Methods

In this study, we conducted two simulations to evaluate the performace of LASSO models in identifying important features in high-dimensional data. The first simulation assumes sparsity of the features in the pixel space, while the second simulation assumes sparsity in the frequency space.

The pixel space refers to the original high-dimensional space where each dimension represents a pixel in an image. In this space, features are directly observed and may have inherent correlations. Conversely, the frequency space is a transformed version of the pixel space, obtained through techniques like eigen decomposition.

Suppose X is a column vector representing 256 pixels. Its covariance matrix, Σ, is defined to have an exponential correlation structure, where $\Sigma_{ij} = -\exp(\text{dist}(i,j))$. Here, $\text{dist}(i,j)$ is the distance between the pixels i and j in a 16×16 matrix.

Let V be the matrix of eigenvectors of Σ, with each column representing an eigenvector. We can transform the random vector X into the frequency space by $X_{\text{freq}} = V^T X$. The covariance matrix of X_{freq} is given by $\text{cov}(X_{\text{freq}}) = V^T \Sigma V$, which is a diagonal matrix.

For the simulations, in each iteration, we randomly generate X_{freq} from a multivariate normal distribution with the covariance matrix $\text{cov}(X_{\text{freq}})$. We repeat this process 1000 times. Then, we calculate X as $X = V X_{\text{freq}}$.

In the first simulation, we assume sparsity in the coefficient vectors in the pixel space. The coefficient vector β was specified to have non-zero values exclusively within a central 8×8 region. The response variable y was drawn from a binomial distribution with success probabilities determined by $\eta = X\beta$. The non-zero coefficients in β were chosen such that the probability $p = \frac{1}{1+\exp(-\eta)}$ was uniformly distributed across interval $[0, 1]$.

In the second simulation, we assume sparsity in the coefficient vectors in the frequency space. We defined a sparse coefficient vector b in the frequency space, where most of the 256 entries were zero and a randomly 10% were non-zero. The response variable y was generated similarly to the first simulation, ensuring $p = \frac{1}{1+\exp(-\eta)}$ was evenly distributed.

For both simulations, we fit two models: one using the covariates in the pixel space and another using the covariates in the frequency space. Each dataset, generated in size 1000×256 and representing images of 16×16 pixels, was split into training (80%) and test (20%) sets. The regularization parameter λ was tuned using cross-validation with the default binomial deviance metric. The dataset was divided into 10 folds, with the model trained and validated iteratively across these folds, varying λ. Here we use two λ values, 'lambda.min' is the regularization value that minimizes cross-validated error, and 'lambda.1se' is the largest lambda within one standard error of the minimum.

After selecting the optimal λ, model performance was evaluated using accuracy and AUC metrics. Additionally, a permutation test was conducted 100 times to calculate p-values for each covariate. Across all iterations, we calculated the mean and standard deviation of the metrics, as well as the percentage of significant p-values for each covariate.

Results

Firstly we show how the top 3 eigenvectors and the bottom eigenvector looks like.

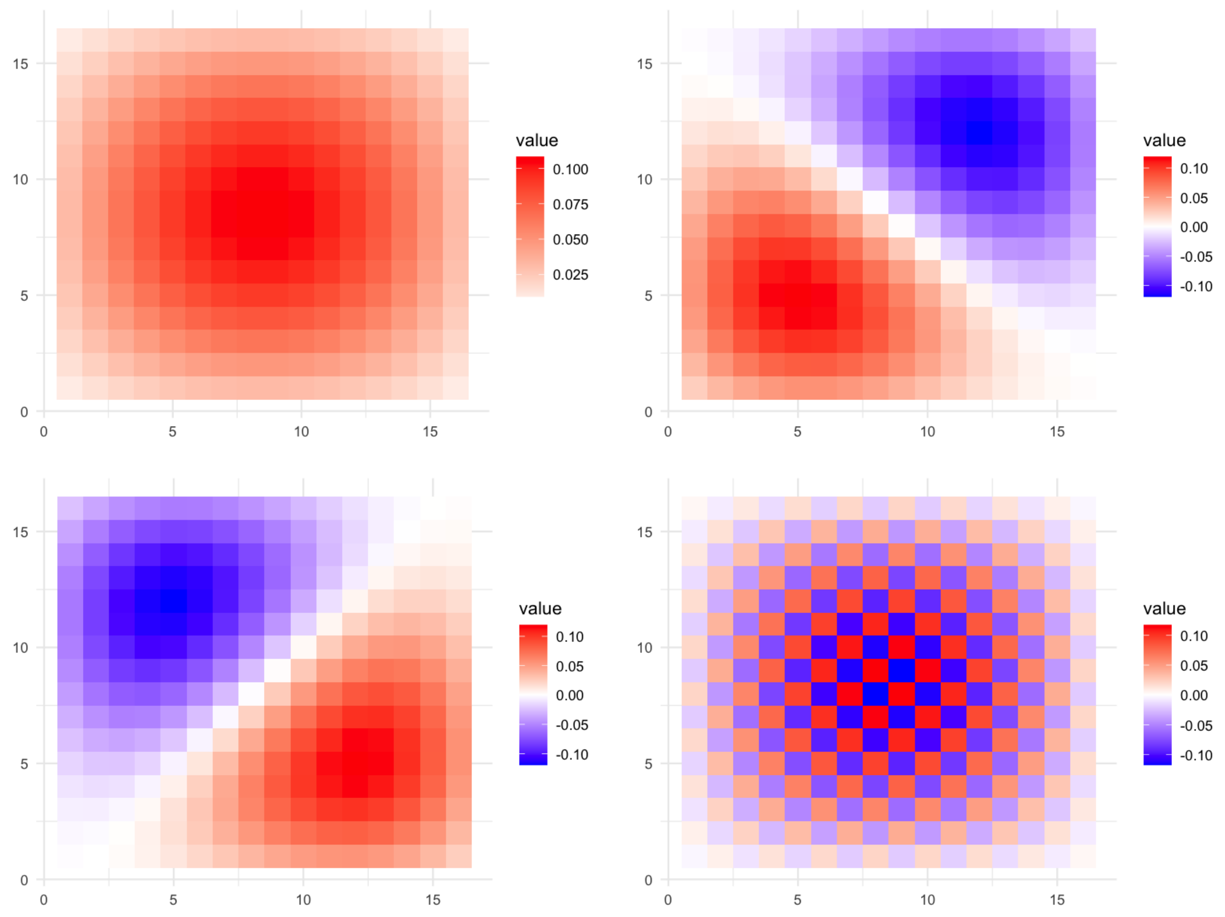


Figure 1: Frequencies associated with the top 3 eigenvalues, as well as the frequency associated with the bottom eigenvalue

In simulation 1, figure 2 shows the distribution of p for different values of β. β were evaluated with value of 0.01, 0.05, 0.1, 0.2 and 1. We can see that when $\beta = 0.1$ does it achieve an even spread of probabilities, thus the effect size of β were set to 0.1.

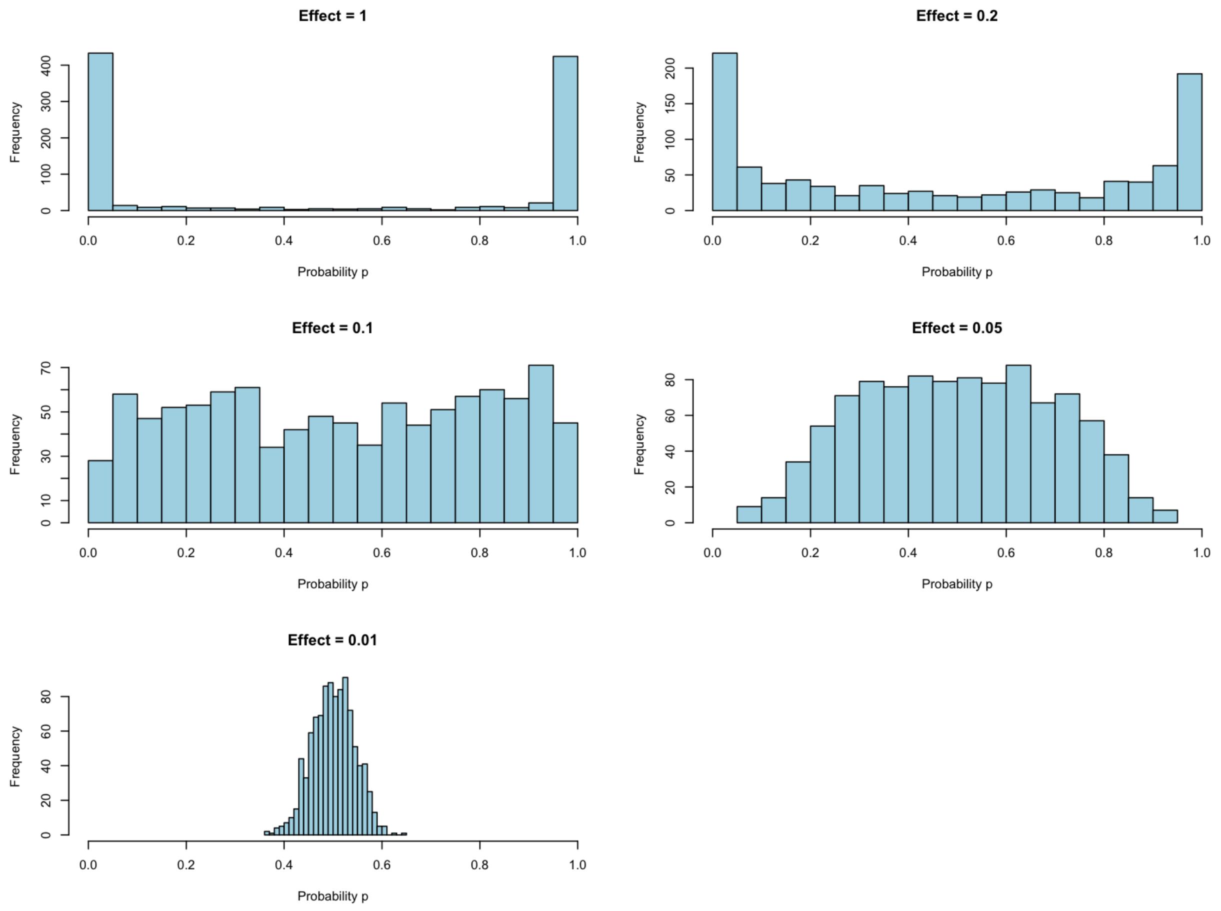


Figure 2: The distribution of p at different β values. $\beta = 0.1$ was chosen for model fitting as it gives the most evenly distributed values.

In simulation 2, figure 3 shows the distribution of p for different values of b. b were evaluated at 0.1, 0.2, 0.4, 0.6, 0.8 and 1. When b equals 0.4 did we achieve an even spread of probabilites.

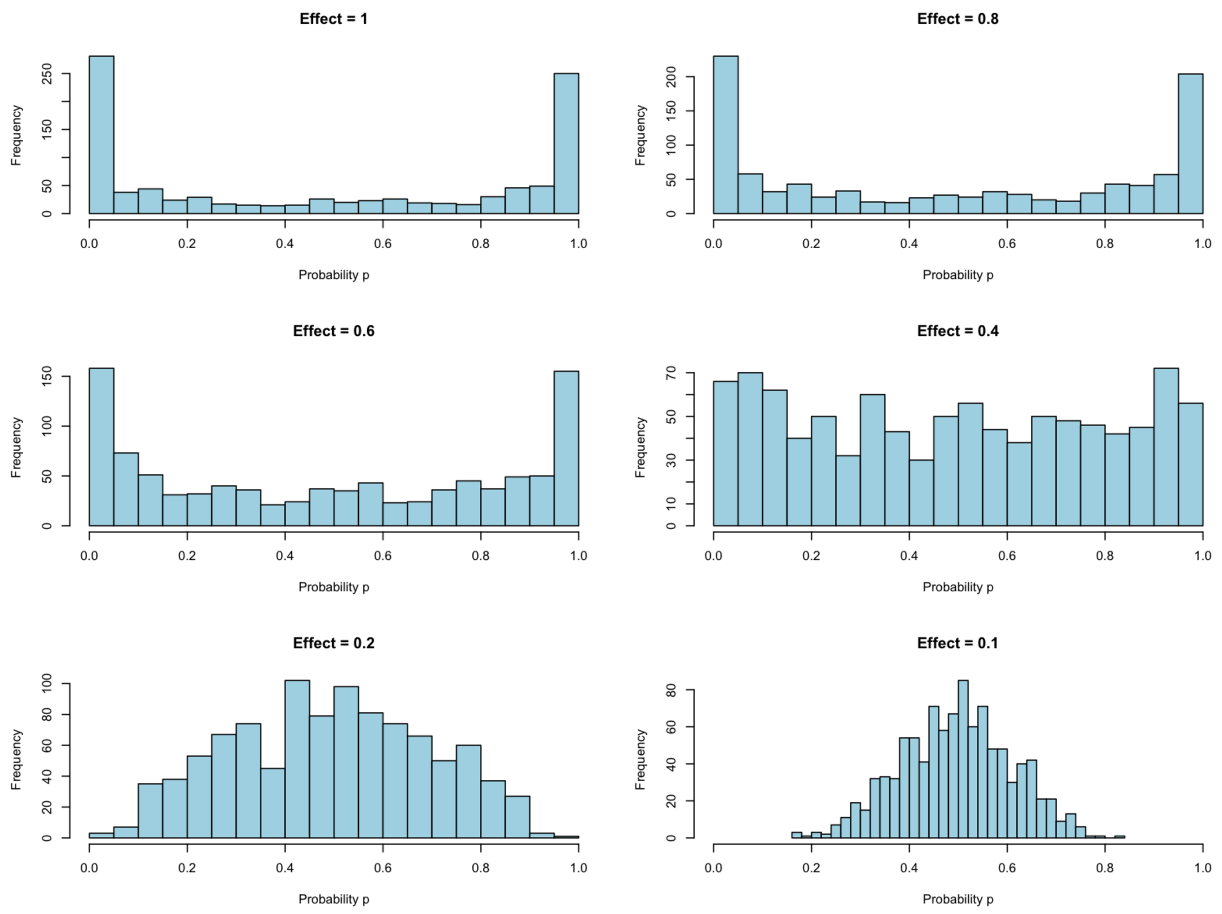


Figure 3: The distribution of p at different b values. $b = 0.4$ was chosen for model fitting as it gives the most evenly distributed values.

We then used the chosen β effect size to generate the coefficient vector in the pixel space β for Simulation 1, and then converted it into the corresponding b in the frequency space. Similarly, we simulated the coefficient vector in the frequency space using the chosen b effect for Simulation 2, and calculated the corresponding β vector. Figure 5 shows the generated coefficient vectors and Figure 4 shows the group mean difference in X and X_{freq} whose corresponding y equals 1 to the group with $y = 0$.

We can see that the heatmaps of covariates match with the heatmaps of coefficients in the corresponding space. Using coefficients and covariates in the pixel space as an example, since only the center of β have non-zero positive values, it means X whose center values are high have higher probability of being assigned with $y = 1$. Similar explanation can be applied to coefficients and coefficients in the pixel space.

Table 1 shows the average AUCs and accuracys calculated by fitting models on the 1000 simulated dataset, on both the frequency sapce and pixel space. We can see that no matter we assume sparsity in coefficient vectors in the pixel space of the frequency space, fitting models on the frequency space provides better performance in AUCs and accuracys. When assuming sparsity in coefficient vectors in the pixel space and using ‘lambda.min‘ as the regularization value, models fitting with covariates in the pixel space (X) achieves an AUC of 0.803 (se =

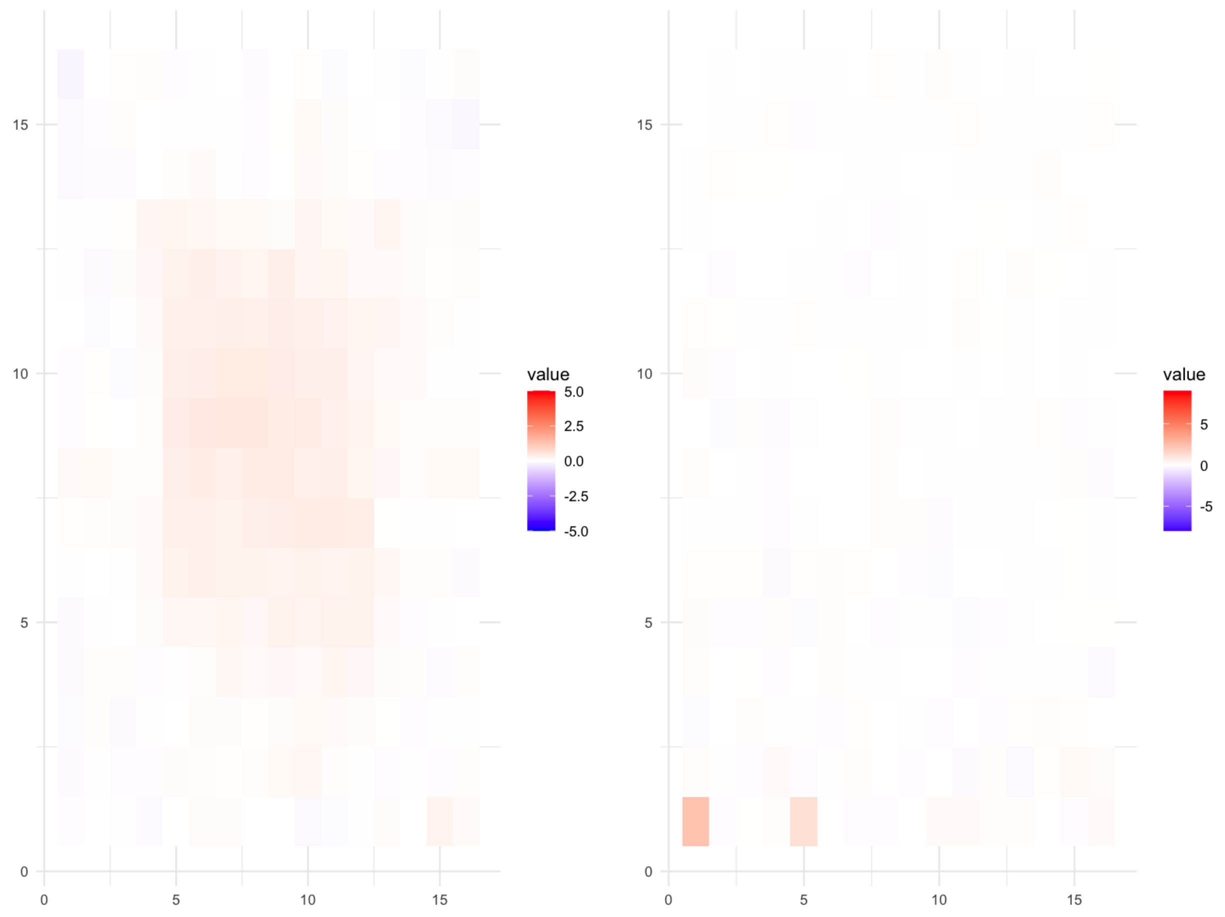


Figure 4: The group mean difference between $y = 1$ and $y = 0$.

0.031), and an accuracy of 72.5% (se = 0.032); while models fitting with covariates in the corresponding frequency space (X_{freq}) achieves a slightly better AUC of 0.827 (se = 0.029) and better accuracy of 74.6% (se = 0.031). We see a similar advantage of fitting models on the frequency space when assuming sparsity in coefficient vectors in the frequency space, and using either ‘lambda.min‘ or ‘lambda.1se‘ does not affect this advantage.

Table 1:

	Model on the pixel space		Model on the freq space	
Lambda	AUC (SE)	Accuracy (SE)	AUC (SE)	Accuracy (SE)
lambda.min	0.803 (0.031)	0.725 (0.032)	0.827 (0.029)	0.746 (0.031)
lambda.1se	0.800 (0.031)	0.723 (0.032)	0.826 (0.029)	0.745 (0.031)
lambda.min	0.756 (0.036)	0.687 (0.035)	0.816 (0.032)	0.735 (0.033)
lambda.1se	0.734 (0.040)	0.669 (0.036)	0.816 (0.032)	0.734 (0.034)

Then, for both Simulation 1 and Simulation 2, we fitted two models, one using covariates on the frequency space, another using covariates on the pixel space. In

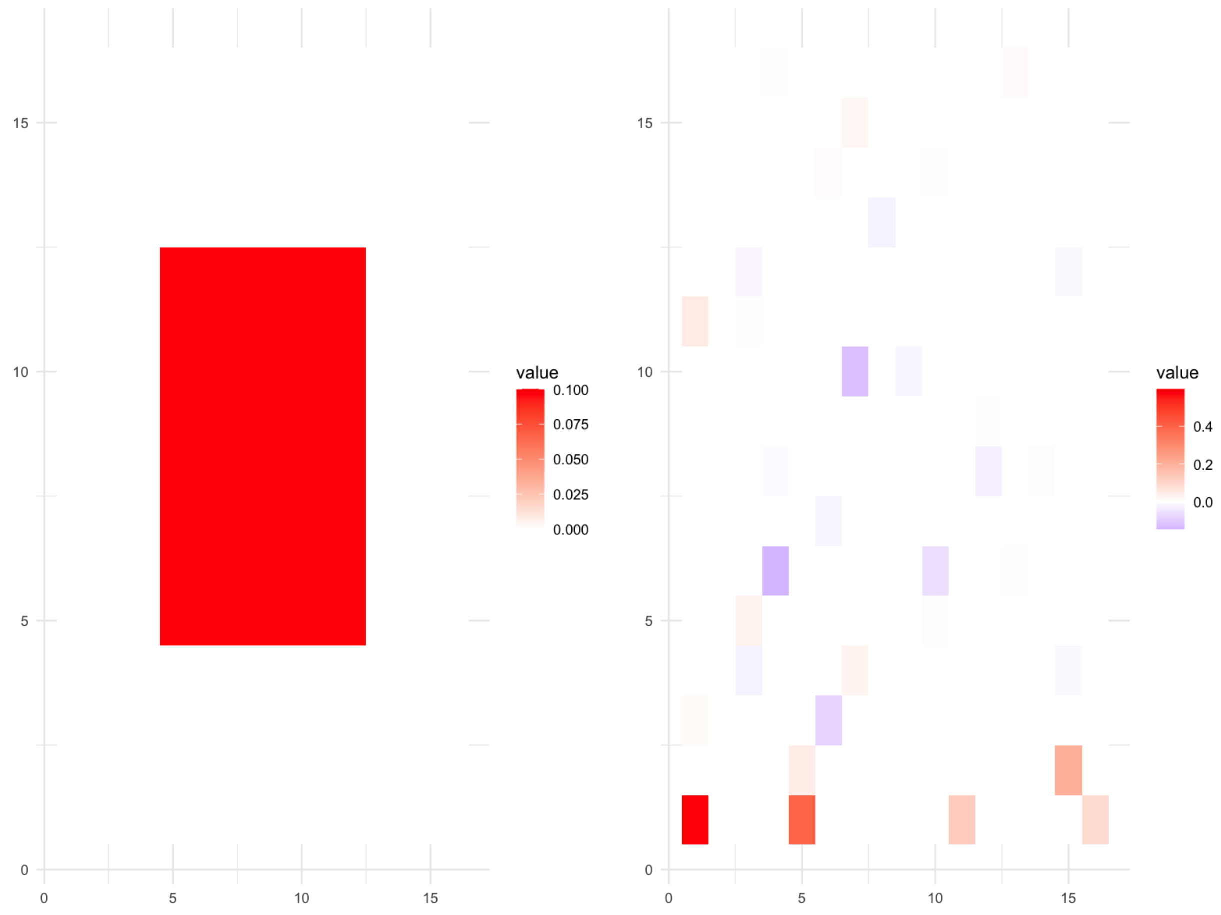


Figure 5: Coefficients for Simulation 1.

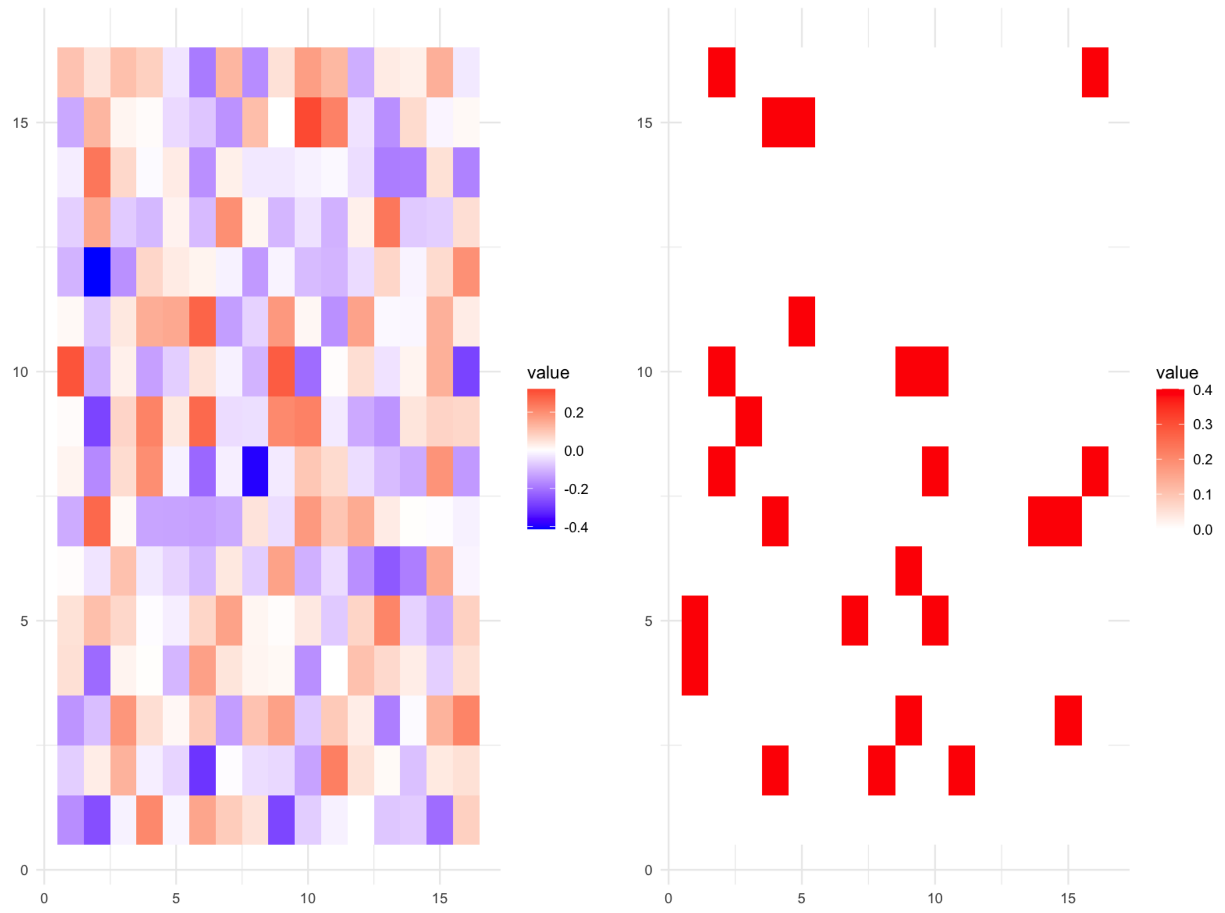
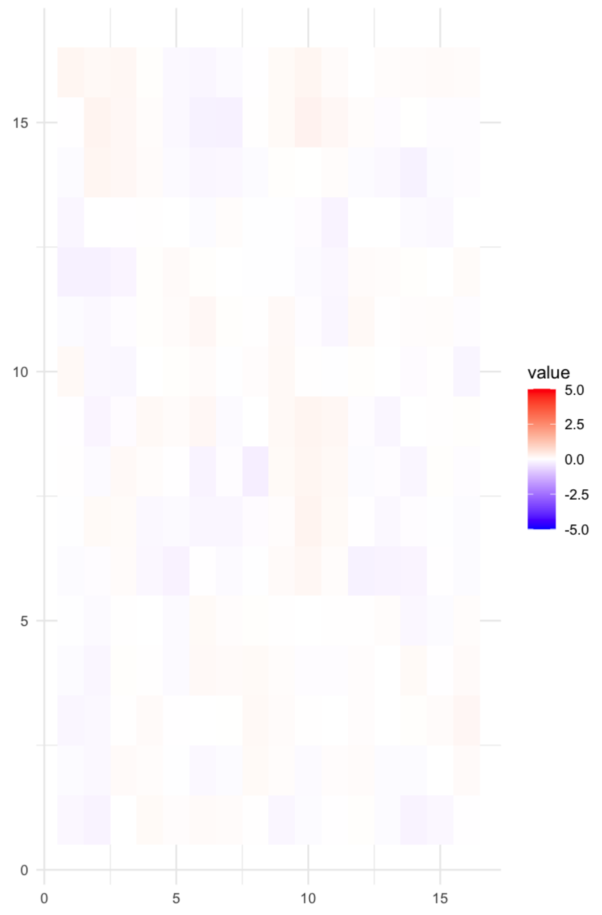


Figure 6: Coefficients for Simulation 2.



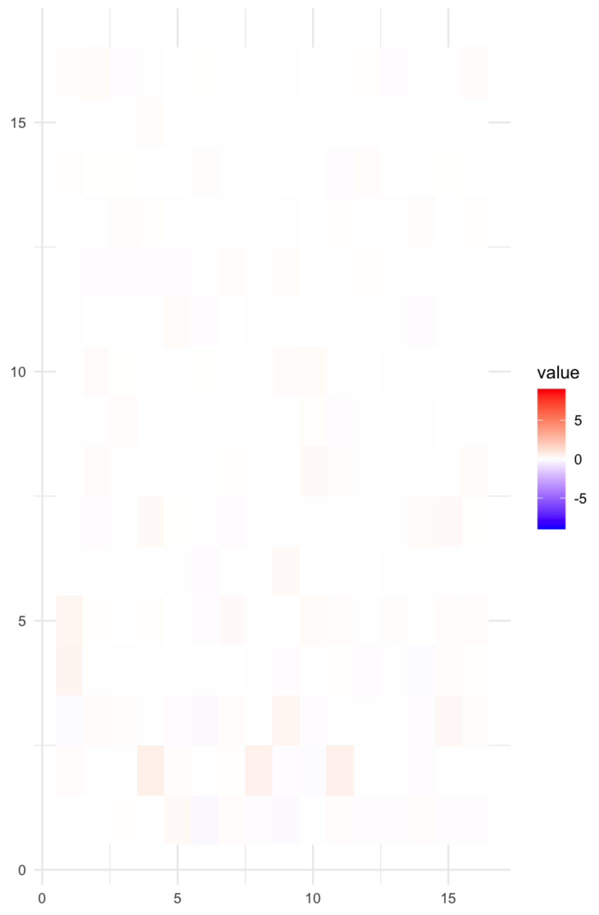


Figure 7: The group mean difference between $y = 1$ and $y = 0$.