Evaluating LASSO Performance in High-Dimensional Data: A Comparison Between Pixel and Frequency Domains

Siyang Ren, Nichole E. Carlson, William Lippitt, Yue Wang

Notes

Everything surrounded by [] are my thoughts/questions/to-dos.

1 Introduction

High-dimensional imaging data present unique challenges for predictive modeling and feature selection due to the strong spatial correlations between neighboring pixels. These dependencies introduce multicollinearity, which can obscure important features and reduce predictive accuracy when standard models like LASSO are applied.

To address this, we leverage the eigendecomposition of the spatially adjusted matrix MCM, where M centers the data and C encodes pixel adjacency. This transformation projects imaging data into a frequency space, reducing spatial dependencies while preserving interpretable spatial structures. While MCM-based transformations do not fully decorrelate the data, they strike a balance between computational efficiency and interpretability, making them well-suited for high-dimensional imaging analyses.

This study evaluates the utility of these transformations by simulating imaging datasets under varying sparsity scenarios. By applying LASSO in both the pixel and frequency spaces, we assess whether the whitening transformation improves predictive performance, focusing on classification accuracy, coefficient recovery, and feature selection.

2 Methods

This section presents the methodology for analyzing high-dimensional imaging data, addressing the challenges posed by spatial correlations. We describe how data is transformed into a frequency space using spatial adjacency matrices, enabling the reduction of spatial dependencies while preserving meaningful patterns. The approach is then applied to simulated datasets, where LASSO is used for feature selection and model evaluation, providing insights into the effectiveness of this transformation in improving predictive performance.

2.1 Spatial Correlations in Imaging Data

Imaging data are often characterized by spatial autocorrelation, where neighboring pixels exhibit similar values due to their proximity. This spatial dependency introduces challenges in predictive modeling, such as multicollinearity, reduced computational efficiency, and obscured spatial signals.

When image data are used as predictors, each pixel serves as a feature to predict a scalar outcome y. Let X represent a dataset of images, where each image x_i is a row vector of p pixels. The predictive model can be expressed as:

$$y_i = x_i\beta + \epsilon_i,$$

where β is a vector of regression coefficients, and $\epsilon_i \sim N(0, \sigma^2)$ is the error term. In this scenario, the inherent correlations among pixels can lead to multicollinearity, which reduces the stability of coefficient estimation and complicates feature selection.

Alternatively, when image data are treated as outcomes, each pixel becomes a response variable, modeled as a function of predictors. Suppose x_i is a vector of predictors and y_i represents an image outcome with s pixels. The model can be written as:

$$y_i = x_i\beta + \epsilon_i,$$

where β is a $p \times s$ matrix of coefficients and $\epsilon_i \sim \mathcal{N}(0, \Sigma)$ captures the covariance among pixels. A key consideration in this case is the structure of the covariance matrix Σ, which determines how spatial relationships between pixels are modeled. Simplifying Σ improves computational efficiency but may sacrifice accuracy, while a more complex structure captures fine details but increases computational demands.

In both scenarios, the spatial correlations between pixels necessitate strategies to reduce dependencies and improve model performance. Two common approaches are dimension reduction, such as principal component analysis (PCA), and modifying the covariance structure. While PCA effectively reduces dimensionality, it often sacrifices interpretability, as the components are linear combinations of the original features.

2.2 Whitening Transformations to the Frequency Space

Reducing spatial dependencies in imaging data is crucial for effective modeling and feature selection. This study leverages the eigendecomposition of MCM, where M centers the data and C encodes spatial relationships, to transform data into a frequency space. This transformation aligns features with orthogonal spatial patterns, reducing dependencies while preserving interpretable spatial structures. The eigenvalues of MCM, connected to Moran's Coefficient, quantify the strength of spatial autocorrelation along each eigenvector, ensuring the transformed features capture meaningful spatial variations.

2.2.1 Moran's Coefficient

Moran's Coefficient (MC) is a widely used statistic for quantifying spatial autocorrelation, which measures the degree to which similar values cluster in space. For an image represented as a vector $x = (x_1, x_2, \ldots, x_p)^T$ of length p, where x_i denotes the intensity of the i-th pixel,

and a spatial adjacency matrix C of size $p \times p$, Moran's Coefficient is defined in summation form as follows Griffith and Chun (2014):

$$MC(x) = \frac{p}{\sum_{i=1}^{p}\sum_{j=1}^{p} C_{ij}} \cdot \frac{\sum_{i=1}^{p}(x_i - \bar{x})\left[\sum_{j=1}^{p} C_{ij}(x_j - \bar{x})\right]}{\sum_{i=1}^{p}(x_i - \bar{x})^2}$$

where C_{ij} encodes the spatial relationship between pixels i and j.

To simplify this expression, we transform it into matrix form by introducing the centering matrix $M = I - \frac{1}{p}11^T$. The centering matarix M ssatisfies two essential properties: $M^2 = M$ (idempotencen) and $M^T = M$ (symmetry). The idempotence of M can be shown as follows. By definition, we have:

$$M^2 = MM = \left(I - \frac{11^T}{p}\right)\left(I - \frac{11^T}{p}\right).$$

Expanding this expression yields:

$$M^2 = I - \frac{11^T}{p} - \frac{11^T}{p} + \frac{11^T 11^T}{p^2}.$$

Since $1^T 1 = p$, the last term simplifies to $\frac{11^T}{p}$, and we get:

$$M^2 = I - \frac{2}{p}11^T + \frac{1}{p}11^T = I - \frac{11^T}{p} = M.$$

Symmetry is evident because both I and $\frac{1}{p}11^T$ are symmetric matrices, so M is also symmetric.

The general property of matrix-vector multiplication also aids in rewriting the summation form of Moran's Coefficient. For any two column vectors a and c of length p, and a matrix B of size $p \times p$, the expression $a^T Bc$ can be written in summation form as:

$$a^T Bc = \sum_{i=1}^{p}\sum_{j=1}^{p} a_i B_{ij} c_j.$$

This property can be verified by examining the operations step-by-step. First, the matrix-vector product Bc produces a column vector, where the i-th entry is:

$$(Bc)_i = \sum_{j=1}^{p} B_{ij} c_j.$$

Multiplying this result by the vector a^T computes the weighted sum of these entries:

$$a^T Bc = \sum_{i=1}^{p} a_i (Bc)_i = \sum_{i=1}^{p} a_i \left(\sum_{j=1}^{p} B_{ij} c_j\right).$$

Rearranging the terms gives:

$$a^T Bc = \sum_{i=1}^{p}\sum_{j=1}^{p} a_i B_{ij} c_j,$$

proving the equivalence.

Using this approach, the numerator of Moran's Coefficient, originally written as

$$\sum_{i=1}^{p}\sum_{j=1}^{p} C_{ij}(x_i - \bar{x})(x_j - \bar{x}),$$

becomes $(Mx)^T C(Mx)$, representing the covariance of x along the spatial structure defined by C. Similarly, the denominator, $\sum_{i=1}^{p}(x_i - \bar{x})^2$, simplifies to $(Mx)^T(Mx)$, which captures the covariance of x along an independent structure. The normalization factor, $\sum_{i=1}^{p}\sum_{j=1}^{p} C_{ij}$, is rewritten as $1^T C1$, the total weight of the spatial adjacency matrix. Combining these terms, Moran's Coefficient in matrix form is expressed as:

$$MC(x) = \frac{p}{1^T C1} \cdot \frac{(Mx)^T C(Mx)}{(Mx)^T(Mx)}.$$

Finally, substituting the properties of M into the matrix form, Moran's Coefficient can be further compacted to:

$$MC(x) = \frac{p}{1^T C1} \cdot \frac{x^T MCMx}{x^T Mx}.$$

2.2.2 Eigenvector Spatial Filtering (ESF)

Eigenvector Spatial Filtering is a method for identifying orthogonal spatial patterns by using the eigendecomposition of MCM. The eigendecomposition of MCM is expressed as:

$$MCM = E\Lambda E^T,$$

where Λ is a diagonal matrix containing eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$, and E is a matrix whose columns are the corresponding eigenvectors. Each eigenvector v_i represents a distinct spatial pattern, and its associated eigenvalue λ_i quantifies the strength of spatial autocorrelation for that pattern.

The eigendecomposition of MCM exhibits several important properties. First, since MCM is symmetric, the eigenvectors are orthogonal ($E^T E = I$). Second, eigenvectors corresponding to nonzero eigenvalues are orthogonal to the vector of ones because $MCM1 = 0$, ensuring that the patterns are mean-centered.

Connecting Moran's Coefficient and Eigenvector Spatial Filtering de Jong et al. (1984) demonstrated that when C is symmetric, the eigenvalues of MCM correspond to the Moran coefficients of their respective eigenvectors. This result implies that, for any spatial structure defined by C, the eigenvector associated with the largest eigenvalue of MCM achieves the highest possible Moran coefficient among all vectors. This eigenvector represents the dominant direction of spatial autocorrelation, capturing the strongest spatial pattern inherent to the structure of C.

To establish the connection, consider the definition of the Moran coefficient for an eigenvector v_i:

$$MC(v_i) = \frac{p}{1^T C1} \cdot \frac{v_i^T MCMv_i}{v_i^T Mv_i}.$$

We first prove that for any eigenvector v_i, $Mv_i = v_i$. Given that $MCMv_i = \lambda_i v_i$, where λ_i is the eigenvalue corresponding to v_i, we apply M on both sides:

$$MCMv_i = \lambda_i v_i \implies M(MCMv_i) = M(\lambda_i v_i).$$

Since $M^2 = M$ (idempotence), this becomes:

$$MCMv_i = \lambda_i Mv_i.$$

By comparing with the original equation $MCMv_i = \lambda_i v_i$, we deduce:

$$\lambda_i v_i = \lambda_i Mv_i.$$

For $\lambda_i \neq 0$, this implies $Mv_i = v_i$, as required.

With this property established, we simplify the numerator of $MC(v_i)$. Using $MCMv_i = \lambda_i v_i$, we find:

$$v_i^T MCMv_i = v_i^T(\lambda_i v_i) = \lambda_i(v_i^T v_i).$$

The denominator, $v_i^T Mv_i$, simplifies as:

$$v_i^T Mv_i = v_i^T v_i,$$

since $Mv_i = v_i$.

Substituting these results into the definition of $MC(v_i)$, we have:

$$MC(v_i) = \frac{p}{1^T C 1} \cdot \frac{\lambda_i v_i^T v_i}{v_i^T v_i}.$$

The terms $v_i^T v_i$ cancel, leaving:

$$MC(v_i) = \frac{p}{1^T C 1}\lambda_i.$$

This derivation shows that the Moran coefficient of an eigenvector v_i is directly proportional to its eigenvalue λ_i. The eigenvector associated with the largest eigenvalue achieves the highest Moran coefficient, capturing the strongest spatial autocorrelation, while eigenvectors with smaller or negative eigenvalues represent weaker or dispersed spatial patterns.

The introduction of Moran's Coefficient (MC) and Eigenvector Spatial Filtering (ESF) provides a foundation for understanding and reducing spatial dependencies in imaging data. MC quantifies spatial autocorrelation, while ESF leverages the eigendecomposition of MCM to decompose spatial patterns into orthogonal components. By establishing the proportional relationship between MC and the eigenvalues of MCM, we highlight how eigenvectors capture distinct spatial autocorrelation patterns inherent to the structure of C. This connection allows us to project data into a frequency space, where each component represents a specific spatial pattern, enabling the reduction of spatial dependencies while preserving interpretable relationships.

2.2.3 Whitening Transformations

We now explain how the properties of the Moran coefficient and the eigenvectors of MCM can be utilized to reduce data complexity when fitting image data into models. Specifically, we demonstrate how these properties can guide whitening transformations, which are often employed to reduce covariance complexity in high-dimensional data.

A whitening transformation reduces correlations between variables by transforming a vector of random variables (e.g., an image) such that its covariance matrix becomes diagonal. Among various whitening methods, orthogonal transformations are particularly popular because they preserve the length of the transformed vector.

Orthogonal matrices are typically derived from the eigendecomposition of a symmetric matrix. In our context, we perform eigendecomposition on the centered adjacency matrix MCM, as described earlier. This approach is particularly useful for data with spatial autocorrelation, as it provides interpretable transformations. The eigenvector associated with the i-th largest eigenvalue represents the i-th strongest direction of spatial autocorrelation, as defined by the adjacency matrix C. Thus, this transformation aligns the data along spatial patterns encoded by the eigenvectors, effectively reducing dependencies in the data.

It is important to note that this transformation is not directly based on the empirical covariance of the dataset but instead on the spatial structure defined by C. Consequently, while this transformation may not fully eliminate pixel correlations, a reasonable adjacency matrix C can substantially reduce these dependencies, enhancing computational efficiency and model performance.

The choice of adjacency matrix C is crucial for effectively reducing spatial dependencies. For an image with p pixels, C is a $p \times p$ matrix, where each element C_{ij} encodes the adjacency relationship between pixels i and j. There are various strategies for defining adjacency. One approach is data-independent and unweighted, where adjacency is binary, with values of 1 assigned to adjacent pixels and 0 otherwise. For instance, a "2-neighbor adjacency matrix" considers two pixels adjacent if they are direct neighbors or share a direct neighbor. All other entries, including the diagonal, are set to 0.

Another approach is data-independent but weighted, where adjacency values are based on the distance between pixels. Let d_{ij} denote the Euclidean distance between pixels i and j. The adjacency value is then defined as $C_{ij} = -\exp(d_{ij})$. Both methods rely on pixel positions rather than their actual intensity values, providing a flexible framework for encoding spatial relationships in C.

Once an adjacency matrix C is chosen, the eigendecomposition of MCM yields the orthogonal matrix E, whose columns are the eigenvectors of MCM. Each eigenvector v_i, corresponding to the i-th largest eigenvalue, captures the i-th strongest spatial autocorrelation pattern inherent in the spatial structure of C.

For a dataset X containing n images, represented as an $n \times p$ matrix (where each row corresponds to a flattened image), projecting the data into the frequency space is achieved by:

$$X_{\text{freq}} = X \cdot E.$$

This transformation aligns the data along orthogonal spatial patterns encoded by the eigenvectors, facilitating more efficient modeling and analysis while preserving meaningful spatial relationships.

2.3 Models

To evaluate the performance of the proposed whitening transformation, we focus on binary classification problems where the outcome $y_i \in \{0, 1\}$ is predicted using imaging data as predictors. To model binary outcomes, we use logistic regression, which predicts the probability of an event occurring (e.g., $y_i = 1$) as a function of predictors X. Let x_i represent the vector of predictor variables for the i-th observation, $y_i \in \{0, 1\}$ denote the binary outcome, and β represent the vector of coefficients. The probability of $y_i = 1$ is modeled as:

$$p(y_i = 1) = \frac{1}{1 + \exp(-x_i^T \beta)}.$$

The model estimates β by maximizing the log-likelihood of the observed data, which is expressed as:

$$\ell = \ln(L) = \sum_i \left[y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i) \right].$$

In LASSO logistic regression and L_1 penalty is added to the negative log-likehood to enforce sparsity in the coefficients:

$$\min_\beta -\ell + \lambda \sum_j |\beta_j|,$$

where $\lambda > 0$ controls the strength of regularization, shrinking coefficients β_j with small absolute values to zero. This regularization enforces sparsity, making the model particularly suited for high-dimensional data with sparse true features.

In the context of imaging data, each pixel is treated as a predictor. Spatial autocorrelation between pixels can degrade the performance of the LASSO model by violating the independence assumption among predictors. To address this, whitening transformations can be employed to reduce spatial dependencies by projecting the data into a frequency space where features are decorrelated. As introduced earlier, this transformation is achieved through the eigendecomposition of the spatially adjusted matrix MCM. Let E represent the matrix of eigenvectors of MCM, the transformed data in the frequency space is then:

$$X_{\text{freq}} = X \cdot E.$$

When applying logistic regression, the relationship between predictors and the outcome is defined by the coefficient vector β, which assigns weights to predictors. In the pixel space, β indicates the importance of each pixel in determining the classification. When data is transformed into the frequency space, the coefficients must also be transformed to align with the new basis defined by E. The relationship between the data and coefficients can be expressed as:

$$X \cdot \beta = X_{\text{freq}} \cdot b = (X \cdot E) \cdot b.$$

For this equality to hold for all X, it follows that:

$$\beta = E \cdot b.$$

Taking the transpose of both sides and leveraging the orthogonality of E ($E^T E = I$), the coefficient vector in the frequency space is:

$$b = E^T \beta.$$

The transformed coefficient vector b encodes the contribution of each frequency component to the classification, analogous to how β encodes the contribution of each pixel in the original space.

To evaluate the effectiveness of whitening transformations, we compare the performance of LASSO models in the pixel space and the frequency space. In the pixel space, the data matrix X, representing flattened images, is used directly, with the coefficient vector β indicating the importance of each pixel for classification. In the frequency space, the data is transformed using the eigenvectors E of the spatially adjusted matrix MCM, resulting in $X_{\text{freq}} = X \cdot E$. Simultaneously, the coefficient vector β is transformed into the frequency space as $b = E^T \beta$. The transformed coefficients b indicate the contribution of each frequency component to the classification outcome. LASSO models are fit in both spaces, producing estimated coefficients $\hat{\beta}$ and $\hat{b}$, respectively, which are compared with the true coefficients (β and b) to evaluate feature selection performance.

To ensure a comprehensive evaluation, we consider two distinct scenarios that reflect different ways imaging data might exhibit important features for binary classification tasks: sparsity in the pixel space and sparsity in the frequency space. Sparsity is a natural assumption in many imaging applications, where only a small subset of pixels or frequency components contributes meaningfully to the outcome, making the data well-suited for LASSO regression.

In the first scenario, sparsity exists in the pixel space. Here, the true coefficient vector β is sparse in the pixel space, meaning that only a subset of pixels contributes to the classification outcome. This scenario mimics applications where localized image regions determine the outcome. The data matrix X is transformed into the frequency space using $X_{\text{freq}} = X \cdot E$, and the coefficients are transformed as $b = E^T \beta$. LASSO is fit in both the pixel space and the frequency space, and the estimated coefficients ($\hat{\beta}$ and $\hat{b}$) are compared to the true coefficients to assess feature selection and classification performance.

In the second scenario, sparsity exists in the frequency space. This assumes that the image pattern is dominated by a few frequency components, representing scenarios where broader spatial patterns, rather than individual pixels, determine the outcome. The data is simulated directly in the frequency space as X_{freq}, with sparse coefficients b indicating the important frequencies. The outcomes are generated using $X_{\text{freq}} \cdot b$. The data and coefficients are then transformed back into the pixel space as $X = X_{\text{freq}} \cdot E^T$ and $\beta = E \cdot b$. Similar to the first scenario, LASSO models are fit in both spaces, and the estimated coefficients are compared to evaluate the effectiveness of whitening transformations.

The performance of the LASSO models is evaluated based on two criteria. The first is the ability to recover the true nonzero coefficients, indicating the model's capacity for accurate feature selection. The second is the classification accuracy on the outcomes y, reflecting the overall predictive performance. By comparing these results across the pixel and frequency spaces, we aim to determine whether whitening transformations mitigate the effects of spatial autocorrelation and enhance the robustness of LASSO regression in scenarios with different sparsity structures.

2.4 Simulations

To ensure robust and reliable results, each simulation is repeated 500 times. This repetition mitigates the influence of random variability inherent in data generation and model fitting, allowing us to obtain stable and generalizable performance metrics by averaging outcomes across repetitions.

In the first scenario, sparsity exists in the pixel space. Here, data X is simulated as a 1000×256 matrix, representing 1000 images, each with 16×16 pixels flattened into 1D vectors of length 256. The rows of X are sampled from a multivariate normal distribution with mean zero and a covariance matrix C_{cov}, where $C_{\text{cov}} = \exp(-\text{dist}(x_i, x_j))$, and $\text{dist}(x_i, x_j)$ is the Euclidean distance between pixels i and j. The coefficient vector β is constructed to ensure sparsity, with coefficients set to 0 for pixels outside the central 8×8 region and assigned a constant nonzero value for pixels within this region. The nonzero value of β is carefully chosen to ensure that the probabilities $p = 1/(1 + \exp(-X\beta))$ are evenly distributed across the range $[0, 1]$. A balanced distribution of p ensures that the classification problem is neither too easy (e.g., probabilities close to 0 or 1) nor too difficult (e.g., probabilities clustered near 0.5), providing a meaningful test of the model's performance.

In the second scenario, sparsity exists in the frequency space. Data X_{freq} is simulated as a 1000×256 matrix, where rows are sampled from a multivariate normal distribution with mean zero and a diagonal covariance matrix. The diagonal values decrease linearly, reflecting stronger spatial autocorrelation in lower-frequency components. The coefficient vector b is created with 10% of its entries randomly assigned nonzero values, while the rest are set to 0, ensuring sparsity in the frequency domain. Similar to the first scenario, the nonzero values of b are chosen to produce probabilities $p = 1/(1 + \exp(-X_{\text{freq}} \cdot b))$ that are evenly distributed. The data and coefficients are then transformed back to the pixel space using $X = X_{\text{freq}} \cdot E^T$ and $\beta = E \cdot b$, enabling model fitting and comparison in both spaces.

In both scenarios, we fit LASSO models in the pixel and frequency spaces to evaluate their performance in terms of coefficient recovery, classification accuracy, and the percentage of coefficients with p-values below 0.05. This analysis allows us to assess not only the models' ability to identify relevant features and accurately classify data but also the statistical significance of the identified coefficients. Detailed results and further analysis of these performance metrics are presented in the following section.

2.5 Analyses

In this section, we explore how well the simulated data aligns with the intended scenarios and evaluate the performance of LASSO models in both the pixel and frequency spaces. We start by visualizing the group mean differences to verify the quality of the simulated data and its adherence to the designed sparsity patterns. We then assess model performance using cross-validation and multiple metrics, such as classification accuracy, AUC, and coefficient recovery, providing a comprehensive evaluation of the effectiveness of whitening transformations.

We will introduce what analyses we performed on the simulated data, which include visualization to check the simulatation quality, and then the model metrics of the LASSO models.

2.5.1 Group Mean Difference

In both simulations, group mean differences were computed for the pixel space and the frequency space to evaluate whether the simulated coefficients (β or b) align with the intended classification patterns. The results were visualized using heatmaps for the pixel space and scatterplots for the frequency space, highlighting the differences in patterns across these domains. For visualization purposes, only the data from the first iteration of each simulation was used.

In the pixel space, the group mean difference for each pixel was calculated by subtracting the average value of that pixel for images with $y = 0$ from the average value for images with $y = 1$. These differences were displayed as a heatmap, providing a visual representation of how well the simulated coefficients influence the classification outcomes. For simulations where β is nonzero only in the central 8×8 region, we expect the heatmap to show positive group mean differences in this region, while the differences outside this area should remain close to zero.

In the frequency space, group mean differences were computed for each frequency by calculating the difference in the average values of that frequency between $y = 1$ and $y = 0$. These differences were visualized as a scatterplot, with frequencies on the x-axis and the corresponding group mean differences on the y-axis. When the spatial adjacency matrix C_{adj} accurately reflects the covariance structure of the data, frequencies corresponding to higher eigenvalues of MCM are expected to show larger group mean differences because these eigenvalues represent directions of greater spatial variance, leading to larger differences between the groups.

For the second simulation, where data was generated directly in the frequency space, we expect the group mean differences to align with the diagonal covariance structure of the simulated data. Frequencies with larger diagonal covariance values should display more pronounced group mean differences, while frequencies with smaller diagonal values should show weaker differences. In the pixel space, however, the heatmap is expected to lack clear patterns, as the pixel-level differences are primarily driven by the transposed coefficients (β) rather than localized spatial effects.

By visualizing the group mean differences in both the pixel and frequency spaces, this analysis provides insights into the alignment between the simulated coefficients and the classification outcomes, as well as the effects of spatial transformations.

2.5.2 Cross-Validation

To compare the performance of LASSO models in the pixel and frequency spaces, separate models were fit using data from each space. Logistic regression with L_1 penalty was applied to select relevant features, with the optimal regularization parameter λ determined via 10-fold cross-validation. The dataset was divided into 80% training data and 20% test data. Cross-validation was performed on the training set to identify the best λ, which balances the trade-off between achieving higher classification accuracy and enforcing sparsity by penalizing nonzero coefficients. Two values of λ were considered: `lambda.min`, which minimizes the cross-validated error, and `lambda.1se`, which represents the largest λ within one standard error of the minimum.

In 10-fold cross-validation, the training set is randomly divided into 10 subsets of approximately equal size. The model is trained on nine subsets and evaluated on the remaining subset, with this process repeated 10 times so that each subset serves as the test fold once. This process is performed for a range of λ values, and the value of λ that yields the best average performance is selected. The final model is then trained on the entire training set using the selected λ and evaluated on the held-out test set.

By applying this process to models in both the pixel and frequency spaces, we evaluate whether the whitening transformation enhances model performance. The comparison focuses on the models' ability to enforce sparsity, select relevant features, and accurately predict outcomes.

2.5.3 Metrics and Visualization

To evaluate the performance of LASSO models, several metrics were used. After predicting the probabilities p_i for $y = 1$, classifications were made by assigning $y_i = 1$ if $p_i > 0.5$, and $y_i = 0$ otherwise. The classification accuracy was then calculated as the proportion of correctly predicted outcomes. Another key metric was the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve, which provides a measure of the model's ability to balance true positive and false positive rates across various thresholds. Higher AUC values indicate better classifier performance, particularly in distinguishing between classes. Additionally, p-values for the predictors were computed using the `hdi` package. While logistic regression with L_1 penalty does not inherently produce p-values, this package facilitates their calculation, and further details on its methodology will be provided.

The analysis also included a detailed comparison of the coefficients estimated by LASSO models in the pixel and frequency spaces. For each simulation, we visualized the true coefficients, the mean estimated coefficients across 500 iterations, and their transformations between the two spaces. In the pixel space, coefficients were visualized as 16×16 heatmaps, reflecting their spatial arrangement. In the frequency space, coefficients were displayed as scatterplots, ordered by the corresponding eigenvalues of MCM from smallest to largest.

For Simulation 1, where data was generated in the pixel space, the visualizations included: the true coefficients β and their transformation into the frequency space as b, the coefficients estimated from models fit in the pixel space and their transformed estimates in the frequency space, and the coefficients estimated in the frequency space with their transformed counterparts in the pixel space. Similarly, for Simulation 2, where data was generated in the frequency space, we visualized b, the transformed coefficients β, and the corresponding estimates and transformations for both spaces.

Since two regularization parameters (`lambda.min` and `lambda.1se`) were used, each scenario produced 10 visualizations in total. These visualizations provided insights into the effects of whitening transformations on coefficient estimation and feature selection accuracy in both the pixel and frequency spaces.

Additionally, the p-values from the `hdi` package were analyzed. For each simulation, the percentage of predictors with $p < 0.05$ was calculated and visualized. In the pixel space, these percentages were displayed as heatmaps, while in the frequency space, they were shown as scatterplots ordered by eigenvalues. This analysis offered a deeper understanding of the statistical significance of predictors under different modeling approaches and sparsity

scenarios.

The p-values from the `hdi` package were analyzed to assess the statistical significance of predictors in both the pixel and frequency spaces. For each simulation, the percentage of iterations (out of 500) in which each predictor had a p-value less than 0.05 was calculated. This provided a measure of the consistency with which predictors were identified as significant across simulations.

In the pixel space, these percentages were visualized as heatmaps, reflecting the spatial distribution of statistically significant predictors. In the frequency space, the percentages were displayed as scatterplots, with frequencies ordered by the corresponding eigenvalues of MCM. By comparing these visualizations, we could assess how the whitening transformation influenced the identification of significant predictors under different modeling approaches and sparsity scenarios.

3 Results

3.1 Effect Size Determination

In Simulation 1, we evaluated the distribution of the success probability $\mathbf{p}$ at different non-zero values of β (0.01, 0.05, 0.1, 0.2, and 1). As shown in Figure 1, a value of 0.1 produced the most uniform distribution of $\mathbf{p}$, making it the optimal choice for model fitting in this scenario.

Similarly, in Simulation 2, we assessed the distribution of $\mathbf{p}$ at various non-zero values for $\mathbf{b}$ (0.1, 0.15, 0.2, 0.25, and 0.3). As shown in Figure 2, the value of 0.2 resulted in the most uniform distribution of $\mathbf{p}$, making it the best option for this simulation.

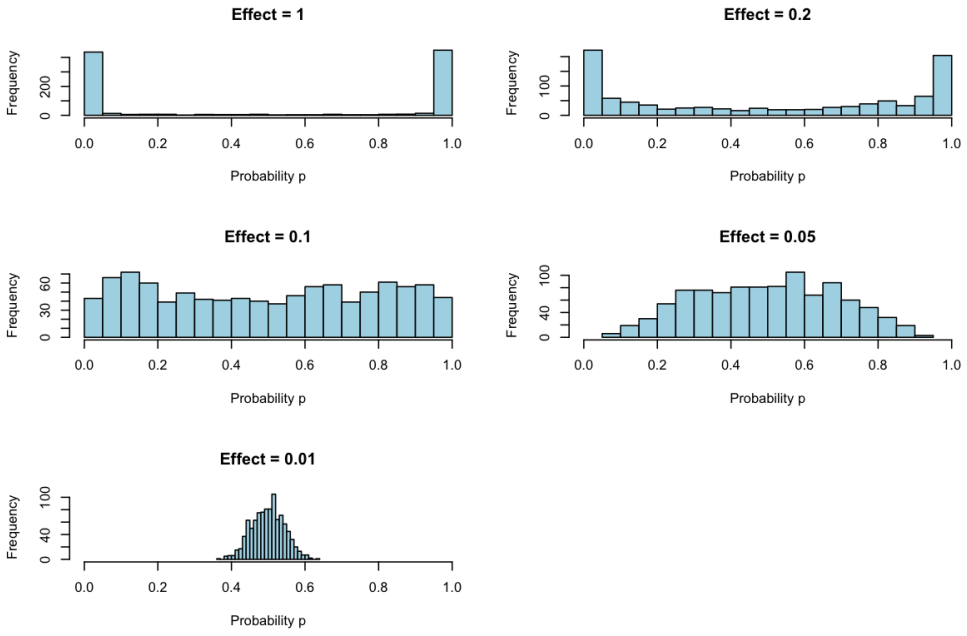

Figure 1: Distribution of success probability $\mathbf{p}$ at different non-zero values in Simulation 1.

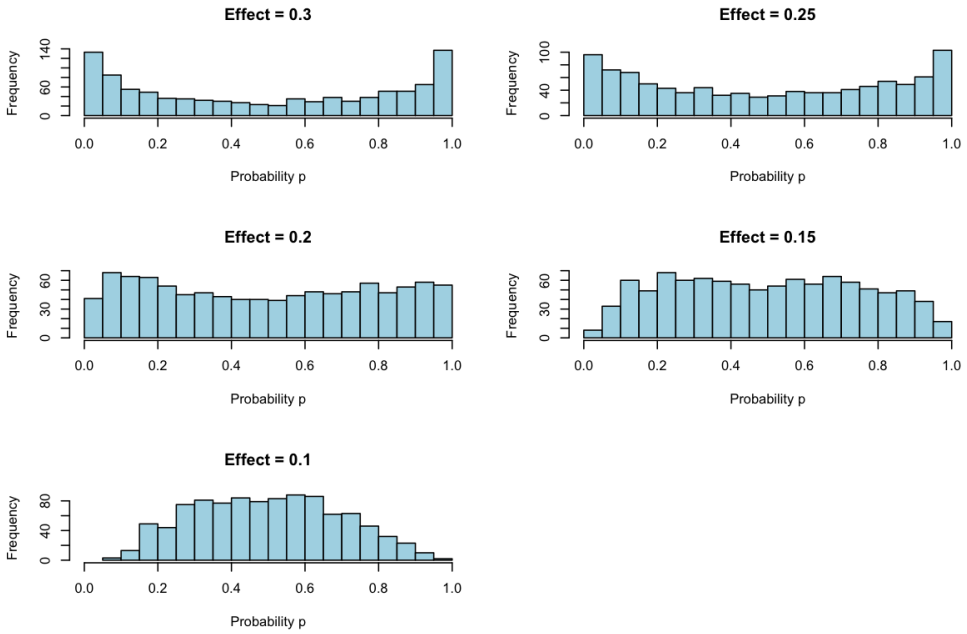

Figure 2: Distribution of success probability $\mathbf{p}$ at different non-zero values in Simulation 2.

Group Mean Difference

In this subsection, we examine the group mean differences in covariate values between instances where $y = 1$ and $y = 0$ for both Simulation 1 and Simulation 2.

Figure 3 presents the group mean differences for Simulation 1, with the heatmap on the left showing that regions corresponding to non-zero coefficients in β exhibit larger mean differences between $y = 1$ and $y = 0$, as larger covariate values in these locations have higher probabilities of being assigned to $y = 1$. The scatterplot on the right displays the group mean differences in the frequency domain, where each point represents a frequency component; frequencies associated with larger eigenvalues tend to have larger mean differences. Figure 4 shows the actual coefficients used in Simulation 1, where non-zero coefficients in β are localized to specific pixels, corresponding to the areas with larger mean differences in the group comparison.

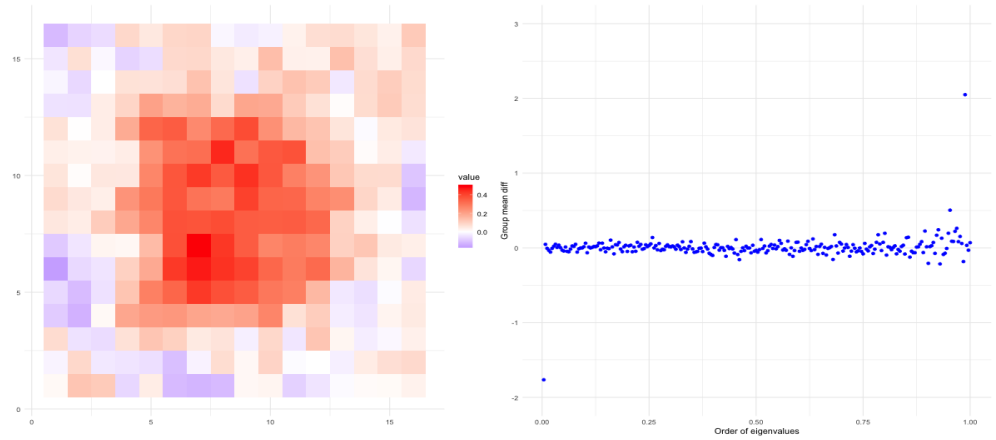

Figure 3: Group mean difference in covariate values between instances where $y = 1$ and $y = 0$ in Simulation 1, shown for both the pixel space (left) and frequency space (right).

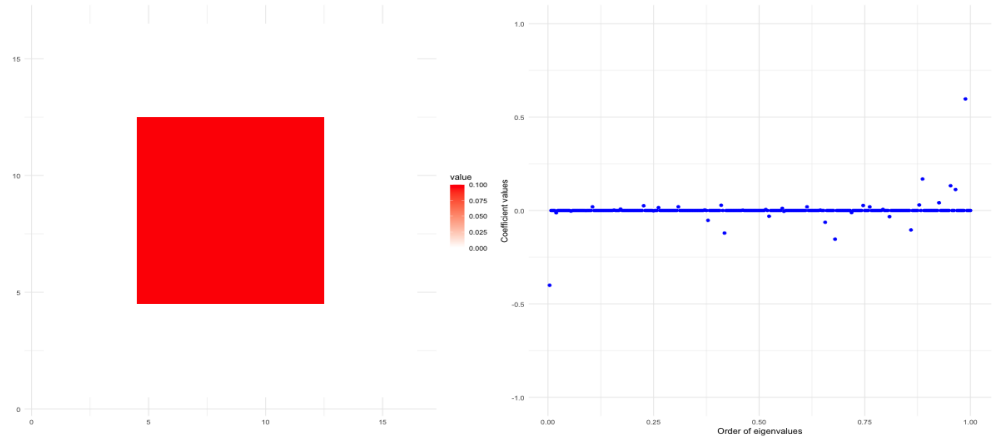

Figure 4: Actual coefficients in Simulation 1 for the pixel space (left) and frequency space (right).

Figure 5 shows the group mean differences for Simulation 2, while Figure 6 displays the actual coefficients. The non-zero coefficients in $\mathbf{b}$ are uniformly set to 0.2. However, the scatterplot in the frequency space does not clearly highlight the non-zero components, with increasing variance observed for larger eigenvalues. This variance pattern is consistent with the diagonal covariance matrix used in the simulation. The difficulty in identifying the non-zero components suggests that the effect size may be too small relative to the variance, making detection challenging. [Further adjustments to either the effect size or the covariance matrix could improve the detectability of these non-zero coefficients in future analyses.]

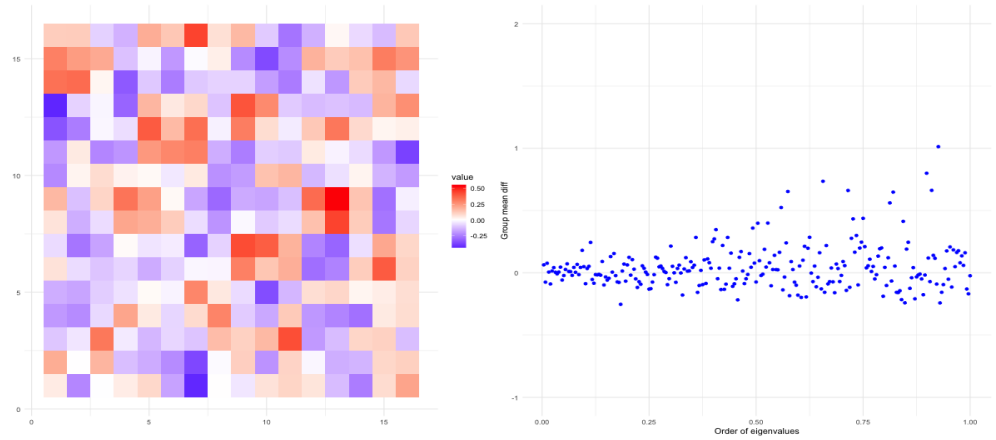

Figure 5: Group mean difference in covariate values between instances where $y = 1$ and $y = 0$ in Simulation 2.

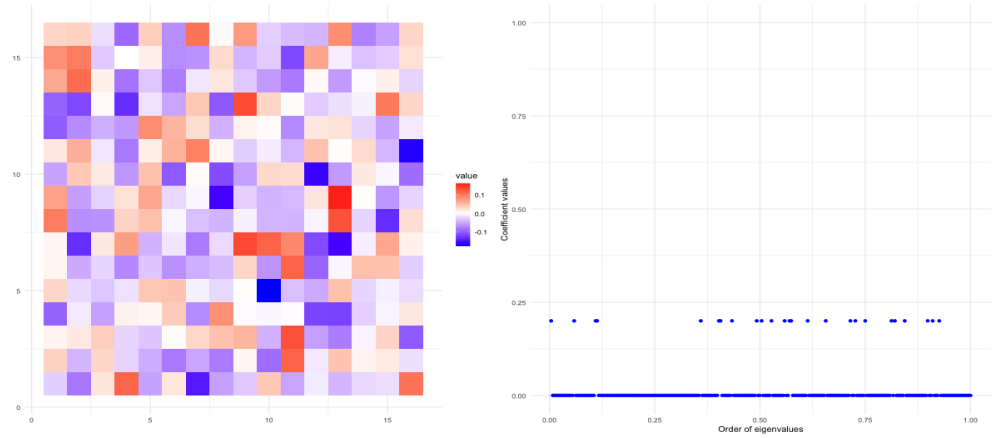

Figure 6: Actual coefficients in Simulation 2 for the pixel space (left) and frequency space (right).

AUC and Accuracy

Table 1 summarizes the average AUCs and accuracies over 500 iterations. In both Simulation 1 (pixel space sparsity) and Simulation 2 (frequency space sparsity), models fitted in the frequency space consistently outperformed those fitted in the pixel space. For example, in Simulation 1, using `lambda.min` as the regularization parameter, models trained with pixel space covariates achieved an AUC of 0.803 (SE = 0.031) and an accuracy of 72.6% (SE = 0.032). In contrast, models trained with frequency space covariates produced a slightly higher AUC of 0.826 (SE = 0.028) and a higher accuracy of 74.5% (SE = 0.030). A similar trend was observed in Simulation 2, with frequency space models showing superior performance regardless of the regularization parameter used.

Table 1: Comparison of AUC and accuracy between models fitted in the pixel space and frequency space across 500 iterations for Simulation 1 and Simulation 2.

Simulation	**Model in Pixel Space**		**Model in Frequency Space**	
	AUC (SE)	**Accuracy (SE)**	**AUC (SE)**	**Accuracy (SE)**
Simulation 1				
`lambda.min`	0.803 (0.031)	0.726 (0.032)	0.826 (0.028)	0.745 (0.030)
`lambda.1se`	0.800 (0.032)	0.722 (0.032)	0.826 (0.029)	0.745 (0.031)
Simulation 2				
`lambda.min`	0.755 (0.036)	0.684 (0.034)	0.812 (0.030)	0.732 (0.032)
`lambda.1se`	0.735 (0.039)	0.669 (0.038)	0.812 (0.031)	0.732 (0.032)

Coefficients Estimation

Figure 8 presents the mean estimated b values plotted against the order of eigenvalues. The order of eigenvalues are calculated the same way as above. For Simulation 1, `lambda.1se` shrinks the estimated coefficients more than `lambda.min`, as it provides a larger panelty on it. For Simulation 2, even though it is not obvious, I feel the estimated values has an increase trend as the eigenvalues increase. [Still, I am wondering whether this is related to the covariance matrix, the decreased diagonal values, consider math proof?].

The mean estimated coefficients across iterations were calculated, and Figure 7 displays the mean estimated β values. Two key observations can be made: (1) There is no significant difference in the estimated coefficients when using `lambda.min` versus `lambda.1se`, and (2) the estimated values align well with the actual values, indicating that the model is accurately identifying the relevant features.

Figure 8 shows the mean estimated $\mathbf{b}$ values plotted against the order of eigenvalues. The eigenvalue ordering is consistent with earlier calculations. In Simulation 1, `lambda.1se` applies a stronger regularization penalty, shrinking the estimated coefficients more than `lambda.min`. For Simulation 2, although the trend is less clear, there seems to be an upward trend in the estimated values as the eigenvalues increase. This trend may be related to the structure of the covariance matrix, specifically its decreasing diagonal values [consider math proof?].

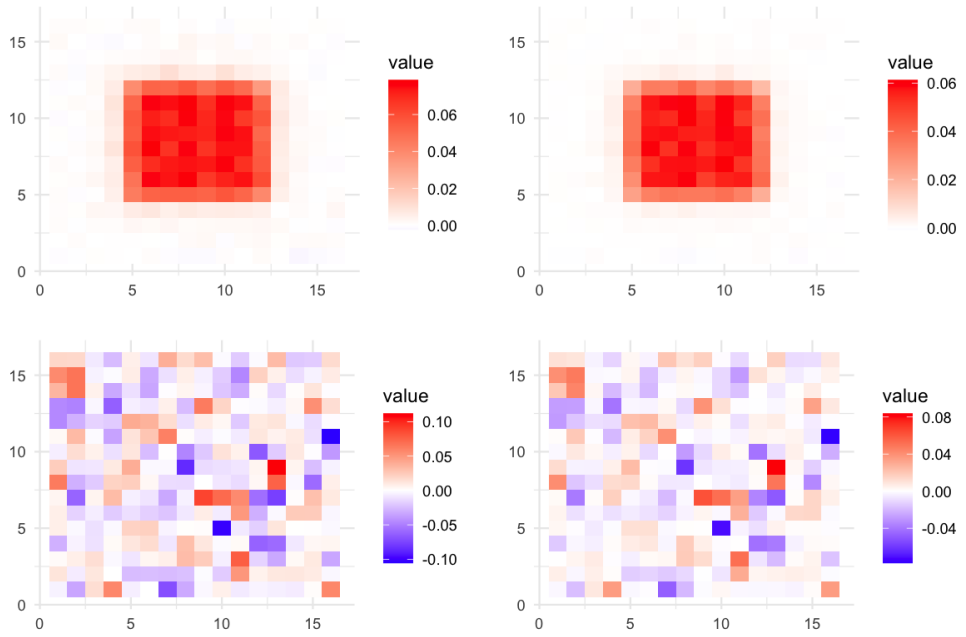

Figure 7: Mean estimated β values across simulations, with models fitted using `lambda.min` (left) and `lambda.1se` (right). The top row shows results for Simulation 1, while the bottom row shows results for Simulation 2.

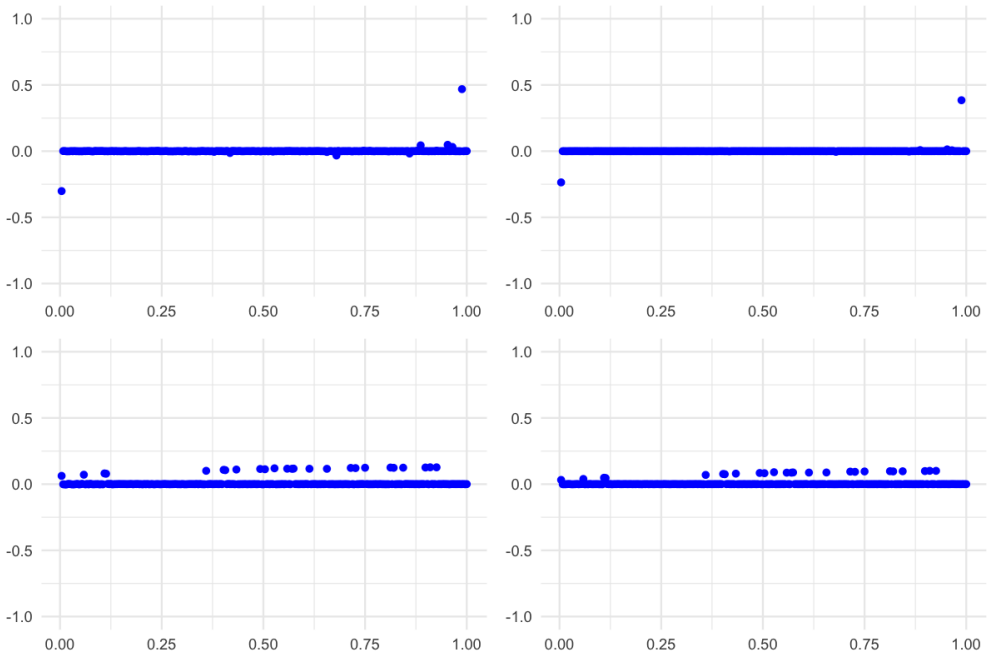

Figure 8: Mean estimated b values across simulations, plotted against ordered eigenvalues. Models fitted using `lambda.min` are on the left and models fitted with `lambda.1se` on the right. The top row shows results for Simulation 1, while the bottom row shows results for Simulation 2.

Significant P-values

It is interesting to observe that, although the heatmap for significance of β in Simulation 1 follows the pattern of the actual non-zero values, the percentage of significance is relatively low (Figure 9 left). In contrast, the non-zero values of $\mathbf{b}$ (Figure 10 left) show a much higher percentage of significance, reaching as high as 100% across iterations.

Another observation is that, although the non-zero effect size for $\mathbf{b}$ is constant in Simulation 2, the percentage of significant p-values increases as the eigenvalues grow (Figure 10 right). [I am considering creating a plot to visualize the actual β values in Simulation 2 and the actual b values in Simulation 1, where the non-zero values vary, and compare them with the corresponding percentage of significant p-values. The goal is to examine whether the size of the actual non-zero values correlates with the percentage of significance. I suspect that a larger absolute effect size should result in a higher percentage of significance, but this doesn't seem to be the case for b in Simulation 2, so I want to investigate other factors as well.]

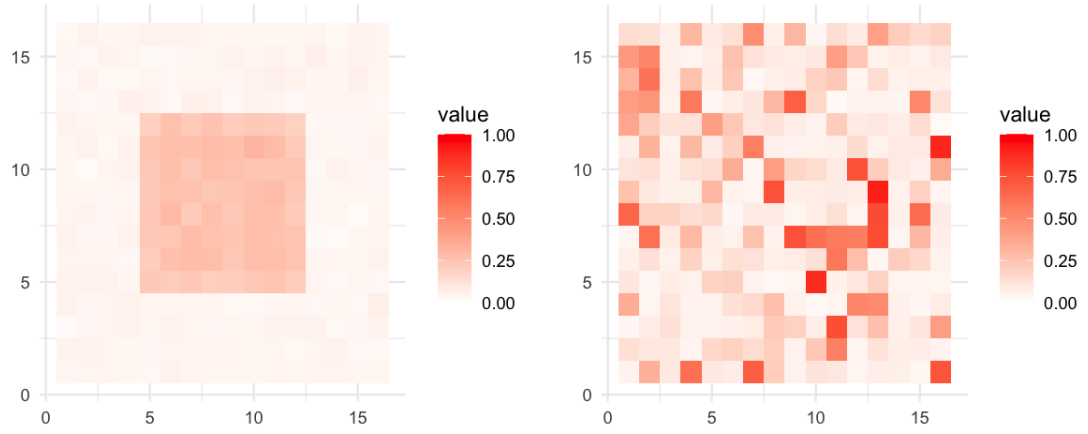

Figure 9: Percentage of significant p-values for elements of β when fitting models in the pixel space in Simulation 1 (left) and Simulation 2 (right).

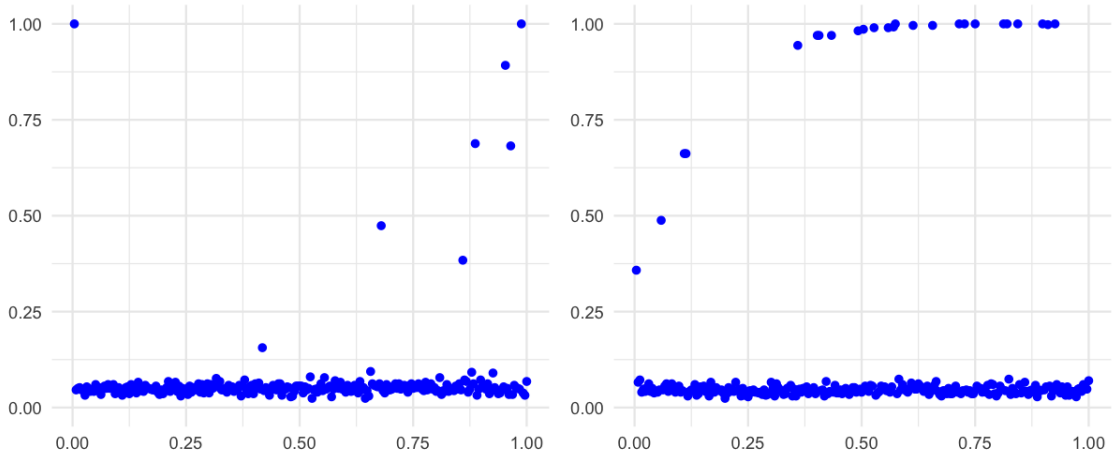

Figure 10: Percentage of significant p-values for elements of b across ordered eigenvalues in both simulations.

Future Work

- Adding details about how `hdi` package calculated p-values and why my permutation test didn't work.
- Increase b effect size (how to keep p evenly distributed in the same time?) see whether the pattern of coefficient estimates disappear or relieve.
- What is the next step in higher level?

References

Peter de Jong, C Sprenger, and Frans van Veen. On extreme values of moran's i and geary's c. *Geographical Analysis*, 16(1):17–24, 1984.

Daniel Griffith and Yongwan Chun. Spatial autocorrelation and spatial filtering. *Handbook of regional science*, pages 1477–1507, 2014.