}\mathbf{\Psi}_i.$$

After simple algebra,

$$\delta \mid \cdot \sim \mathcal{N}\left(\frac{\sum_{i=1}^N \left(h(\nu)+s_i\right)\mathbf{1}_{2n_i}^\top \mathbf{\Omega}_{i,\delta}^{-1}\mathbf{Y}^\star_{i,\delta}}{\sigma_\delta^{-2}+\sum_{i=1}^N \left(h(\nu)+s_i\right)^2 \mathbf{1}_{2n_i}^\top \mathbf{\Omega}_{i,\delta}^{-1}\mathbf{1}_{2n_i}}, \frac{1}{\sigma_\delta^{-2}+\sum_{i=1}^N \left(h(\nu)+s_i\right)^2 \mathbf{1}_{2n_i}^\top \mathbf{\Omega}_{i,\delta}^{-1}\mathbf{1}_{2n_i}}\right).$$

Update S_i

Let

$$\mathbf{Y}^\star_{i,S_i} = \mathbf{Y}_i - \boldsymbol{\theta}_i - h(\nu)\delta\mathbf{1}_{2n_i},$$

and

$$\mathbf{\Omega}_{i,S_i} = u_i^{-1}\mathbf{\Psi}_i.$$

After simple algebra,

$$S_i \mid \cdot \sim \mathcal{N}^+\left(\frac{\delta\mathbf{1}_{1\times 2n_i}\mathbf{\Omega}_i^{-1}\mathbf{Y}_i^\star}{u_i+\delta^2\mathbf{1}_{1\times 2n_i}\mathbf{\Omega}_{i,\delta}^{-1}\mathbf{1}_{2n_i\times 1}}, \frac{1}{u_i+\delta^2\mathbf{1}_{1\times 2n_i}\mathbf{\Omega}_{i,\delta}^{-1}\mathbf{1}_{2n_i\times 1}}\right).$$

Update U_i

Let

$$\mathbf{Y}^\star_{i,U_i} = \mathbf{Y}_i - \boldsymbol{\theta}_i - h(\nu)\delta\mathbf{1}_{2n_i\times 1} - \delta s_i\mathbf{1}_{2n_i\times 1}.$$

$$U_i \mid \cdot \sim \text{Gamma}\left(0.5\left(2n_i+\nu+1\right), 0.5\left(\mathbf{Y}^\star_{i,U_i}{}^\top \mathbf{\Psi}_i^{-1}\mathbf{Y}^\star_{i,U_i}+s_i^2+\nu\right)\right).$$

Update b_i

Let

$$\mathbf{Y}^\star_{i,b_i} = \mathbf{Y}_i - \boldsymbol{\theta}_i.$$

$$b_i \mid \cdot \sim \mathcal{N}\left(\frac{\sigma^{-2}\left(\mathbf{1}_{n_i}^\top\mathbf{Y}^\star_{i,b_i}\right)+\delta\left(h(\nu)+s_i\right)d^{-2}}{2n_i\sigma^{-2}+d^{-2}}, \frac{1}{2n_iu_i\sigma^{-2}+u_id^{-2}}\right).$$

Update σ^2

The prior for σ^2 is

$$\sigma^2 \sim \text{Inverse Gamma}\left(a_{\sigma^2}, b_{\sigma^2}\right).$$

Let

$$\mathbf{Y}^\star_{i,\sigma^2} = \mathbf{Y}_i - \boldsymbol{\theta}_i - \mathbf{1}_{2n_i}b_i.$$

$$\sigma^2 \mid \cdot \sim \text{Inverse Gamma}\left(a_{\sigma^2}+\sum_{i=1}^N n_i, b_{\sigma^2}+0.5\sum_{i=1}^N u_i\left(\mathbf{Y}^\star_{i,\sigma^2}{}^\top\mathbf{Y}^\star_{i,\sigma^2}\right)\right).$$

Update d^2

The prior for d^2 is

$$d^2 \sim \text{Inverse Gamma}\left(a_{d^2}, b_{d^2}\right).$$

$$d^2 \mid \cdot \sim \text{Inverse Gamma}\left(0.5N + a_{d^2}, b_{d^2} + 0.5\sum_{i=1}^{N} u_i \left(b_i - \delta\left(h(\nu) + s_i\right)\right)^2\right).$$

References

Chen, J., Mu, W., Li, Y., and Li, D. (2024). On the identifiability and interpretability of gaussian process models. *Advances in Neural Information Processing Systems*, 36.

Gunawan, D., Panagiotelis, A., Griffiths, W., and Chotikapanich, D. (2020). Bayesian weighted inference from surveys. *Australian & New Zealand Journal of Statistics*, 62(1):71–94.

Maatouk, H. and Bay, X. (2017). Gaussian process emulators for computer experiments with inequality constraints. *Mathematical Geosciences*, 49(5):557–582.

Murray, I., Adams, R., and MacKay, D. (2010). Elliptical slice sampling. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 541–548. JMLR Workshop and Conference Proceedings.

Pakman, A. and Paninski, L. (2014). Exact hamiltonian monte carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.

Ray, P., Pati, D., and Bhattacharya, A. (2020). Efficient bayesian shape-restricted function estimation with constrained gaussian process priors. *Statistics and Computing*, 30:839–853.

Schumacher, F. L., Matos, L. A., and Cabral, C. R. (2021). Canonical fundamental skew-t linear mixed models. *arXiv preprint arXiv:2109.12152*.

Zhang, H. (2004). Inconsistent estimation and asymptotically equal interpolations in model-based geostatistics. *Journal of the American Statistical Association*, 99(465):250–261.