Bayesian Spike-and-Slab Approach for Differential Protein Expression in the Ts65Dn Mouse Model

Deeptendra Banerjee (MB2409)
Indian Statistical Institute, Kolkata

1 Introduction

Down syndrome (DS), the leading genetic cause of intellectual disability, results from trisomy of chromosome 21, affecting about one in 1,000 live births. Overexpression of chromosome 21 genes disrupts cellular signalling, impairing synaptic plasticity, learning, and memory.

The Ts65Dn mouse model, carries a partial trisomy encompassing orthologues of human chromosome 21 genes and recapitulates core phenotypes of the disorder, including hippocampal dysfunction and learning deficits. In particular, Ts65Dn mice exhibit impaired performance in context–fear conditioning (CFC), a paradigm assessing associative learning, which can be pharmacologically restored ("rescued") by treatment with the NMDA receptor antagonist Memantine.

Higuera et al. [2015] systematically quantified the expression of 77 nuclear-enriched cortical proteins across eight experimental conditions defined by genotype, behavioral context, and drug treatment. Using self-organizing maps, they identified sets of proteins differentially associated with normal learning, failed learning, and rescued learning phenotypes.

Here, we reanalyse the same dataset using Bayesian spike-and-slab probit regression framework to infer posterior inclusion probabilities (PIPs) for each protein. This approach enables quantitative assessment of variable relevance within each biological contrast, extending Higuera et al. [2015]'s qualitative findings

2 Dataset Description

2.1 Summary

- **Number of instances:** 1080 observations corresponding to 72 mice (38 control and 34 trisomic), each providing 15 replicate measurements.
- **Number of features:** 80
- **Continuous variables:** 77 protein or protein-modification expression levels measured in the nuclear fraction of cortical tissue.
- **Categorical variables:** Genotype (control or trisomy), Treatment (saline or memantine), and Behavior (context-shock or shock-context).
- **Identifier:** MouseID (encoding both the unique biological subject and replicate index).
- **Missing values:** Present in some protein measurements.

2.2 Class Structure

Each observation falls into one of eight experimental classes defined by the cross of genotype (control or trisomy), behavior (context–shock or shock–context), and treatment (saline or memantine). The resulting classes are:

$$\text{c-CS-s},\ \text{c-CS-m},\ \text{c-SC-s},\ \text{c-SC-m},$$
$$\text{t-CS-s},\ \text{t-CS-m},\ \text{t-SC-s},\ \text{t-SC-m},$$

Here, "CS" denotes the condition in which mice were expected to learn, while "SC" represents the non-learning control.

2.3 Source

UCI Machine Learning Repository: *Mice Protein Expression Data Set*, available at `https://archive.ics.uci.edu/dataset/342/mice+protein+expression`.

3 Model

To identify protein expression covariates that discriminate between experimental classes, we employ a Bayesian spike-and-slab probit regression model. The probit formulation follows Albert and Chib [1993], and the spike-and-slab prior specification follows Kuo and Mallick [1998].

For observation $i = 1, \ldots, n$ and covariate $j = 1, \ldots, p$,

$$y_i = \mathbb{I}\{z_i \geq 0\},$$

$$z_i = \mathbf{x}_i^\top (\boldsymbol{\gamma} \odot \boldsymbol{\beta}) + \varepsilon_i, \quad \varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2),$$

where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})^\top$ is the p-dimensional feature vector for the ith observation, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)^\top$ is the vector of binary inclusion indicators, and $\odot$ denotes elementwise multiplication. The hierarchical priors are specified as:

$$\begin{aligned}\beta &\overset{\text{ind}}{\sim} \mathcal{N}_p(\beta_0, D_0),\\ \gamma_j &\overset{\text{ind}}{\sim} \text{Bernoulli}(\pi_j),\\ \sigma^2 &\sim \text{InvGamma}(\alpha_0, \eta_0).\end{aligned}$$

McMC steps

$$z_i \mid y_i, \mathbf{x}_i, \boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma^2 \sim \begin{cases}\mathcal{N}\big(\mathbf{x}_i^\top(\boldsymbol{\gamma}\odot\boldsymbol{\beta}),\, \sigma^2\big) \text{ truncated to } [0,\infty), & \text{if } y_i = 1,\\ \mathcal{N}\big(\mathbf{x}_i^\top(\boldsymbol{\gamma}\odot\boldsymbol{\beta}),\, \sigma^2\big) \text{ truncated to } (-\infty,0), & \text{if } y_i = 0.\end{cases}$$

$$\boldsymbol{\beta} \mid \mathbf{z}, \boldsymbol{\gamma}, \sigma^2 \sim \mathcal{N}_p\left(\left(D_0^{-1} + \frac{1}{\sigma^2}X_{\boldsymbol{\gamma}}^\top X_{\boldsymbol{\gamma}}\right)^{-1}\left(D_0^{-1}\boldsymbol{\beta}_0 + \sigma^{-2}X_{\boldsymbol{\gamma}}^\top \mathbf{z}\right).\left(D_0^{-1} + \frac{1}{\sigma^2}X_{\boldsymbol{\gamma}}^\top X_{\boldsymbol{\gamma}}\right)^{-1}\right)$$

$$\sigma^2 \mid \mathbf{z}, \boldsymbol{\beta}, \boldsymbol{\gamma} \sim \text{Inv-Gamma}\left(\alpha_0 + \frac{n}{2},\ \eta_0 + \frac{1}{2}\|\mathbf{z} - X_{\boldsymbol{\gamma}}\boldsymbol{\beta}\|_2^2\right)$$

$$\Pr(\gamma_j = 1 \mid \mathbf{z}, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\gamma}_{-j}) = \frac{c_j}{c_j + d_j}$$

where

$$\begin{aligned}X_\gamma &= [\gamma_1 \boldsymbol{x_1}, \cdots, \gamma_p \boldsymbol{x_p}]\\ c_j &= \pi_j \exp\left[-\frac{1}{2\sigma^2}\|\mathbf{z} - X_{\boldsymbol{\gamma}^{(j=1)}}\boldsymbol{\beta}\|_2^2\right],\\ d_j &= (1-\pi_j)\exp\left[-\frac{1}{2\sigma^2}\|\mathbf{z} - X_{\boldsymbol{\gamma}^{(j=0)}}\boldsymbol{\beta}\|_2^2\right],\end{aligned}$$

and $\boldsymbol{\gamma}^{(j=1)}$ denotes the vector $\boldsymbol{\gamma}$ with its jth component set to 1 (analogously for $\boldsymbol{\gamma}^{(j=0)}$).

4 Overview of Analysis

4.1 Missing Data Imputation

Missing values in protein expression levels were imputed using a two-tier median strategy. For a given protein variable and mouse:

- If fewer than 7 out of 15 replicate measurements were missing, the missing entries were replaced by the median of the non-missing replicate values for that mouse.
- If at least 7 replicate measurements were missing, the missing entries were replaced by the median of all non-missing values within the corresponding experimental class (as defined by genotype, behavior, and treatment).

This method aims to preserve within-mouse information when sufficient replicates are available and otherwise borrow "strength" from the class-level distribution.

4.2 Bayesian Variable Selection Analysis

After imputation, the Bayesian spike-and-slab model described in Section 3 was fitted separately to selected pairs of experimental classes to identify discriminative protein covariates. Covariate relevance was quantified by the *posterior inclusion probability* (PIP):

$$\text{PIP}_j = \Pr(\gamma_j = 1 \mid \text{data}),$$

and proteins were ranked in descending order of PIP.

In addition to individual proteins, we evaluated the relevance of protein pairs and triplets. For a subset $\mathcal{S} \subseteq \{1, \ldots, p\}$ with $|\mathcal{S}| \in \{2, 3\}$, the corresponding inclusion probability was computed as

$$\text{PIP}_{\mathcal{S}} = \Pr\left(\gamma_j = 1 \text{ for all } j \in \mathcal{S} \mid \text{data}\right),$$

and the subsets were ranked in descending order of PIP.

Although there are $\binom{8}{2} = 28$ possible class comparisons, only biologically interpretable and previously reported contrasts were analyzed, following Higuera et al. [2015]. For clarity and consistency, we assign concise labels to each comparison (e.g., `NL`, `FL`, `RL`), which are used throughout the results and discussion. Their study provided qualitative lists of relevant proteins, whereas our analysis yields quantitative posterior rankings of single proteins, protein pairs, and protein triplets.

Table 1: Group comparisons and biological relevance.

Groups	Label	Biological interpretation
c-CS-s vs. c-SC-s	NL	Effects of CFC training in saline injected controls (normal learning)
c-CS-m vs. c-SC-m	NLm	Effects of CFC training in memantine injected controls (normal learning + memantine)
c-SC-m vs. c-SC-s	Base_m	Effects of memantine on control baseline
c-CS-m vs. c-CS-s	Final_m	Effects of memantine on control final conditions
t-CS-s vs. t-SC-s	FL	Effects of CFC training in saline injected Ts65Dn (failed learning)
t-CS-m vs. t-SC-m	RL	Effects of CFC training in memantine-injected Ts65Dn (rescued learning)
t-SC-m vs. t-SC-s	TriBase	Effects of memantine on trisomy baseline
t-CS-m vs. t-CS-s	TriFinal	Effects of memantine on Ts65Dn final conditions (RL vs. FL)
t-SC-s vs. c-SC-s	SC_diff	Initial trisomy vs. control differences

Cross-validation procedure. To estimate misclassification error, a five-fold cross-validation was conducted. The dataset was partitioned into five mutually exclusive folds of approximately equal size. For each fold:

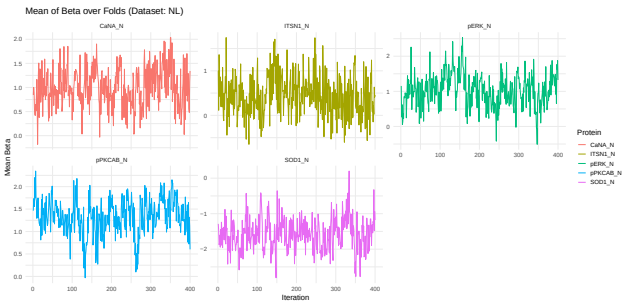
- The model was trained on four folds and evaluated on the remaining fold.
- Misclassification error was computed on the held-out observations.

This process was repeated so that each observation served once as test data. Reported misclassification errors and posterior inclusion probabilities (PIPs) were averaged over the five folds to obtain aggregate estimates.

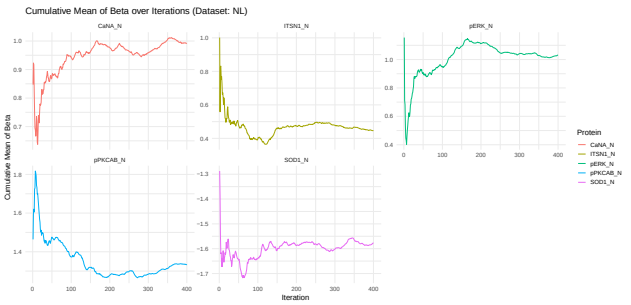
5 Results

5.1 McMC Diagnostics

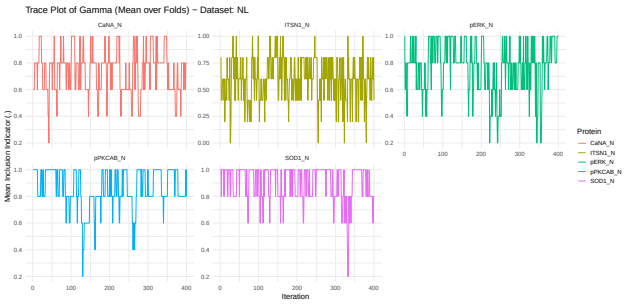
Convergence and stability of the posterior estimates were assessed using trace plots and cumulative mean plots of β_j, γ_j, and σ^2. Given the dimensionality of the model ($p = 77$ proteins), 5 cross-validation folds, and 9 class comparisons, exhaustive visualization for all parameters is infeasible. We therefore report diagnostics for the first comparison (normal learning, c-CS-s vs. c-SC-s), restricted to the five covariates with the highest average posterior inclusion probabilities (PIPs) across folds. These parameters are representative of the most relevant subset of the model and sufficient to evaluate stability. Analogous diagnostics were performed for all remaining comparisons, and no substantial convergence or stability issues were detected.



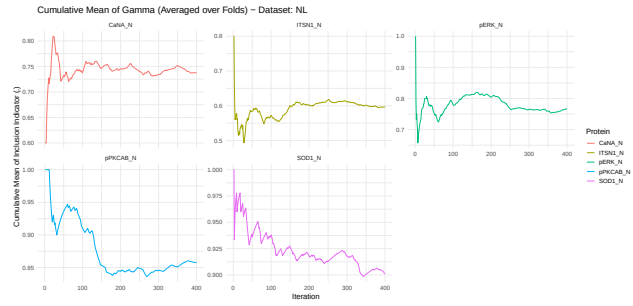
(a) Trace plots of β_j for the five highest-PIP covariates.



(b) Cumulative mean plots of β_j for the same covariates.

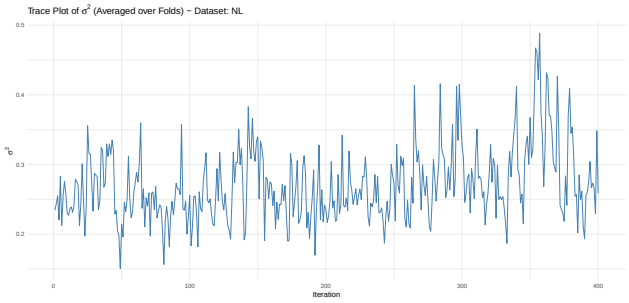


(c) Trace plots of γ_j for the five highest-PIP covariates.

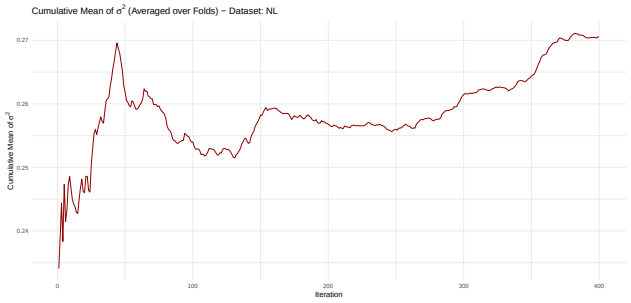


(d) Cumulative mean plots of γ_j for the same covariates.

Figure 1: Trace and Cumulative Mean plots for the normal learning comparison (c-CS-s vs. c-SC-s). The plots are shown for β_j, γ_j corresponding to the five covariates with the highest posterior inclusion probabilities.



(a) Trace plot of σ^2.



(b) Cumulative mean plot of σ^2.

Figure 2: Trace and Cumulative Mean plots for the normal learning comparison (c-CS-s vs. c-SC-s). The plots are shown for σ^2.

5.2 Posterior Summary

For each dataset, we report:

1. The posterior inclusion probability for each protein.
2. The posterior mean and 95% credible interval (CrI) for each β_j conditional on $\gamma_j = 1$.
3. The cross-validated misclassification error.

Tables 2−10 display the top five single-protein covariates ranked by their PIPs, along with fold-specific posterior summaries of β_j. Tables 11−14 summarize the best-performing protein subsets for each dataset, reporting the average misclassification error and posterior inclusion probability (Γ) across folds.

Additionally, for all datasets, we present in Figure 3 the posterior density estimates of the top five covariates (ranked by PIP) for the first cross-validation fold, illustrating the shape of their marginal posterior distributions.

Table 2: Top 5 single covariates for the `NL` dataset ranked by posterior inclusion probability (PIP).

Variable	**PIP**	β_j **(95% CrI)**				
		Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
pPKCAB_N	0.878	1.13 [−0.15, 2.51]	1.29 [0.14, 2.70]	1.91 [0.53, 3.52]	1.50 [0.17, 3.02]	1.49 [0.24, 2.96]
SOD1_N	0.872	−1.72 [−3.36, −0.31]	−1.55 [−3.24, 0.13]	−1.40 [−2.81, 0.19]	−1.70 [−3.54, −0.06]	−2.25 [−3.73, −0.63]
pERK_N	0.7775	1.64 [−0.10, 3.51]	1.60 [−0.11, 3.23]	0.92 [−0.40, 2.30]	1.51 [−0.22, 3.18]	1.24 [−0.72, 3.12]
CaNA_N	0.758	1.41 [0.07, 2.94]	1.54 [0.20, 3.10]	0.65 [−0.85, 2.21]	1.41 [0.02, 2.70]	1.42 [−0.09, 2.85]
BRAF_N	0.5865	0.91 [−0.52, 2.85]	1.01 [−0.96, 2.86]	0.33 [−1.33, 1.83]	1.07 [−0.61, 2.83]	0.69 [−1.10, 2.60]

Table 3: Top 5 single covariates for the `NLm` dataset ranked by posterior inclusion probability (PIP).

Variable	**PIP**	β_j **(95% CrI)**				
		Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
pCAMKII_N	0.950	−1.32 [−2.35, −0.46]	−0.90 [−1.78, −0.25]	−1.89 [−3.14, −0.65]	−0.88 [−1.35, −0.23]	−1.20 [−1.96, −0.42]
CaNA_N	0.813	1.72 [0.14, 3.25]	1.16 [−0.56, 2.75]	1.38 [0.04, 2.73]	1.59 [0.38, 2.90]	1.26 [−0.35, 3.04]
pPKCAB_N	0.798	1.34 [−0.09, 2.89]	1.53 [0.12, 3.23]	0.88 [−0.48, 2.14]	1.15 [−0.01, 2.42]	1.16 [−0.40, 2.55]
SOD1_N	0.678	−1.41 [−3.31, 0.54]	−0.97 [−2.73, 0.74]	−1.10 [−2.78, 0.78]	−1.65 [−3.16, 0.32]	−0.88 [−2.61, 0.82]
pERK_N	0.649	1.14 [−0.54, 2.97]	1.21 [−0.60, 3.21]	0.73 [−1.23, 2.44]	1.28 [−0.56, 3.00]	0.89 [−0.79, 2.82]

Table 4: Top 5 single covariates for the `Base_m` dataset ranked by posterior inclusion probability (PIP).

Variable	**PIP**	β_j **(95% CrI)**				
		Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**
NR2A_N	0.5485	−0.65 [−2.07, 0.95]	−0.57 [−1.59, 0.80]	−0.80 [−2.32, 0.79]	−0.47 [−2.15, 0.81]	−0.63 [−2.10, 0.88]
pPKCG_N	0.5325	−0.43 [−2.13, 1.38]	−0.13 [−1.68, 1.51]	−0.39 [−2.10, 1.23]	−0.36 [−2.29, 1.38]	−0.45 [−2.45, 1.45]
ERK_N	0.5295	−0.42 [−2.17, 1.11]	−0.35 [−1.74, 1.34]	−0.47 [−2.30, 1.31]	−0.54 [−2.32, 1.24]	−0.45 [−2.10, 1.46]
pPKCAB_N	0.529	−0.43 [−2.19, 1.33]	−0.19 [−2.12, 1.76]	−0.57 [−2.28, 0.98]	−0.43 [−2.23, 1.28]	−0.42 [−2.21, 1.20]
CDK5_N	0.526	−0.19 [−2.20, 1.91]	−0.01 [−1.99, 1.72]	−0.03 [−1.84, 1.88]	0.02 [−1.74, 1.67]	−0.13 [−2.20, 1.78]

Table 5: Top 5 single covariates for the `Final_m` dataset ranked by posterior inclusion probability (PIP).

Variable	PIP	β_j (95% CrI)				
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
pPKCAB_N	0.974	−1.60 [−2.71, −0.63]	−0.88 [−1.37, −0.37]	−0.94 [−1.56, −0.42]	−1.02 [−1.79, −0.25]	−1.28 [−2.41, −0.31]
pGSK3B_Tyr216_N	0.900	2.08 [0.18, 3.75]	2.10 [0.53, 3.64]	2.07 [0.69, 3.75]	2.13 [0.34, 4.00]	1.20 [−0.38, 2.94]
IL1B_N	0.824	1.80 [0.09, 3.51]	1.16 [−0.49, 2.95]	2.43 [0.86, 3.98]	3.04 [1.53, 4.86]	0.96 [−0.88, 2.95]
CaNA_N	0.798	−2.69 [−3.79, −1.68]	−1.47 [−2.30, −0.55]	−0.86 [−1.66, −0.08]	−0.57 [−1.64, 0.50]	−0.93 [−1.84, −0.11]
pNUMB_N	0.749	−1.07 [−2.97, 0.92]	−1.67 [−3.44, 0.01]	−2.04 [−3.53, −0.40]	−1.64 [−3.28, 0.12]	−0.50 [−2.24, 1.20]

Table 6: Top 5 single covariates for the `FL` dataset ranked by posterior inclusion probability (PIP).

Variable	PIP	β_j (95% CrI)				
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
pCAMKII_N	1.0000	−1.55 [−2.96, −0.61]	−1.14 [−1.85, −0.51]	−1.29 [−2.06, −0.57]	−0.97 [−1.82, −0.32]	−0.95 [−1.78, −0.31]
pPKCAB_N	0.9165	0.99 [−0.54, 2.45]	2.07 [0.69, 3.41]	2.04 [0.61, 3.67]	1.95 [0.46, 3.42]	2.01 [0.73, 3.41]
SOD1_N	0.7975	−1.05 [−3.16, 0.80]	−1.88 [−3.39, −0.25]	−1.74 [−3.53, 0.01]	−1.92 [−3.54, −0.33]	−1.67 [−3.31, −0.22]
pELK_N	0.6375	0.76 [−1.00, 2.49]	1.13 [−0.24, 2.75]	0.82 [−0.81, 2.22]	1.19 [−0.27, 2.71]	1.08 [−0.34, 2.51]
CaNA_N	0.6255	0.86 [−0.85, 2.69]	1.00 [−0.82, 2.60]	0.78 [−0.63, 2.44]	1.06 [−0.61, 3.02]	0.96 [−0.57, 2.62]

Table 7: Top 5 single covariates for the `RL` dataset ranked by posterior inclusion probability (PIP).

Variable	PIP	β_j (95% CrI)				
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
CaNA_N	0.9410	1.66 [−0.05, 3.18]	1.95 [0.43, 3.44]	2.11 [0.67, 3.56]	1.87 [0.55, 3.31]	1.82 [0.31, 3.28]
SOD1_N	0.8145	−1.32 [−2.98, 0.35]	−1.51 [−3.24, 0.05]	−1.25 [−2.91, 0.37]	−1.54 [−3.04, −0.02]	−1.70 [−3.23, −0.22]
pPKCAB_N	0.7685	1.23 [0.06, 2.57]	1.31 [−0.09, 2.54]	0.96 [−0.30, 2.32]	1.61 [0.11, 2.98]	1.21 [−0.18, 2.64]
pERK_N	0.7465	1.57 [−0.24, 3.32]	1.30 [−0.36, 2.97]	1.51 [−0.19, 3.33]	1.35 [−0.30, 2.94]	1.47 [−0.18, 3.06]
Ubiquitin_N	0.6245	−0.91 [−2.55, 0.53]	−0.61 [−2.25, 1.07]	−1.01 [−2.50, 0.70]	−0.89 [−2.61, 0.61]	−0.56 [−2.27, 0.92]

Table 8: Top 5 single covariates for the `TriBase` dataset ranked by posterior inclusion probability (PIP).

Variable	PIP	β_j (95% CrI)				
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
pPKCG_N	1.0000	−2.17 [−3.41, −1.06]	−2.43 [−3.63, −1.40]	−2.47 [−3.64, −1.33]	−2.34 [−3.71, −1.15]	−2.23 [−3.45, −1.12]
pPKCAB_N	0.7605	−1.07 [−2.46, 0.14]	−1.48 [−3.06, 0.22]	−1.50 [−3.19, 0.09]	−1.39 [−2.93, −0.05]	−1.32 [−2.82, 0.16]
PSD95_N	0.7335	1.17 [0.05, 2.37]	0.87 [−0.61, 2.34]	1.42 [0.01, 2.66]	0.97 [−0.20, 2.41]	1.22 [−0.02, 2.55]
Ubiquitin_N	0.6745	1.32 [−0.28, 2.92]	0.99 [−0.49, 2.89]	0.88 [−0.53, 2.46]	0.93 [−1.01, 2.80]	0.99 [−0.65, 2.73]
pP70S6_N	0.6565	−1.09 [−2.75, 0.57]	−0.90 [−2.63, 0.90]	−0.86 [−2.87, 0.75]	−1.16 [−3.09, 0.62]	−0.90 [−2.64, 0.53]

Table 9: Top 5 single covariates for the `TriFinal` dataset ranked by posterior inclusion probability (PIP).

Variable	PIP	β_j (95% CrI)				
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
ERK_N	1.0000	−2.19 [−3.41, −1.05]	−1.81 [−3.24, −0.68]	−1.61 [−2.79, −0.55]	−1.62 [−2.62, −0.63]	−2.35 [−3.52, −1.23]
NR2A_N	0.8950	1.27 [0.19, 2.33]	1.34 [0.38, 2.69]	1.37 [0.35, 2.43]	0.96 [−0.06, 2.03]	1.45 [0.37, 2.44]
S6_N	0.8825	−2.02 [−3.80, −0.31]	−2.27 [−4.00, −0.85]	−1.90 [−3.42, −0.19]	−1.44 [−3.50, 0.44]	−1.74 [−3.39, −0.02]
pCAMKII_N	0.8755	0.66 [0.22, 1.25]	0.71 [0.31, 1.28]	0.33 [0.01, 0.73]	0.83 [0.30, 1.41]	0.42 [0.09, 0.75]
CaNA_N	0.8010	1.21 [−0.16, 2.60]	1.37 [−0.02, 2.65]	1.84 [0.61, 3.16]	1.08 [−0.29, 2.30]	0.83 [−0.43, 1.92]

Table 10: Top 5 single covariates for the `SC_diff` dataset ranked by posterior inclusion probability (PIP).

Variable	PIP	β_j (95% CrI)				
		Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
pPKCG_N	0.9590	1.42 [0.51, 2.42]	1.55 [0.72, 2.44]	2.17 [1.26, 3.34]	1.67 [0.77, 2.88]	1.33 [0.49, 2.25]
NR2A_N	0.8415	-1.38 [-2.75, -0.26]	-1.15 [-2.26, -0.24]	-0.31 [-1.57, 0.62]	-1.37 [-2.45, -0.30]	-1.05 [-1.93, -0.11]
ITSN1_N	0.8150	2.10 [0.34, 3.90]	1.50 [-0.27, 3.27]	0.35 [-1.27, 1.96]	1.49 [-0.39, 3.11]	1.78 [-0.10, 3.60]
IL1B_N	0.7220	-1.49 [-3.35, 0.36]	-1.62 [-3.15, 0.27]	-0.53 [-2.49, 1.49]	-1.24 [-2.99, 0.52]	-1.36 [-3.09, 0.27]
ERK_N	0.6575	1.04 [-0.20, 2.16]	1.24 [-0.12, 2.50]	0.07 [-1.33, 1.24]	1.49 [0.22, 2.91]	0.67 [-0.51, 1.87]

Table 11: Top 5 discriminative variable subsets for the NL comparison (c-CS-s vs. c-SC-s), sorted by increasing average misclassification error. Only subsets with PIP > 0.75 are considered.

Variables	Misclassification Error	PIP
pERK_N, SOD1_N	0.022	0.825
pCAMKII_N, pPKCAB_N, SOD1_N	0.093	0.758
NR1_N, pPKCAB_N, SOD1_N	0.156	0.756
pPKCAB_N, AKT_N, SOD1_N	0.156	0.773
pPKCAB_N, SOD1_N, Ubiquitin_N	0.156	0.778

Table 12: Top 5 discriminative variable subsets for the NLm comparison (c-CS-m vs. c-SC-m), sorted by increasing average misclassification error. Only subsets with PIP > 0.75 are considered.

Variables	Misclassification Error	PIP
NR2A_N, pCAMKII_N, pPKCAB_N	0.087	0.776
pCAMKII_N, pPKCAB_N, CaNA_N	0.100	0.853
pCAMKII_N, pPKCAB_N, ADARB1_N	0.150	0.771
pCAMKII_N, pPKCAB_N, ERK_N	0.171	0.772
pCAMKII_N, pERK_N, pPKCAB_N	0.210	0.799

Table 13: Top 5 discriminative variable subsets for the Base_m comparison (c-SC-m vs. c-SC-s), sorted by increasing average misclassification error. Only subsets with PIP > 0.53 are considered.

Variables	Misclassification Error	PIP
NR2A_N	0	0.548
NR2A_N, ERK_N	0	0.539
NR2A_N, pPKCAB_N	0	0.539
NR2A_N, GFAP_N	0	0.535
NR2A_N, pAKT_N	0	0.534

Table 14: Top 5 discriminative variable subsets for Final_m comparison (c-CS-m vs. c-CS-s), sorted by increasing average misclassification error. Only subsets with PIP > 0.75 are considered.

Variables	Misclassification Error	PIP
pPKCAB_N, P70S6_N, pGSK3B_Tyr216_N	0	0.789
pPKCAB_N, TRKA_N, nNOS_N	0	0.781
pNR2B_N, pPKCAB_N, S6_N	0	0.775
pPKCAB_N, pP70S6_N, IL1B_N	0	0.773
pPKCAB_N, pNUMB_N, pGSK3B_N	0	0.765

Table 15: Top 5 discriminative variable subsets for the FL comparison (t-CS-s vs. t-SC-s), sorted by increasing average misclassification error. Only subsets with PIP > 0.7 are considered.

Variables	Misclassification Error	PIP
pERK_N, SOD1_N	0.052	0.708
DYRK1A_N, pCAMKII_N, pERK_N	0.063	0.706
pCAMKII_N, pELK_N, H3MeK4_N	0.063	0.708
pCAMKII_N, pERK_N, H3AcK18_N	0.063	0.703
pCAMKII_N, pERK_N, P3525_N	0.063	0.705

Table 16: Top 5 discriminative variable subsets for the RL comparison (t-CS-m vs. t-SC-m), sorted by increasing average misclassification error. Only subsets with PIP > 0.75 are considered.

Variables	Misclassification Error	PIP
pERK_N, SOD1_N	0	0.780
SOD1_N, Ubiquitin_N, CaNA_N	0.144	0.793
pNR2B_N, SOD1_N, CaNA_N	0.204	0.755
SOD1_N, ADARB1_N, CaNA_N	0.296	0.759
Bcatenin_N, SOD1_N, CaNA_N	0.322	0.768

Table 17: Top 5 discriminative variable subsets for the TriBase comparison (t-SC-m vs. t-SC-s), sorted by increasing average misclassification error. Only subsets with PIP > 0.7 are considered.

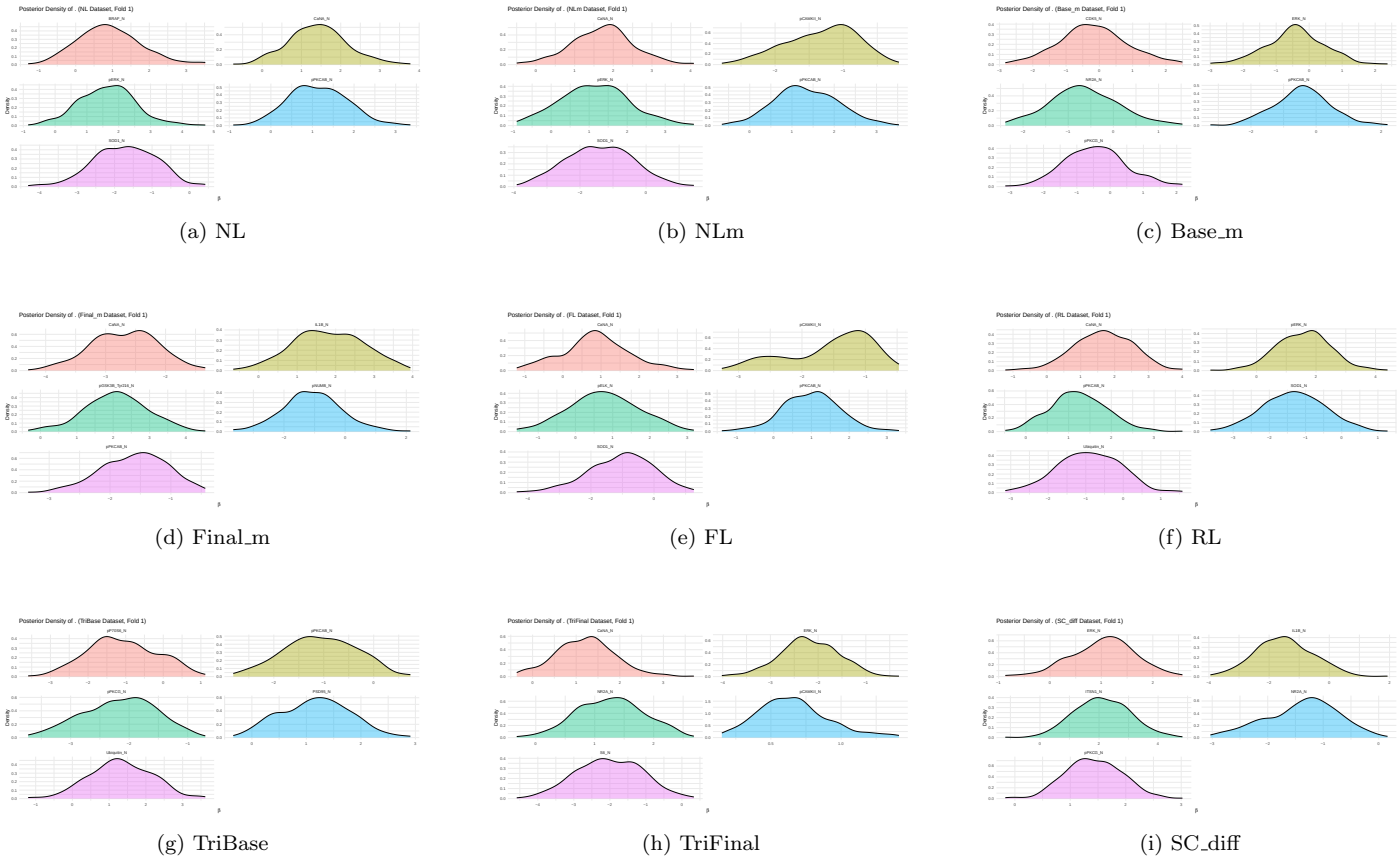
Variables	Misclassification Error	PIP
MTOR_N, pPKCG_N, PSD95_N	0.0926	0.736
SOD1_N, pPKCG_N, PSD95_N	0.139	0.718
pPKCG_N, PSD95_N, Ubiquitin_N	0.159	0.803
pNR2A_N, pPKCG_N, S6_N	0.167	0.709
pNR2A_N, pPKCG_N, BAD_N	0.167	0.703

Table 18: Top 5 discriminative variable subsets for the TriFinal comparison (t-CS-m vs. t-CS-s), sorted by increasing average misclassification error. Only subsets with PIP > 0.75 are shown.

Variables	Misclassification Error	PIP
ERK_N, pCASP9_N, Ubiquitin_N	0	0.771
pBRAF_N, ERK_N, Ubiquitin_N	0	0.768
NR2A_N, TRKA_N, S6_N	0	0.764
ERK_N, pGSK3B_N, Ubiquitin_N	0	0.763
ERK_N, P70S6_N, Ubiquitin_N	0	0.763

Table 19: Top 5 discriminative variable subsets for the SC_diff comparison (t-SC-s vs. c-SC-s), sorted by increasing average misclassification error. Only subsets with PIP > 0.75 are shown.

Variables	Misclassification Error	PIP
pPKCG_N, IL1B_N, PSD95_N	0.0778	0.772
pNR2B_N, pPKCG_N, IL1B_N	0.111	0.762
NR2A_N, pCAMKII_N, pPKCG_N	0.125	0.778
NR2A_N, Bcatenin_N, pPKCG_N	0.241	0.790
ITSN1_N, NR2A_N, pPKCG_N	0.241	0.872



(a) NL (b) NLm (c) Base_m

(d) Final_m (e) FL (f) RL

(g) TriBase (h) TriFinal (i) SC_diff

Figure 3: Posterior density estimates of the top five covariates (ranked by PIP) for the first cross-validation fold of each comparison.

5.3 Summary of Findings

Across most comparisons, subsets of covariates with an average posterior inclusion probability (PIP) exceeding 0.75 were sufficient to yield at least one combination with an average misclassification error below 0.1. For the `FL` and `TriBase`

datasets, this threshold relaxed to PIP > 0.7, whereas for `Base_m`, a lower cutoff of PIP > 0.53 was required to obtain discriminative subsets. In the `Base_m`, `Final_m`, `RL`, and `TriFinal` comparisons, at least one subset achieved an average misclassification error of 0, indicating perfect cross-validated classification.

In nearly all comparisons, at least one covariate exhibits an average PIP exceeding 0.85, signifying strong marginal posterior evidence for inclusion. The sole exception is the `Base_m` dataset, where the maximum average PIP attains only 0.5485, motivating the need to consider subsets with lower inclusion probabilities. Notably, the covariates `pCAMKII_N`, `pPKCG_N`, and `ERK_N` achieve an average PIP of 1.0 in the `FL`, `TriBase`, and `TriFinal` comparisons, respectively.

For each comparison, the posterior mean of the regression coefficient (β_j) for each top-ranked protein in a given fold consistently lies within the 95% credible interval (CrI) of the corresponding posterior distributions from the other folds, indicating cross-validation stability.

Overall, these findings are consistent with the results reported in Higuera et al. [2015]

6 Conclusion

Using a Bayesian spike-and-slab framework, we identified subsets of proteins that effectively discriminate between biologically relevant experimental groups. This method not only reproduced the qualitative findings of Higuera et al. [2015] but also provided a quantitative measure of variable relevance through posterior inclusion probabilities (PIPs). The model yielded subsets of proteins with high discriminative power, demonstrating that our model could capture biologically meaningful patterns in high-dimensional proteomic data.

References

James H. Albert and Siddhartha Chib. Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88(422):669–679, 1993. ISSN 01621459, 1537274X. URL `http://www.jstor.org/stable/2290350`.

Clara Higuera, Katheleen J. Gardiner, and Krzysztof J. Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLoS ONE*, 10, 2015. URL `https://api.semanticscholar.org/CorpusID:7955336`.

Lynn Kuo and Bani Mallick. Variable selection for regression models. *Sankhyā: The Indian Journal of Statistics, Series B (1960-2002)*, 60(1):65–81, 1998. ISSN 05815738. URL `http://www.jstor.org/stable/25053023`.

R Code

```
library(dplyr)
library(truncnorm)
library(MASS)
library(parallel)
library(purrr)
set.seed(2409)
file_path <- "" #Enter file path, after converting data to csv format
data <- read.csv(file_path, stringsAsFactors = TRUE)
data$Mouse <- sub("_(\\d+)$", "", data$MouseID)
data$Measurement <- as.integer(sub(".*_(\\d+)$", "\\1", data$MouseID))
class_map <- unique(data[, c("Mouse", "class")])
imputed <- data
protein_cols <- names(data)[2:78]
for (prot in protein_cols) {
  for (mouse in unique(data$Mouse)) {
    sub_mouse <- data[data$Mouse == mouse, ]
    miss_idx <- which(is.na(sub_mouse[[prot]]))
    n_miss <- length(miss_idx)
    if (n_miss == 0) next
    if (n_miss < 7) {
      med_val <- median(sub_mouse[[prot]], na.rm = TRUE)
    } else {
      class_id <- class_map$class[class_map$Mouse == mouse]
      med_val <- median(data[data$class == class_id, prot], na.rm = TRUE)
    }
    imputed[imputed$Mouse == mouse & is.na(imputed[[prot]]), prot] <- med_val
  }
}
comparisons <- list(
  "NL"   = c("c-CS-s", "c-SC-s"),
  "NLm"  = c("c-CS-m", "c-SC-m"),
  "Base_m" = c("s-SC-m", "c-CS-s"),
  "Final_m" = c("c-CS-m", "c-CS-s"),
  "FL"   = c("t-CS-s", "t-SC-s"),
  "RL"   = c("t-CS-m", "t-SC-m"),
  "TriBase" = c("t-SC-m", "t-SC-s"),
  "TriFinal" = c("t-CS-m", "t-CS-s"),
  "SC_diff"  = c("t-SC-s", "c-SC-s")
)
datasets <- list()
for (name in names(comparisons)) {
  grp <- comparisons[[name]]
  df_sub <- imputed[imputed$class %in% grp, ]
  df_sub <- droplevels(df_sub)
  df_sub$group <- ifelse(df_sub$class == grp[1], 1, 0)
  drop_cols <- c("Genotype", "Treatment", "Behavior",
                 "Mouse", "Measurement", "MouseID",
                 "class")
  df_sub <- df_sub[, !(names(df_sub) %in% drop_cols)]
  datasets[[name]] <- df_sub
}
gibbs_sampler_KM <- function(X, y, n_iter = 5000, beta0 = NULL, k = 1.0, pi_vec = NULL,
                             alpha0 = 1.0, eta0 = 1.0) {
  N <- nrow(X)
  p <- ncol(X)
  if (is.null(beta0)) beta0 <- rep(0, p)
  if (is.null(pi_vec)) pi_vec <- rep(0.5, p)
  gamma <- rep(1L, p)
```

```
  beta <- beta0
  sigma2 <- 1.0
  D0_inv <- diag(1 / k^2, p)
  samples <- list(beta = vector("list", n_iter),
                  gamma = vector("list", n_iter),
                  sigma2 = numeric(n_iter))
  for (it in seq_len(n_iter)) {
    mu <- X %*% (gamma * beta)
    z <- numeric(N)
    for (i in seq_len(N)) {
      if (y[i] == 1) {
        z[i] <- truncnorm::rtruncnorm(size = 1, mean = mu[i], sd = sqrt(sigma2),
                                      a = 0, b = Inf)
      } else {
        z[i] <- truncnorm::rtruncnorm(size = 1, mean = mu[i], sd = sqrt(sigma2),
                                      a = -Inf, b = 0)
      }
    }
    X_star <- sweep(X, 2, gamma, `*`)
    D_post <- solve(D0_inv + (crossprod(X_star) / sigma2))
    beta_post <- D_post %*% (D0_inv %*% beta0 + (crossprod(X_star, z) / sigma2))
    beta <- as.numeric(MASS::mvrnorm(1, mu = beta_post, Sigma = D_post))
    resid <- z - X_star %*% beta
    shape_post <- alpha0 + N / 2
    scale_post <- eta0 + sum(resid^2) / 2
    sigma2 <- 1 / rgamma(1, shape = shape_post, rate = scale_post)
    for (j in seq_len(p)) {
      gamma_temp <- gamma
      gamma_temp[j] <- 1L
      mu1 <- X %*% (gamma_temp * beta)
      c_j <- pi_vec[j] * exp(((-0.5) / sigma2) * sum((z - mu1)^2))
      gamma_temp[j] <- 0L
      mu0 <- X %*% (gamma_temp * beta)
      d_j <- (1 - pi_vec[j]) * exp(((-0.5) / sigma2) * sum((z - mu0)^2))
      gamma[j] <- rbinom(1, 1, c_j / (c_j + d_j))
    }
    samples$beta[[it]] <- beta
    samples$gamma[[it]] <- gamma
    samples$sigma2[it] <- sigma2
  }
  return(samples)
}

n_cores <- max(1, detectCores())
cl <- makeCluster(n_cores)
clusterEvalQ(cl, {
  library(dplyr)
  library(purrr)
})
clusterExport(cl, c("datasets", "gibbs_sampler_KM"))

cv_results <- parLapply(cl, names(datasets), function(dname) {
  data <- datasets[[dname]]
  y_all <- as.numeric(data$group)
  X_all <- as.matrix(data[, !(names(data) %in% c("group")), drop = FALSE])
  if ("pS6_N" %in% colnames(X_all)) {
    X_all <- X_all[, !(colnames(X_all) == "pS6_N"), drop = FALSE]
  }
  N <- nrow(X_all)
  folds <- cut(seq_len(N), breaks = 5, labels = FALSE)
```

```
fold_rows <- vector("list", length = 5)
fold_post <- vector("list", length = 5)
for (fold in 1:5) {
  test_idx <- which(folds == fold)
  train_idx <- setdiff(seq_len(N), test_idx)
  X_train <- X_all[train_idx, , drop = FALSE]
  y_train <- y_all[train_idx]
  X_test  <- X_all[test_idx, , drop = FALSE]
  y_test  <- y_all[test_idx]
  res <- gibbs_sampler_KM(
    X_train, y_train,
    n_iter = 5000,
    beta0 = rep(0, ncol(X_train)),
    k = 1.0,
    pi_vec = rep(0.5, ncol(X_train)),
    alpha0 = 1.0,
    eta0 = 1.0
  )
  beta_mat  <- do.call(rbind, res$beta)
  gamma_mat <- do.call(rbind, res$gamma)
  sigma2_vec <- res$sigma2
  idx_keep <- seq(1001, nrow(gamma_mat), by = 10)
  beta_post  <- beta_mat[idx_keep, , drop = FALSE]
  gamma_post <- gamma_mat[idx_keep, , drop = FALSE]
  sigma2_post <- sigma2_vec[idx_keep]
  colnames(beta_post) <- colnames(gamma_post) <- colnames(X_train)
  freq <- colMeans(gamma_post)
  ranked <- data.frame(variable = colnames(X_train), freq = freq)
  top10_single <- head(ranked[order(-ranked$freq), ], 20) #10
  p <- ncol(gamma_post)
  top10_pairs <- data.frame(var1=character(), var2=character(), freq=numeric())
  top10_triplets <- data.frame(var1=character(), var2=character(), var3=character(),
                               freq=numeric())
  if (p >= 2) {
    pair_idx <- combn(p, 2)
    pair_freqs <- apply(pair_idx, 2, function(idx)
      mean(gamma_post[, idx[1]] == 1 & gamma_post[, idx[2]] == 1))
    pair_freqs_sorted <- data.frame(
      var1 = colnames(gamma_post)[pair_idx[1, ]],
      var2 = colnames(gamma_post)[pair_idx[2, ]],
      freq = pair_freqs,
      stringsAsFactors = FALSE
    )[order(-pair_freqs), ]
    top10_pairs <- head(pair_freqs_sorted, 190) #10
  }
  if (p >= 3) {
    triplet_idx <- combn(p, 3)
    triplet_freqs <- apply(triplet_idx, 2, function(idx)
      mean(gamma_post[, idx[1]] == 1 & gamma_post[, idx[2]] == 1 & gamma_post[, idx[3]] == 1))
    triplet_freqs_sorted <- data.frame(
      var1 = colnames(gamma_post)[triplet_idx[1, ]],
      var2 = colnames(gamma_post)[triplet_idx[2, ]],
      var3 = colnames(gamma_post)[triplet_idx[3, ]],
      freq = triplet_freqs,
      stringsAsFactors = FALSE
    )[order(-triplet_freqs), ]
    top10_triplets <- head(triplet_freqs_sorted, 1140) #10
  }

  beta_avg <- sapply(seq_len(p), function(j) {
```

```
      active_idx <- which(gamma_post[, j] == 1)
      if (length(active_idx) > 0) mean(beta_post[active_idx, j]) else 0
    })
    names(beta_avg) <- colnames(X_train)

    models_single  <- lapply(1:nrow(top10_single), function(i) top10_single$variable[i])
    models_pairs   <- lapply(1:nrow(top10_pairs),  function(i)
      c(top10_pairs$var1[i], top10_pairs$var2[i]))
    models_triplet <- lapply(1:nrow(top10_triplets), function(i)
      c(top10_triplets$var1[i], top10_triplets$var2[i], top10_triplets$var3[i]))
    model_combinations <- c(models_single, models_pairs, models_triplet)

    misclass_by_model <- numeric(length(model_combinations))
    model_vars <- character(length(model_combinations))
    for (m in seq_along(model_combinations)) {
      vars <- model_combinations[[m]]
      X_test_sub <- X_test[, vars, drop = FALSE]
      beta_sub <- beta_avg[vars]
      yhat <- as.numeric(X_test_sub %*% beta_sub >= 0)
      misclass_by_model[m] <- mean(yhat != y_test)
      model_vars[m] <- paste(vars, collapse = ", ")
    }

    fold_rows[[fold]] <- data.frame(
      Fold = fold,
      Model = paste0("M", seq_along(model_combinations)),
      Variables = model_vars,
      MisclassError = misclass_by_model,
      stringsAsFactors = FALSE
    )

    fold_post[[fold]] <- list(
      beta_post = beta_post,
      gamma_post = gamma_post,
      sigma2_post = sigma2_post,
      top10_single = top10_single,
      top10_pairs = top10_pairs,
      top10_triplets = top10_triplets,
      model_combinations = model_combinations,
      misclass_by_model = misclass_by_model
    )
  }

  folds_df <- bind_rows(fold_rows)
  avg_by_vars <- folds_df %>%
    group_by(Variables) %>%
    summarise(AvgMisclassError = mean(MisclassError), .groups = "drop") %>%
    arrange(AvgMisclassError)

  list(
    folds_df = folds_df,
    fold_post = fold_post,
    avg_by_vars = avg_by_vars
  )
})

stopCluster(cl)

names(cv_results) <- names(datasets)
```