Exploring the Use of a LASSO-penalized Logistic Regression with a Log Contrast Constraint in High-Dimensional Compositional Data Analysis

Anna C. St Jean
Research Advisor: Peter J. Mucha, PhD.
Dartmouth College
June 3, 2025

Abstract

As omics data derived from high-throughput technologies becomes more ubiquitous in biological and medical research, the development of predictive models that efficiently and accurately analyse and classify this data is paramount. The compositional nature of such data imposes constraints that must be considered in such models, and while existing transformations address such limitations, they result in high-dimensional feature spaces, limited accuracy, and/or sizable computational demands.

The log-contrast LASSO (LCL) model leverages the log-contrast constraint to implicitly represent log-ratios without expanding the feature space, significantly lowering computational complexity. Using rigorous nested cross-validation and a robust convex optimization solver, LCL produces sparse, interpretable logistic regression models. Experiments using synthetic data, publicly available 16S and WGS sequencing datasets, and two case studies reveal that the LCL model produces results competitive with those of the much more computationally expensive `DiCoVar` framework. Though `DiCoVar` often archives slightly better accuracy due to the ensemble learning methods it employs, LCL represents a valid, less-computationally intensive alternative that has the potential to meet or exceed `DiCoVar` performance with future integration of ensemble learning techniques.

Introduction

Metagenomic sequencing and metabolomic data give insight into how alterations of microbiomes in and on the human body are associated with human disease. Metagenomics analyzes the DNA content of microbial communities, providing insight into the type and quantity of microbes present. In contrast, metabolomics identifies and quantifies metabolites – the compounds produced by microbes.

Both technologies represent promising new methods of early detection and diagnosis of both chronic and infectious disease. However, the provision of such insights relies on the identification of microbial biomarkers indicative of the absence, presence, or advancement of a given disease. Current metagenomic sequencing techniques – namely, whole genome shotgun (WGS) and 16S ribosomal-RNA methods – produce count data with limits on the total number of reads, meaning that analysis of data is limited to relative rather than absolute comparisons (compositional data). That is, data form parts of a whole and their relative rather than absolute values are of interest. Metabolomic data is similarly structured in a compositional manner.

The constraints that arise in compositional data cannot be ignored in its analysis, but many standard approaches to classification of metagenomic or metabolomic data involve machine learning algorithms trained on untransformed relative abundance data (i.e., percentages). Such models that fail to account for compositional constraints lack both interpretability and generalizability. For example, models that do not consider compositional constraints cannot be generalized to other datasets in which there are differences in the features sampled – if a model is constructed using the relative abundances of features A, B, C but another dataset contains features A, B, C *and* D, the relative abundances of this new feature (D) will change those of the features used in the model, so this model cannot accurately generalize to the new setting with the additional feature. Further, the fixed total sum of 1 or 100% in untransformed compositional data can lead to spurious correlations between otherwise unrelated features. That is, because the sum is kept constant, changes in abundance of one component inherently lead to changes in abundance of others even if there is no direct causal relationship.

However, various transformations including the pairwise log ratio, additive log ratio, and centered log ratio transformations have been proposed to overcome these difficulties. Various studies combine these transformations with existing methods for predictive modeling, including machine learning and generalized linear regression models. However, machine learning models such as `DiCoVarML` (Hinton & Mucha, 2021) are extremely computationally expensive, while many GLM methods lack accuracy and some also interpretability and generalization.

Therefore, we seek to develop a framework for a less computationally expensive and potentially more accurate model that adheres to the constraints of compositional data analysis while providing effective feature selection for biomarker discovery and disease classification in metagenomic and metabolomic research.

Background

Aitchison (1982) proposes the use of log transformation as an effective method for compositional data analysis. This method is based upon coherence, the idea that ratios of parts do not vary with respect to the normalization of data. Relationships between parts remain constant.

Aitchison's 1982 paper represents the first exhaustive treatment of compositions using log ratios with the additive log ratio (ALR) transformation, described in detail below. This compositional data analysis, however, was performed on small datasets with ten or fewer components. Today, compositional data analysis with log ratios is typically employed for much higher dimensional data – such as omics data – in which feature selection is paramount and computational cost a key consideration. As a result, additional log ratio transformations subject to various forms of feature selection have been proposed to develop parsimonious models for predicting outcomes from omics data.

Key Log Ratio Transformations

In their 2023 paper, Greenacre et al. discuss the three key log ratio transformations commonly used in compositional data analysis – the pairwise log ratio transformation (PLR), additive log ratio transformation, and centered log ratio transformation (CLR).

A general pairwise logratio (LR) can be defined as follows, with j, k $\in$ {1,...,F} and j $\neq$ k:

$$(1)\ LR(j,k) = log(\frac{x_j}{x_k}) = log(x_j) - log(x_k)$$

For a given dataset with F features, $\frac{1}{2}(F)(F-1)$ unique log ratios can be constructed. For small datasets with few features, this represents a comprehensive and effective method of exploring all possible LRs and selecting features according to the chosen algorithm. However, when working with high-dimensional metabolomic or metagenomic datasets, the PLR transformation increases the feature space from F dimensions to $\binom{F}{2}$ dimensions, drastically increasing the computational cost of feature selection. Notably, all of these unique LRs are *not* linearly independent. Thus, though there are $\binom{F}{2}$ LRs that result from this transformation, their dimensionality is only $F-1$ (the same as with the ALR transformation described below) (Greenacre et al., 2023).

As an alternative to this exhaustive search, the ALR transformation can be employed. Among all pairwise LRs, there is a subset of F–1 LRs that share a denominator. In the ALR transformation, F–1 log ratios are calculated with this same denominator (also known as the reference part). There are F–1 total LRs because each feature with the exception of the reference part has an associated ratio calculated for it (thus in Equation (2), j $\neq$ *reference*).

$$(2)\ ALR(j|reference) = log(\frac{x_j}{x_{reference}}) = log(x_j) - log(x_{reference})$$

Because there are *F* features, there are therefore *F* potential reference parts and thus *F* possible ALR transformations. Though ALRs are useful in simplifying compositional data analysis results, the use of this transformation requires the selection of a given reference feature without knowing which feature is best for this purpose (Greenacre et al., 2023).

Finally, the CLR transformation takes the log ratio of each part with respect to their geometric mean g(x), thus representing a symmetric transformation of all parts. In the CLR transformation shown in Equation (4), $j \in \{1,...,F\}$.

$$(3)\ g(x) = (x_1, x_2, ..., x_F)^{1/F}$$

$$(4)\ CLR(j) = log(\tfrac{x_j}{g(x)}) = log(x_j) - \tfrac{1}{F}\sum_{k=1}^{F} log(x_k)$$

Because the denominator is the same for all possible CLRs, there are *F* CLRs that can be constructed for a given dataset (Greenacre et al. 2023). However, the calculation of this denominator as the geometric mean of all parts means that in the CLR transformation all features are retained, eliminating the possibility of any true feature selection.

Both PLR and ALR meet Aitchison's requirement of subcompositional coherence (Aitchison, 1982) while providing a simple interpretation of the constructed log ratios (Greenacre et al., 2023). Nonetheless, all three methods (among other transformations) have been employed in the analysis of omics data and construction of both supervised and unsupervised learning methods.

Methods of Feature Selection for Log Ratio-Transformed Data

Regardless of which one of the above transformations (if any) is employed, it is necessary to select a subset of log ratios for high dimensional datasets like those arising from mass spectrometry. We first provide a brief overview of unsupervised classification and discriminative balance methods. We then further examine two additional techniques most relevant to the approach proposed in this paper – the usage of generalized linear models and Hinton & Mucha's differential compositional variation machine learning framework (`DiCoVarML`) (2021).

Hron et al. (2013) introduce an unsupervised stepwise procedure using a variation of the CLR transformation (the isometric log ratio or ILR transformation) to reduce a *D*-dimensional composition to a sparse subcomposition. The ILR transformation enables compositional data to be represented in a real Euclidean space, allowing for the use of standard statistical methods. Additionally, Greenacre (2019) proposes an additional unsupervised technique, also using a stepwise method, to select the pairwise log ratios (PLRs) that explain (through linear regression) the largest proportion of logratio variance in the composition. However, we are more interested in supervised learning techniques that predict classes based upon the selected log ratios.

Much of the research on supervised compositional data analysis and classification has focused on generalized linear models. The selbal approach, developed by Rivera-Pinto et al. (2018), is a greedy stepwise

supervised algorithm that picks two microbial subcompositions – one positively correlated with the binary dependent variable and the other negatively correlated with it – and finds the log ratio of the geometric mean of the first over the second sub composition. In selbal, feature selection is applied at the parts level (rather than at the log ratio level) – the algorithm looks for two groups of taxa whose relative abundances are most associated with the outcome of interest according to a generalized linear model. However, selbal is computationally expensive and performance is limited in certain scenarios.

Various approaches using penalized regression have also been proposed as a means to develop sparse models. Lin et al. (2014) discuss the use of an LASSO-regularized linear log-contrast model, building upon the work of Aitchison & Bacon-Shone (1984) and their proposed log-contrast model.

Because of the unit-sum constraint intrinsic to compositional data, traditional methods of analysis necessitate the omission of specific components and therefore challenges with interpretation of results. However, the application of the log-ratio transformation results in the linear log-contrast model:

$$(5)\ y = Z^F \beta^*_{\setminus F} + \epsilon$$

where $Z^F = \{log(x_{ij}/x_{iF})\}$ is the $n \times (F-1)$ log-ratio matrix whose pth component is the reference component, $\beta^*_{\setminus F} = (\beta^*_1, ..., \beta^*_{F-1})^T$ is the corresponding vector of regression coefficients, and ϵ is a vector of independent noise (Aitchison & Bacon-Shone, 1984).

Lin et al. (2014) then add a LASSO regularization parameter, creating a convex optimization problem subject to the constraint that all regression coefficients sum to zero (see Equation 6). The addition of this log-contrast constraint allows for ease of interpretation above and beyond traditional LASSO-regularized estimators while creating a sparse model. Reinforcing and expanding upon their findings, Shi et al. (2016) propose an improved coordinate descent algorithm to implement LASSO penalization under the log-contrast constraint as well as a method of obtaining debiased coefficients. Lu et al. (2018) then extend Lin et al.'s approach to generalized linear models.

Bates & Tibshirani (2019) similarly develop a linear model initially containing all possible pairwise log ratios and use LASSO penalization to create a sparse model. However, they also introduce a subsequent pruning step to increase model sparsity and minimize overfitting. Because this model expands the feature space to $\binom{F}{2}$ dimensions, much of their paper is devoted to addressing the computational cost of developing such a model.

The development of generalized linear models constructed from CLR-transformed data has also been proposed. In the CLR-LASSO method, compositional data is projected to the Euclidean space according to the CLR transformation described in Equation (4). Then, various penalized regression methods (including LASSO, L2, and elastic net regularization) are applied to develop a sparse linear model. However, the application of this penalized regression onto CLR-transformed data limits the interpretability of the results and reduces the model's ability to detect real association, especially when the geometric mean *g(x)* has a high variance (Susin et al., 2020).

In addition to being used with generalized linear models, the CLR transformation has also been applied in machine learning models such as random forest models and support vector machines. However, the usage of CLR-transformed data in these frameworks leads to inaccurate AUCs and associated limitations in generalizability.

Motivated by this shortcoming of existing machine learning techniques in compositional data analysis as well as the limited accuracy of existing generalized linear models trained on PLR-transformed data, Hinton & Mucha (2021) propose the differential compositional variation machine learning framework (`DiCoVar`). `DiCoVar` combines the PLR and ALR transformations used in the aforementioned GLMs with ML techniques, attempting to obtain an optimal set of log ratios between parts for use as features in machine learning classification methods. These ALR- or PLR- transformed features are scored according to their variation between classes, and the algorithm then selects a small subset of these log ratios to predict classes (Hinton & Mucha, 2021).

`DiCoVar` uses nested k-fold cross-validation to minimize overfitting and accurately evaluate classification performance. While the package permits the use of various machine learning model paradigms, Hinton & Mucha concentrate on ridge regression and ensemble learning models. They find that the ALR transformation yields results with possibly lower insight at a lower computational cost while the PLR transformation often provides heightened insight but requires significantly more computational resources. As expected, the ensemble learning models tend to outperform single estimators (Hinton & Mucha, 2021).

Though `DiCoVar` applied to PLR-transformed metagenomic datasets shows good accuracy in the case studies performed by Hinton & Mucha (2021), it is *extremely* computationally expensive. As shown in Equation (1), PLR transformation expands the feature space to include $\binom{F}{2} = \frac{(F)(F-1)}{2}$ dimensions and the given machine learning framework is then presented with all possible pairwise log ratios. Considerable resources are needed to perform effective feature selection in this sizable and highly correlated feature space. While examining the predictive power of all possible pairwise log ratios ensures that no predictive ratios are 'missed', in some cases the `DiCoVar` framework is more computationally expensive than is perhaps necessary.

Therefore, we seek to develop a model performs at a level equivalent to `DiCoVar` without expanding the feature space to $\binom{F}{2}$ dimensions. To accomplish this, we propose the use of a LASSO-regularized logistic regression model subject to a log-contrast constraint. Such a framework requires significantly less computational resources than `DiCoVar` but may exhibit performance and interpretability above and beyond existing generalized linear regression models that use PLR, ALR, and CLR-transformed data.

While some literature exists detailing the development of a similar method (named Coda-LASSO; constructs both linear and logistic log-constrained regression models), this model is minimally documented, does not exist as a formal or maintained package, and uses an underpowered optimization approach (Susin et al., 2020). First, Coda-LASSO does not exist as a CRAN-installable R package or Conda-installable Python package. This is also a problem that arises in the aforementioned studies by Lin et al. (2014), Shi et al. (2016), and Lu et al. (2018), which contain extensive theoretical analyses but no package development. These studies, as well as Coda-LASSO, also perform minimal benchmarking – they examine model performance on only one or

two case studies or real datasets, which makes for much less robust conclusions concerning their ability to predict disease.

Second, there is no formal documentation of parameter selection – that is, it is unclear to us how parameters such as λ (the LASSO regularization parameter) are selected for use in the model. Rigorous cross-validation for parameter selection is vital to creating an effective model, so the model developed here provides a far more reproducible method for finding the optimal regularization parameter.

Finally, the optimizer used is a simple line search algorithm implemented in the model itself, leading to concerns about whether the algorithm truly finds the optimal solution. In preliminary analyses of metagenomic datasets, we find that the ECOS solver – an open source solver specializing in solving second-order cone problems and good for small- to medium-sized problems – often fails. This suggests that a line search algorithm may also fail in high-dimensional feature settings characteristic of metagenomic data.

Therefore, we seek to determine whether a LASSO-regularized logistic regression model with rigorous regularization parameter selection and commercial-grade optimizer may represent a less computationally expensive alternative to `DiCoVar` that similarly outperforms other existing GLMs computed on log-transformed data. To accomplish this, we conduct an in-depth analysis of model performance on synthetically generated data as well as ten real datasets, far exceeding the benchmarking rigor of many existing studies.

Methods

Understanding the Log Contrast Constraint

Given the the inaccuracies associated with using non-transformed relative abundance data, limited performance of existing methods of feature selection for log-transformed data such as `selbal`, and the computational requirements of `DiCoVar`, we propose the usage of a so-called log-contrast constraint paired with LASSO-penalized logistic regression to identify predictive metabolomic or metagenomic features and classify cases. The log-contrast constraint applied in a regression context is represented by the following equation (6), where β is the coefficient vector and F is the number of features:

$$(6)\ \sum_{j=1}^{F} \beta_j = 0$$

The usage of a log-contrast constraint rather than calculating all PLRs or ALRs can be explained as follows. Because metabolomic and metagenomic data are compositional rather than absolute in nature, we remain interested in the *ratios* between features. Consider the case in which the ratio between two features, A and B, predicts incidence of a given disease. If we take the log of each feature as in a logistic regression model, the features of this model are log(A) and log(B). Then, the log of the ratio between features A and B can be expressed as $log(A/B) = log(A) - log(B)$ – the log of the ratio is equivalent to the difference of the logs. Then, log(A) and log(B) can be used as features in a regression model – with their requirement that their coefficients be equal and opposite (that is, β_A and β_B must sum to zero) in order to accurately represent this relationship.

$$(7)\ \beta_A log(A) + \beta_B log(B) = \beta_A log(A) - \beta_A log(B) = \beta_A(log(A)) - log(B)) = \beta_A log(A/B)$$

Generalizing the above to a high-dimensional dataset, we can use this "log-contrast constraint" to ensure that the linear combination of all coefficients sums to 1 (Aitchison & Bacon-Shone, 1984). Then, per equation (6), it is clear that any model that could have been made using ALR- or PLR-transformed data can also be written as a log contrast. With this zero-sum requirement imposed, a LASSO-penalized logistic regression model can then be constructed to determine features relevant in predicting disease and classify individuals. This method of prediction is henceforth referred to as the "LCL framework."

Because this model does not involve the ALR or PLR transformations used in `DiCoVar`, there are inherently fewer possible models that can arise from it. As we have discussed, the PLR transformation results in a feature space of $\binom{F}{2}$ dimensions. Though individual ALR transformations do not increase the number of dimensions (each results in a feature space of size F–1), there exist F possible ALR transformations for a given

dataset. In sum, there are many more possible `DiCoVar` models than LCL models that could be constructed from a given dataset.

Specifically, while every PLR or ALR `DiCoVar` model is associated with exactly one log-contrast model, a given log-contrast model developed using the LCL framework associates with *multiple* ALR or PLR models. One log-contrast model associates with several ALR models (one for each possible reference part used), and with an entire continuous multidimensional family of potential PLR models. In short, the networks revealed using the `DiCoVar` framework (Hinton & Mucha, 2021) are each just one potential way of describing the ratios associated with the equivalent log contrast model.

Data Preprocessing

The LCL framework takes in mass spectrometry data (metabolomic data) or taxonomic tables from WGS or 16S sequencing (metagenomic data) as input. Since this data has a long tail, we first apply the base 10 log transformation to better approximate a normal distribution.

Since some input data includes zeros, which become –∞ when log-transformed, we must impute these undefined values. While there are numerous methods available for imputation, in the present work this is done by replacing zeros with the smallest nonzero value in the training set (Hinton & Mucha, 2021). Zero imputation occurs in the outer loop of the cross validation process (described in detail below) in order to avoid data leakage, because such leakage of information between the train and test set can lead to artificial inflation of classification performance metrics. Zero values in the outer loop test set are *also* replaced by the smallest nonzero value in the training set.

Model

The LCL model takes in this processed data and first rescales it to mimic the functionality of `glmnet`'s logistic regression model (Hastie et al., 2023). Then, the following objective function is used to implement a LASSO-regularized logistic regression. Please note that in Equation (8) and from this point forward, *x* represents the already-log-transformed features.

$$(8)\ Objective = -[\tfrac{1}{N}\sum_{i=1}^{N} y_i(\beta_0 + x_i^T\beta) - log(1 + e^{(\beta_0 + x_i^T\beta)})] + \lambda\sum_{j=1}^{F}|\beta_j|$$

$$(9)\ Log\ contrast\ constraint:\ \sum_{j=1}^{F}\beta_j = 0$$

The first term of this objective function represents the fitting of the logistic regression model and the second term is the LASSO penalization intended to reduce overfitting by penalizing the sum of the absolute values of the coefficients. The objective function and log-contrast constraint are then passed into CVXR, an R package that provides an object-oriented modeling language for convex optimization and allows the user to formulate convex optimization problems in a natural mathematical syntax (Fu et al., 2020).

Preliminary analysis revealed that the ECOS optimizer used by default in CVXR does not perform well and often fails when solving problems with high-dimensional data. To overcome this, the commercial-grade optimizer Mosek is used in the LCL framework for all analyses. Mosek is an optimization solver that works with CVXR to solve optimization problems with the high-dimensional characteristics of metagenomic or metabolomic data (MOSEK ApS, 2025).

Prior to implementation of this model, we verified consistency of results from CVXR and `glmnet`. As the state-of-the-art package used for logistic regression in R, we wanted LCL to mimic the functionality of `glmnet` (with the addition of the log-contrast constraint). Therefore, we ensured that the objective function used matched that of `glmnet` and that preliminary results of our function *without* the log-contrast constraint matched those of `glmnet` (Hastie et al., 2023).

After model generation, coefficients with absolute values below a certain threshold – set at 1e-6 for all models generated in this paper – are removed as a further means to reduce overfitting. While this means that the log-contrast constraint is not precisely met, the values removed are negligibly small.

Cross Validation & Measuring Performance

In order to select relevant features, minimize overfitting, and accurately estimate model performance on novel datasets, we employ a nested cross validation framework.

The purpose of the inner loop cross validation is to select the optimal λ (LASSO regularization parameter) based upon a performance metric of choice. Throughout this paper, we consistently use AUC as our performance metric of choice, because it is preferable to mean squared error (MSE) for data with unbalanced classes (which is the case with much of the real 16S and WGS data used).

The list of λ through which to iterate is generated as described by Friedman et al. (2010). First, we find the smallest value λ_{upper} for which the entire vector $\widehat{\beta} = 0$. Then, λ_{lower} is calculated by finding the value 1.5 orders of magnitude smaller than λ_{upper}. Finally, we generate 102 values of λ, evenly spaced on the log scale, between λ_{lower} and λ_{upper}.

For each λ, we then generate 20 random 80/20 train/test splits using the `createDataPartition` function in `caret`, which conducts within-class random sampling in an effort to balance class distributions between the splits (Kuhn, 2008). We used 20 splits because if there are not enough folds, the standard error for each λ is not sufficiently small to produce pertinent performance data or calculate a meaningful λ_{1SE} given the value of λ_{max}. After generating models and evaluating test set performance for each value of λ on each train/test split, the inner cross validation loop returns the λ (henceforth referred to as λ_{max}) that produces the highest mean AUC. It also returns the largest value of λ within 1 standard error of λ_{max} (henceforth referred to as λ_{1SE}). This value is of interest because its performance is not statistically significantly different from that of λ_{max}, but a larger value of λ (higher penalty term) produces a more sparse model less prone to overfitting. However, it is also possible that in some cases, λ_{max} could outperform λ_{1SE} if the model generated using λ_{1SE} is *too* sparse – therefore, we test both values of lambda in the LCL model. Figure 1 demonstrates how λ_{max} and λ_{1SE} are selected in this inner cross-validation loop.

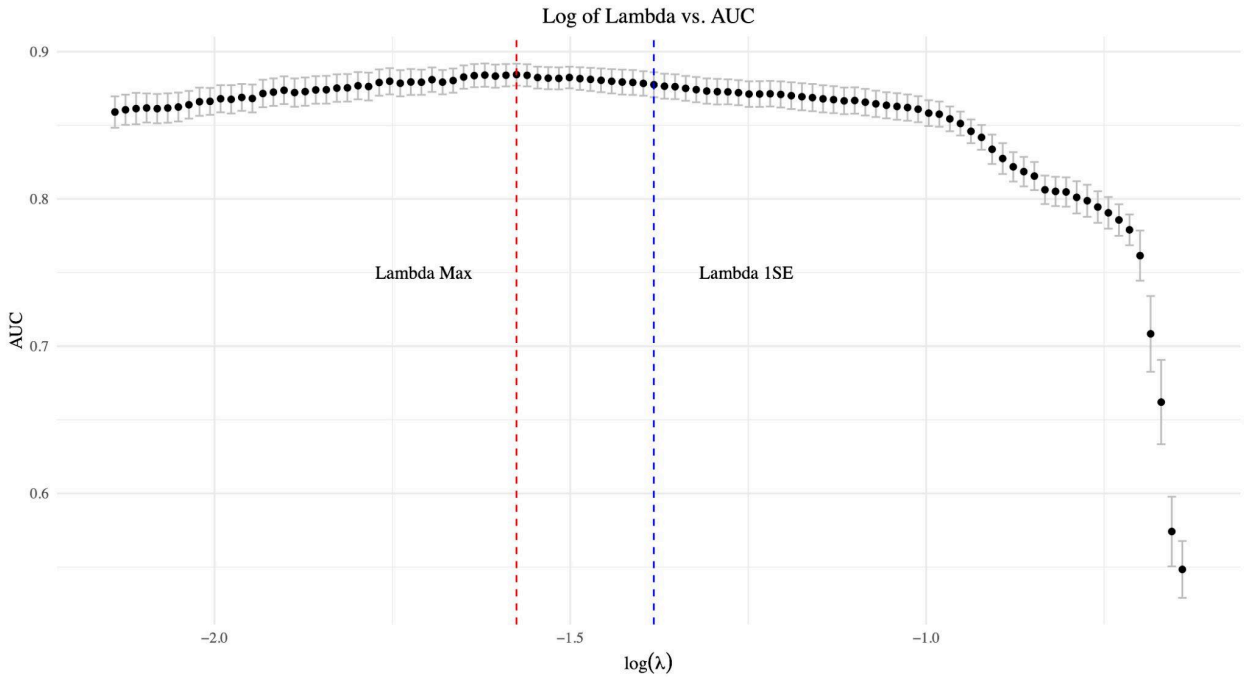

Figure 1 Selection of λ_{max} and λ_{1SE} using the inner cross-validation procedure on the full CDI dataset (Schubert et al., 2014). The blue dotted line marks λ_{1SE} and the red dotted line represents λ_{max}. The black points mark the mean AUC calculated using left-out test data for each of the 20 train/test splits, and the gray error bars represent one standard error above and below the mean AUC calculated for each λ. Note the sharp decrease in model performance at $\lambda \approx 0.66$ (or $\log(\lambda) \approx -0.18$), which reveals that when the model becomes too sparse (as a large λ results in higher penalization of additional features), performance drops rapidly.

Importantly, preliminary trials revealed that the CVXR solver failed seemingly at random for certain splits in combination with certain test values of λ. While this matter certainly warrants further investigation, it happened infrequently (at most once in a full run through the outer loop) and was addressed by implementing R's `tryCatch` function to simply skip a run through the inner loop and calculate the mean and standard error AUC for the given λ without data from this iteration.

This inner loop is then nested within an outer loop whose purpose is to determine how we expect the model to perform in real life. We use 25 random 80/20 train/test splits, again generated using `caret::createDataPartition`. 25 outer splits are used because we need sufficient folds to generate a meaningful mean and standard deviation of model performance.

The full dataset is fed into this outer loop. Then, in each iteration of the outer loop, the full inner loop cross validation process is performed. Each inner loop returns the chosen λ_{max} and λ_{1SE}, which are then each used to fit a model to the training data in the outer loop. After each of the two models is fit, their performance on the test data (which never touches the inner cross-validation loop and therefore represents a novel dataset) is evaluated using the AUC. After all 25 inner loops are performed, the mean and standard of the 25 AUCs generated for each of λ_{max} and λ_{1SE} are calculated to reveal the expected performance of the LCL model on a completely novel dataset.

Synthetic Data Generation

In order to verify the performance of the LCL framework before testing it on real metagenomic and metabolomic data and determine how the model might perform under different conditions (including data with a small number of observations, high-dimensional data with many noisy features, high signal to noise ratio, and unequal class distribution), we generated synthetic data representing already-log-transformed metabolomic/metagenomic data.

All synthetic datasets contained two features of interest, X1 and X2, with the decision boundary based upon the ratio of these two features. For the case group (A), X1 observations were randomly generated from a Gaussian distribution with a mean (μ) of J and variance (σ^2) of 1 and X2 observations were from a Gaussian distribution with $\mu = -J$ and $\sigma^2 = 1$. For the control group (B), X1 observations were randomly generated from a Gaussian distribution with a $\mu = -J$ and $\sigma^2 = 1$ and X2 observations were from a Gaussian distribution with $\mu = J$ and $\sigma^2 = 1$. All noisy features for both A and B were generated from a Gaussian distribution with $\mu = 0$ and $\sigma^2 = 1$. Increasing the value of J used for the mean in X1 and X2 increases the signal to noise ratio (SNR), while decreasing the value decreases the SNR. Rather than manipulating the variance to achieve this, only the means are modified to examine the effect of SNR on model performance.

We started with the simplest case – a three part composition with one noisy feature and a decision boundary based upon the ratio of the other two features. Per the data generation described above, we used $J = 1$. That is, for the case group (A), X1 observations were randomly generated from a Gaussian distribution with a mean (μ) of 1 and variance (σ^2) of 1 and X2 observations were from a Gaussian distribution with $\mu = -1$ and $\sigma^2 = 1$. For the control group (B), X1 observations were randomly generated from a Gaussian distribution with a $\mu = -1$ and $\sigma^2 = 1$ and X2 observations were from a Gaussian distribution with $\mu = 1$ and $\sigma^2 = 1$. This data contained an equal number of observations of control and case observations.

The relative values of X1 and X2 and the noisy feature were chosen to ensure that the log contrast of X1 and X2 would be a better predictor (i.e. greater signal to noise ratio) than the log contrast between either of X1 and X2 and the noisy feature. This ensured that the LCL model would learn the relationship between X1 and X2 in order to predict classes rather than learning to predict classes based upon the relationship between X1 or X2 and the noisy feature. Such a phenomenon would prevent meaningful comparison between the optimal decision boundary AUC defined by the relationship between X1 and X2 and mean model AUC (since the model would be learning a different decision boundary).

After examining LCL performance with this initial case, we then generated data with different numbers of observations, additional noisy features, different signal-to-noise ratios (achieved by increasing the absolute value of the mean of observations in X1 and X2), and different proportions of control vs. case observations.

Publicly Available & Case Study Data

Given the promising results yielded through analysis on the above synthetic data, we then conducted additional analysis on numerous publicly available WGS and 16S gut microbiome datasets used to benchmark `DiCoVar` performance in Hinton & Mucha (2021).

Three datasets were extracted from version 3.3.3 of the `curatedMetagenomicData` R packages (Pasolli et al., 2017). This package, which enables access to standardized metagenomic datasets, was used by Hinton & Mucha (2021) to extract the following studies by study ID: ZhuF_2020, RubelMA_2020, ZellerG_2014. The ZhuF_2020 dataset ("Schizophrenia") examines the association between gut microbiome features and schizophrenia, and consists of 171 observations (n = 171) and 480 features (F = 480) (Zhu et al., 2020), the RubelMA_2020 dataset ("STH") explores metabolomic correlates of soil-transmitted helminth in Cameroonians (n = 175 and F = 366) (Rubel et al., 2020), and the ZellerG_2014 dataset (Zeller 2014, CRC) examines fecal microbiota for early detection of colorectal cancer (n = 156 and F = 652) (Zeller et al., 2014). Please note that an additional dataset was extracted and examined by Hinton & Mucha (2021), and we attempted to apply the LCL framework to this data (on alterations of the human gut microbiome in liver cirrhosis) as we did with all other publicly available datasets. Despite our efforts to manipulate the dataset and the construction of the associated LCL models, however, we were unable to develop a viable LCL model without generating an error in R.

Further, Hinton & Mucha (2021) acquired metagenomic data concerning the association between gut microbiota and nonalcoholic fatty liver disease ("NAFLD"; n = 185 and F = 341) (Sharpton et al., 2018) from Qitta, an open-source online multi-omics platform (Qitta Study ID 11635). They also used MicrobiomeHD, an online database of uniformly-processed case-control studies, to source microbiome data distinguishing patients with Clostridium difficile infections ("CDI"; n = 336 and F = 259) (Schubert et al., 2014). Finally, they used the selbal R package to extract two more datasets – one examining the association between gut microbiome features and Crohn's disease ("Crohn"; n = 975 and F = 48) and the other HIV ("HIV"; n = 155 and F = 60) (Rivera-Pinto et al., 2018).

After examining these publicly available datasets, we also performed the two case studies conducted by Hinton & Mucha on fecal metagenomic data predicting necrotizing enterocolitis in infants ("NEC") and colorectal cancer ("CRC"). The NEC dataset comes from a longitudinal study containing 1100 microbiome-profiled fecal samples from 150 NICU infants (Olm et al., 2019). CRC case study data consists of 11 individual datasets sourced from the aforementioned `curatedMetagenomicData` R package for a total of 1305 samples.

For additional information on the extraction and processing of publicly available and case study datasets, please see Hinton & Mucha (2021).

Results

Classification Performance Evaluated Using Synthetic Data

We analyzed this synthetic data with 80/20 train/test splits for both the inner and outer loops, 25 passes through the outer loop, and 20 passes through the inner loop. For cases (group A), feature 1 (X1) and feature 2 (X2) were distributed normally with a mean (μ) of $-J$ and variance (σ^2) of 1 and a $\mu = J$ and $\sigma^2 = 1$ respectively (where J represents a positive nonzero value). For controls (group B), X1 and X2 were distributed normally with a $\mu = -J$ and $\sigma^2 = 1$ and a $\mu = J$ and $\sigma^2 = 1$ respectively. The noisy feature was normally distributed with $\mu = 0$ and $\sigma^2 = 1$.

Because X1 and X2 represent logged observations S1 and S2 such that $log(S1/S2) = log(S1) - log(S2) = X1 - X2$, the known decision boundary V can be computed as follows. Note that A1 is the mean value of X1 for cases, A2 is the mean value of X2 for cases, B1 is the mean value of X1 for controls, and B2 is the mean value of X2 for controls.

$$(10)\ V = \frac{1}{2}(A1 - A2 + B1 - B2) = \frac{1}{2}(-J + J - J + J) = 0$$

Three Part Composition

First, we use the simple three-part composition described above to determine the viability of the LCL framework. This synthetic dataset was constructed using $J = 1$ and contained 200 observations ($n = 200$) equally distributed between the control and case classes ($p = 0.5$).

Using this known decision boundary, we found the optimal AUC to be 0.9788. Using the LCL framework, we achieved a mean AUC of 0.914 for models generated using λ_{max} and a mean AUC of 0.975 for models generated using λ_{1SE} – almost equivalent to the performance using V as defined above. This demonstrates that the LCL framework effectively learns the predictive features – at least for simple synthetic data with small variance in features that represent both signal and noise.

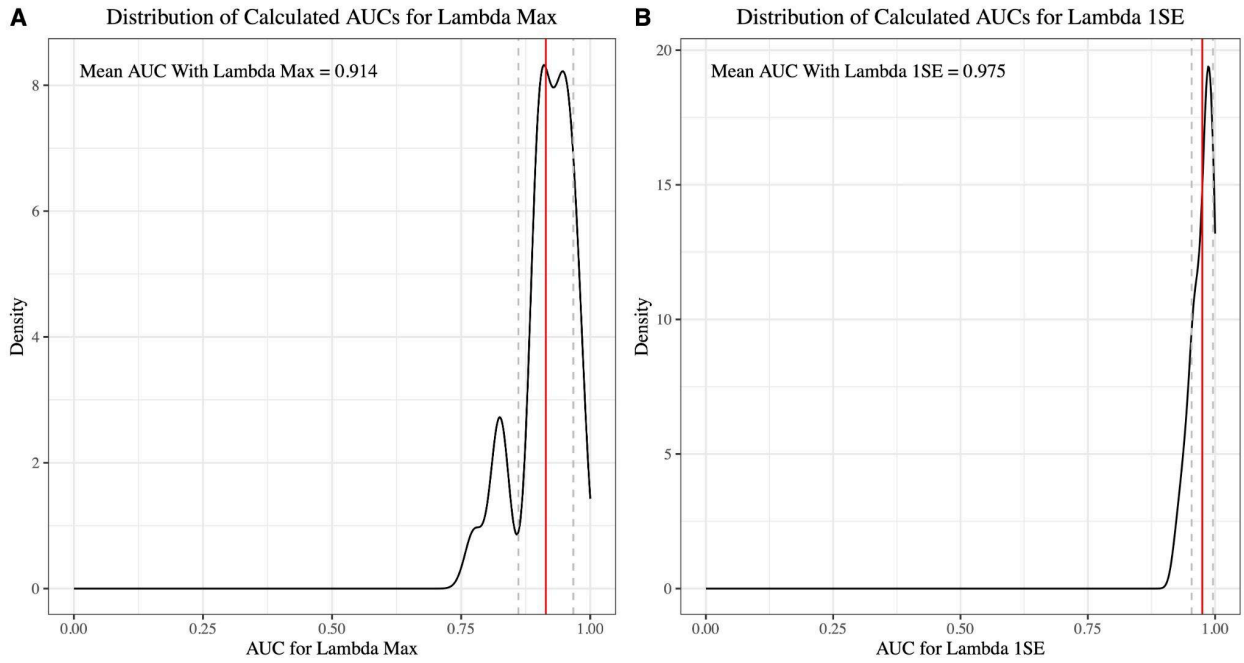

Figure 2 Performance of LCL framework on three-part composition synthetic data using λ_{max} and λ_{1SE} (optimal regularization parameter derived through inner-cross validation as described in Methods). The red line marks the mean calculated AUC, and the dotted lines designate the values one standard deviation above and below the mean AUC. This three-part composition is generated with $J = 1$ (and therefore SNR = 4 per Equation 11), $n = 200$, and $p = 0.5$ (equal number of control and case observations). As a three-part composition, there are two relevant features X1 and X2 and one noisy feature.

Manipulating Dataset Size

After discovering the excellent performance of the LCL framework on this toy three-part dataset, we then went on to test how model performance varies with number of observations. We generated additional simulated datasets with $n = 50$, 100, and 500. For these datasets, the following parameters were used: $F = 5$ (that is, there are five noisy features in addition to X1 and X2), $J = 1$, $\sigma^2 = 1$ (for both X1 and X2 as well as the noisy features), and $p = 0.5$ (observations equally distributed between the control and case conditions). As shown in Figure 3A, this analysis revealed that there is no notable within-λ difference in LCL performance (measured by AUC) based upon number of observations. The LCL framework regularized with λ_{1SE} performs almost optimally (that is, approximately equivalently to the AUC based on the known decision boundary) for all n. While the LCL framework regularized with λ_{max} results in a mean AUC that is consistently lower than that with λ_{1SE} (and therefore lower than the optimal performance), the λ_{max} results do not vary markedly with n. The mean AUC for each n ranges from 0.892 (when $n = 100$) to 0.918 (when $n = 500$). Notably, the standard deviations in AUC decrease with increasing number of observations for both the λ_{max} and λ_{1SE} models such that they no longer overlap at $n = 500$ – both lines are relatively flat but the deviations become smaller as n increases. However, the AUC standard deviation for λ_{max} remains consistently greater than that of λ_{1SE} even as they both decrease.

Manipulating Dimensionality

Next, we simulated data with progressively larger numbers of features (F) to emulate the high dimensionality of metabolomic and metagenomic data. We created datasets with F = 5, 10, 50, 100, and 250 noisy features, in addition to X1 and X2 (for a total of 7, 12, 52, 102, and 252 features in each dataset). For these datasets, the parameters used were as follows: $J = 1$, $\sigma^2 = 1$ (for both X1 and X2 as well as the noisy features), and $p = 0.5$ (observations equally distributed between the control and case conditions). As in our experiment manipulating the number of observations in the dataset, changing the number of noisy features did not yield notable differences in LCL framework performance within lambdas (see Figure 3B).

The performance of the LCL model constructed with λ_{1SE} did not vary *at all* depending upon the number of noisy features. It remained consistent at 0.975, just below the optimal performance based upon the known decision boundary (AUC = 0.979). For each n, σ_{1SE} (the standard deviation of the AUC of the model constructed with λ_{1SE}) remained constant at 0.0210, demonstrating little variation in model performance within each noise level. Note that the optimal decision boundary remained constant at each n because the data generated for X1 and X2 was identical across noise levels.

While there was more variability in performance within each noise level F with the LCL model constructed with λ_{max} (the standard deviation σ_{max} at each F ranged from 0.0474 to 0.0557), there was also no major difference in performance between λ_{max} models trained on models with varying numbers of noisy features. Notably, there were minor differences in performance between different noise levels using LCL with λ_{max}, unlike in the models using λ_{1SE} which exhibited identical performance regardless of noise.

However, like in the preceding experiment manipulating n, we again saw that the mean AUC for the LCL model constructed with λ_{1SE} was consistently above that of the LCL model constructed with λ_{max} for each noise level. That is, models constructed with λ_{1SE} consistently outperformed those constructed with λ_{max} regardless of F.

Manipulating Signal to Noise Ratio

Then, we constructed simulated datasets varying the magnitude of J and therefore the signal-to-noise ratio (SNR) of features X1 and X2. Since we hold $\sigma^2 = 1$ for all constructed datasets, we have that

$$(11)\ SNR = \frac{|(A1-A2)-(B1-B2)|}{\sigma} = \frac{|(J--J)-(-J-J)|}{\sigma} = \frac{4J}{\sigma} = 4J$$

These simulations used J = 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2.5, 5, or SNR = 0.4, 1, 2, 3, 4, 6, 10. As anticipated, the results (Figure 3C) demonstrated that as SNR increases, so does model performance. When SNR = 0.4 (very small), the optimal AUC based upon the known decision boundary is just above chance at 0.556. The mean AUC of the LCL model regularized with λ_{max} is 0.479 and with λ_{1SE} it is 0.486. There is no notable difference in performance between the LCL model with λ_{1SE} versus λ_{max}.

As the signal to noise ratio increases – that is, the signal becomes stronger – so does the AUC according to the known decision boundary and the performance of the LCL model using both λ_{max} and λ_{1SE}. However, the

improvement in performance of LCL with λ_{1SE} occurs at a faster rate than that of the LCL model constructed using λ_{max}. While the performance of LCL with λ_{1SE} is comparable to that of LCL with λ_{max} at SNR = 0.4, the performance of LCL with λ_{1SE} increases to approximately the AUC using the known decision boundary at the next tested SNR of 1 (the model mean AUC is 0.655 and that of the known decision boundary is 0.667). The mean AUC of the LCL model with λ_{max} remains well below these values at 0.596.

Overall, as SNR increases, the performance of λ_{1SE} continues to be consistently better than that of λ_{max}. The mean AUC of the LCL model with λ_{1SE} remains within 0.007 of the known decision boundary AUC for SNR > 0.4. In contrast, the mean AUC of the LCL model with λ_{max} is 0.0705 less than that of the known decision boundary when SNR = 1, 0.0969 less when SNR = 2, and 0.1002 when SNR = 3. While this difference then gradually decreases as the SNR continues to increase, it is important to note that the performance of the LCL model constructed using λ_{max} lags behind that of the model constructed with λ_{1SE} as the signal to noise ratio increases.

Manipulating Class Distribution

Finally, we explored how the distribution of observations between the control and case groups may impact performance of the LCL framework. We generated datasets varying the value of p, the parameter dictating the proportion of cases belonging to the control from, to include values of 0.15, 0.3, 0.5, 0.7, and 0.85. For these datasets, the parameters used were as follows: $J = 1$, $\sigma^2 = 1$ (for both X1 and X2 as well as the noisy features), and $F = 5$ (five noisy features in addition to the two signal features X1 and X2).

Upon generating models using the LCL framework with λ_{max} and λ_{1SE}, we found that while there was no difference between the performance of the LCL model with λ_{1SE} across the different control proportions (p), the performance of the LCL model with λ_{max} *did* vary based upon the value of p. there *were* values of p (see Figure 3D).

For all p, we see that the performance of the LCL model with λ_{1SE} shows little variation within trials for the same p (the standard deviation is approximately 0.025 for all of them). Further, the mean AUC of the λ_{1SE} model is consistently approximately equal to AUC calculated from the known decision boundary. The mean AUC of the λ_{1SE} model ranges between 0.972 and 0.985, never dropping below the known decision boundary AUC by more than 0.013 and sometimes equivalent to it. There is no visible trend in LCL λ_{1SE} performance. The AUC from the known decision boundary also varies slightly between datasets with different p due to randomness in simulated data generation, so it is reasonable to assume that the slight variation in performance seen in the LCL λ_{1SE} results reflects this stochasticity rather than any true difference in performance as a function of class distribution.

However, the mean performance of LCL models constructed using λ_{max} is consistently lower for every value of p, with greater differences between λ_{max} and λ_{1SE} performance generally occurring at more extreme class distributions (that is, those that are majority case or majority control observations). The mean AUC of LCL models constructed with λ_{max} is relatively similar to that of the λ_{1SE} models at $p = 0.5$ – their values are 0.915 and 0.975 respectively. The standard deviation of the AUC for λ_{max} models is 0.0515, which is larger than that of the λ_{1SE} LCL model but not drastically so (however, this does reveal more variation within trials for the same p for the λ_{max} LCL model).

Even more interestingly, mean performance for LCL λ_{max} decreases on either side of $p = 0.5$, and the variance increases. For all datasets except that simulated with $p = 0.5$, the mean AUC of the LCL models constructed using λ_{max} does *not* fall within two standard deviations of the mean AUC of that constructed using λ_{1SE}. This effect is especially notable at $p = 0.7$, where the mean AUC of LCL λ_{max} is 0.638 with a standard deviation of 0.207 while the mean AUC of LCL λ_{1SE} is 0.985 with a standard deviation of 0.0174. In summary, the difference in performance between LCL models constructed using λ_{max} and λ_{1SE} tends to become more pronounced at more extreme values of p.

Importantly, this experiment examining the effect of class distribution LCL model performance was rerun with a different seed to determine whether these somewhat unexpected results were due to the random seed or splits. However, rerunning this with a different seed leads to similar results and a qualitatively analogous takeaway.

This analysis of synthetic data reveals three key insights. First, as anticipated, performance of the LCL framework with both λ_{max} and λ_{1SE} depends significantly upon the signal to noise ratio of the data used, as expected. Second, LCL model performance is sensitive to unequal class distribution – but only when using λ_{max}, not λ_{1SE}. This suggests that models constructed using λ_{max} are prone to overfitting. Third, and perhaps most importantly, the LCL model regularized using λ_{1SE} consistently outperforms (as measured using mean AUC on held-out test data) the LCL models regularized using λ_{max} across all datasets generated using different parameters. In addition to λ_{max} sensitivity to unequal class distribution discussed previously, the elevated performance of λ_{1SE} relative to λ_{max} presents further evidence that the LCL model overfits for λ_{max}. Such a tendency toward overfitting was considered in the ensuing evaluations of model performance on real metabolomic and metagenomic data, which are higher dimensional than simulated data and therefore even more at risk for overfitting. For all ensuing model creation, λ_{1SE} is used on the basis of these results.

Notably, the consistently high model performance shown in experiments manipulating dataset size, number of noisy features, and proportion of observations in the control group (Figure 3A, 3B, and 3D) may be at least partially due to the relatively high signal-to-noise ratio of the data when $J = 1$ (this is the parameter used for all synthetic data analyzed in these three parts). Further exploration with additional synthetic datasets is needed to more thoroughly examine the effects of n, F, and p on LCL model performance.

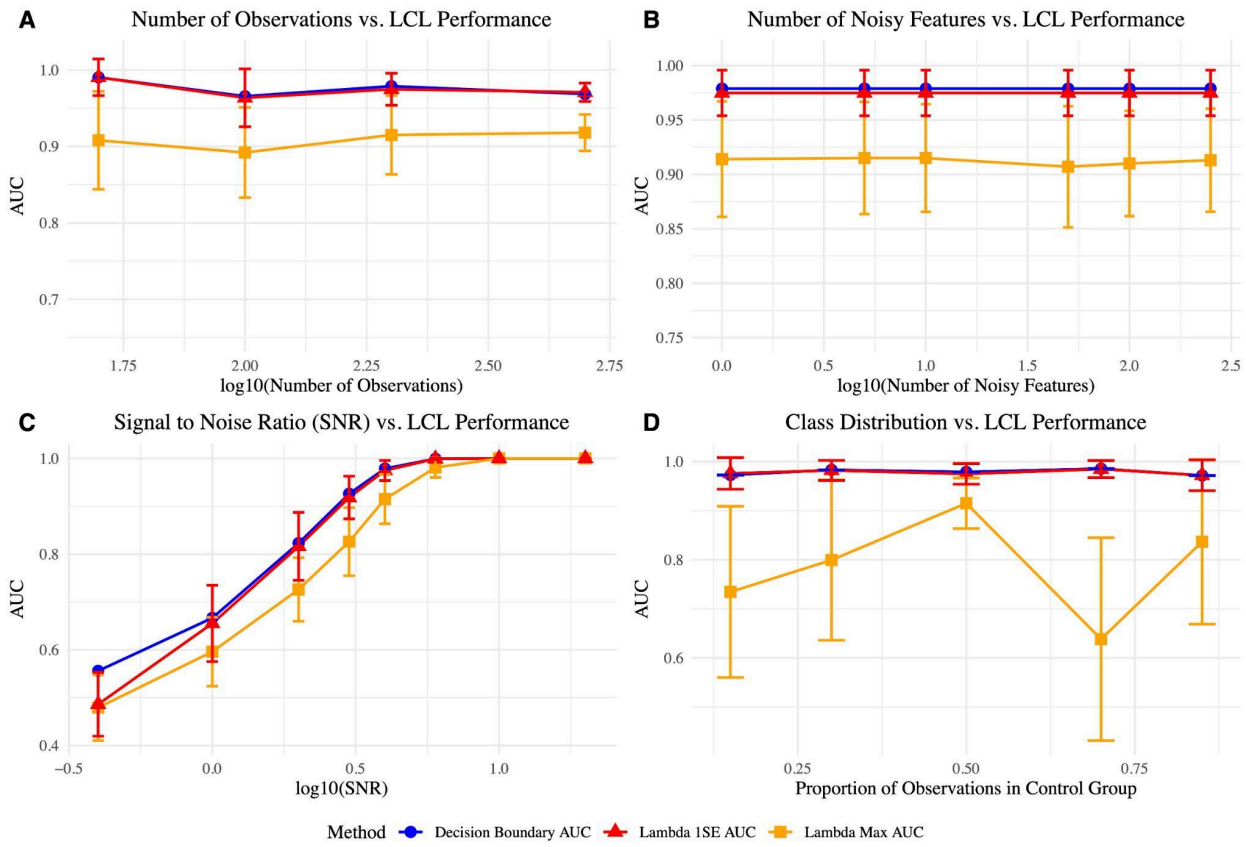

Figure 3 Results of synthetic data experiments with dataset parameter manipulation, showing the performance (measured using AUC) of the LCL model using λ_{1SE} and λ_{max} as well as the AUC calculated using the optimal decision boundary (representing the best possible performance). Error bars show the values one standard deviation above and below the calculated mean AUC. (A) examines the effect of the number of observations (n) on performance with λ_{max} and λ_{1SE}. All other parameters are held constant, with $J = 1$, $F = 5$, $p = 0.5$, and $\sigma = 1$ for both noisy and relevant features. This experiment reveals that while λ_{max} consistently underperforms λ_{1SE}, there is no major difference in model performance as a function of number of observations. (B) examines the effect of the number of noisy features (F) on performance with λ_{max} and λ_{1SE}. All other parameters are held constant, with $J = 1$, $n = 200$, $p = 0.5$, and $\sigma = 1$ for both noisy and relevant features. This experiment reveals that while λ_{1SE} consistently outperforms λ_{max}, there is no notable difference in within-λ model performance as a function of number of noisy features. (C) explores the effect of the signal to noise ratio (manipulated by changing the value of J) on performance with λ_{max} and λ_{1SE} (see Equation 11 for calculation of SNR). All other parameters are held constant, with $F = 5$, $n = 200$, $p = 0.5$, and $\sigma = 1$ for both noisy and relevant features. As expected, this experiment shows that performance improves as SNR increases – however, this improvement in performance occurs more rapidly as SNR increases for λ_{1SE} than λ_{max}. Finally, (D) examines the effect of class distribution (as determined by p, the proportion of observations in the control group) on LCL model performance. All other parameters are held constant, with $J = 1$, $F = 5$, $n = 200$, and $\sigma = 1$ for both noisy and relevant features. While p does not impact LCL model performance when λ_{1SE} is used, the performance of the model with λ_{max} *does* vary – λ_{max} model performance tends to be worse for more extreme values of p.

Classification Performance Evaluated Using Publicly Available 16S & WGS Datasets

Next, we used seven publicly available 16S and WGS datasets to evaluate the performance of the LCL framework relative to existing `DiCoVar` and Coda LASSO methods described in the Background section. In the preliminary models created, we consistently used an 80/20 train test split for both the inner and outer cross-validation loops, lambdas generated as outlined in Methods, and 20 inner and 25 outer folds. This combination of parameters showed the best performance in preliminary experiments exploring the effects of differing train/test splits and differing numbers of inner and outer splits.

The results are shown in Figure 4. Importantly, the results displayed for `DiCoVar` and Coda-LASSO classification frameworks are those published by Hinton & Mucha (2021) – we did not recompute these results. Thus, when comparing performance results between `DiCoVar` and LCL or between Coda-LASSO and LCL it is vital to consider that Hinton & Mucha (2021) use different parameters for the nested cross-validation and different random splits. For analysis of both these publicly available 16S and WGS datasets as well as the case studies described in the following section, the rigor of cross-framework comparisons is therefore limited.

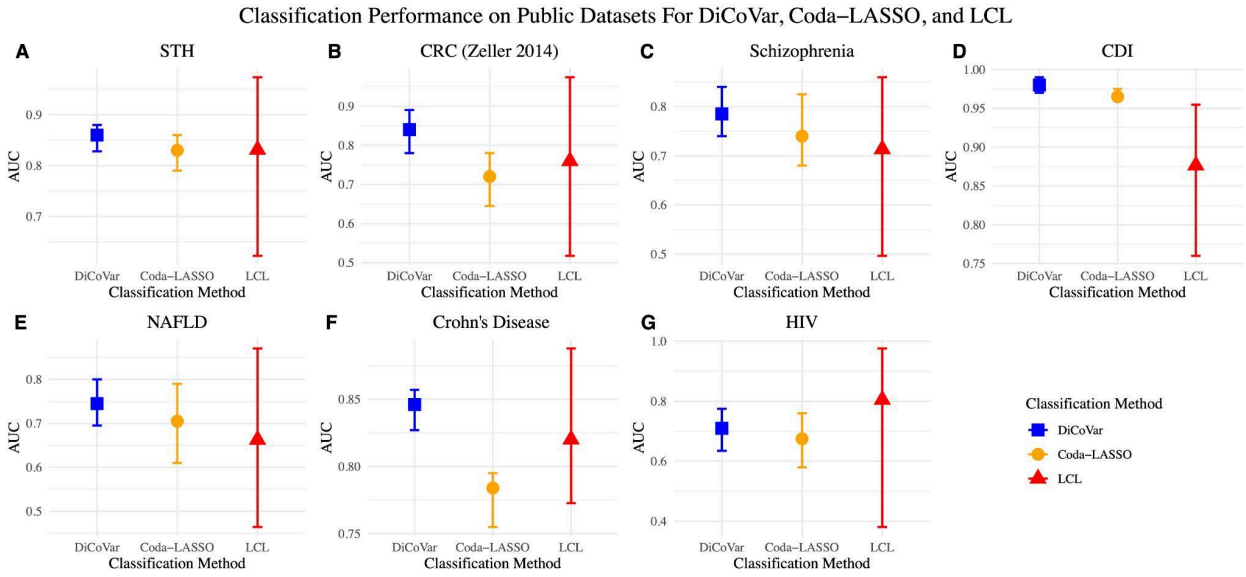

Figure 4 Each graph shows the performance, measured using mean AUC, for each of `DiCoVar`, Coda-LASSO, and λ_{1SE} -regularized LCL classification frameworks on a given publicly available WGS or 16S dataset. The version of `DiCoVar` used to produce the displayed results varies by dataset – the results shown are the results from the *best performing* `DiCoVar` *model* (either PLR, ALR, or hybrid signatures) constructed by Hinton & Mucha (2021). The error bars show the minimum and maximum AUC values observed in each classification method. The LCL results use 20 inner and 25 outer splits, but note that the results from the `DiCoVar` and Coda-LASSO models, from Hinton & Mucha (2021), use different parameters for the nested cross-validation and different random splits. For more information on the dataset size, number of features, and other metrics for each of the datasets analyzed here, please see Table 1.

Dataset Name	Number of Features (F)	Number of Observations (n)	F/n
STH	366	175	2.091
CRC (Zeller 2014)	652	156	4.179
Schizophrenia	480	171	2.807
CDI	259	336	0.771
NAFLD	341	185	1.843
Crohn's Disease	48	975	0.049
HIV	60	155	0.387

Table 1 This table includes the number of features F, the number of observations n, and the ratio between the number of features and number of observations (F/n) for each of the publicly available datasets examined in Figure 4.

Here, we see that the optimal `DiCoVar` framework (ALR, PLR, or hybrid depending upon the dataset) almost always outperforms both Coda-LASSO and the LCL framework based upon mean AUC, except in the case of the HIV dataset (Figure 4G). However, the magnitude of the difference in mean performance is generally not large relative to the variation in performance. Furthermore, Coda-LASSO and LCL often produce results with comparable centers. The mean LCL performance exceeds that of Coda-LASSO in four of the seven datasets and is worse in the remaining three, though Coda-LASSO outperforms LCL by a large margin on the CDI dataset (Figure 4D).

Notably, the variation in performance results from the LCL framework appears in Figure 4 to be consistently greater than that of the Coda-LASSO or LCL frameworks. That is, the difference between the mean AUC and the minimum and maximum observed AUC is much greater for the LCL results than the Coda-LASSO or `DiCoVar` results. While this may be the case, it is important to consider that the differing parameters and random splits used by Hinton & Mucha (2021) to evaluate the performance of `DiCoVar` and Coda-LASSO on each dataset likely contribute to this difference. For example, there are many more outer splits used to evaluate the performance of the LCL model, making it more likely that more extreme minimum and maximum performance metrics will be observed.

Two datasets tested generate results of particular note, the first being the clostridium difficile infection dataset (CDI) (n = 336 and F = 259). While the performance of `DiCoVar` and Coda-LASSO are relatively similar (with mean AUCs of 0.980 and 0.965 respectively), the mean AUC of models constructed using the LCL framework is 0.876 – notably much lower (see Figure 4D). Further, while there is very little variation in performance among both `DiCoVar` and Coda-LASSO models (smaller range than all other datasets examined), there is a difference of almost 0.2 between the min and max performance of models constructed using LCL. While a strict comparison of spread between the models is not feasible given the constraints explained above, it is notable that the relative difference in spread between `DiCoVar`/Coda-LASSO and LCL is much larger for the CDI dataset than for the other datasets examined.

Second, the mean performance of the LCL framework exceeds that of `DiCoVar` *and* Coda-LASSO on the HIV dataset (n = 155 and F = 60). The LCL framework performs with a mean AUC of 0.805, while `DiCoVar` produces a mean AUC of 0.710 and Coda-LASSO produces a mean AUC of 0.675. While there is much variation in performance for the LCL framework (the minimum observed AUC is 0.381 – worse than chance – and the maximum observed is 0.976), it is certainly notable that on average, its performance exceeds both other models for this specific dataset.

Classification Performance Evaluated Using NEC & CRC Case Studies

After exploring the performance of the LCL model on these relatively small, publicly available WGS and 16S datasets, we concluded by comparing the performance of LCL on larger case study datasets consisting of observations compiled from numerous separate studies.

The first case study dataset that we examined focused on colorectal cancer (CRC) and consisted of 1395 independent observations with F = 934. The performance of LCL with each of λ_{max} and λ_{1SE} is shown in Figure 5. Here, we see that the LCL framework regularized with λ_{1SE} (mean AUC = 0.802) performs better than that with λ_{max} (mean AUC = 0.735), as expected given our findings using synthetic data.

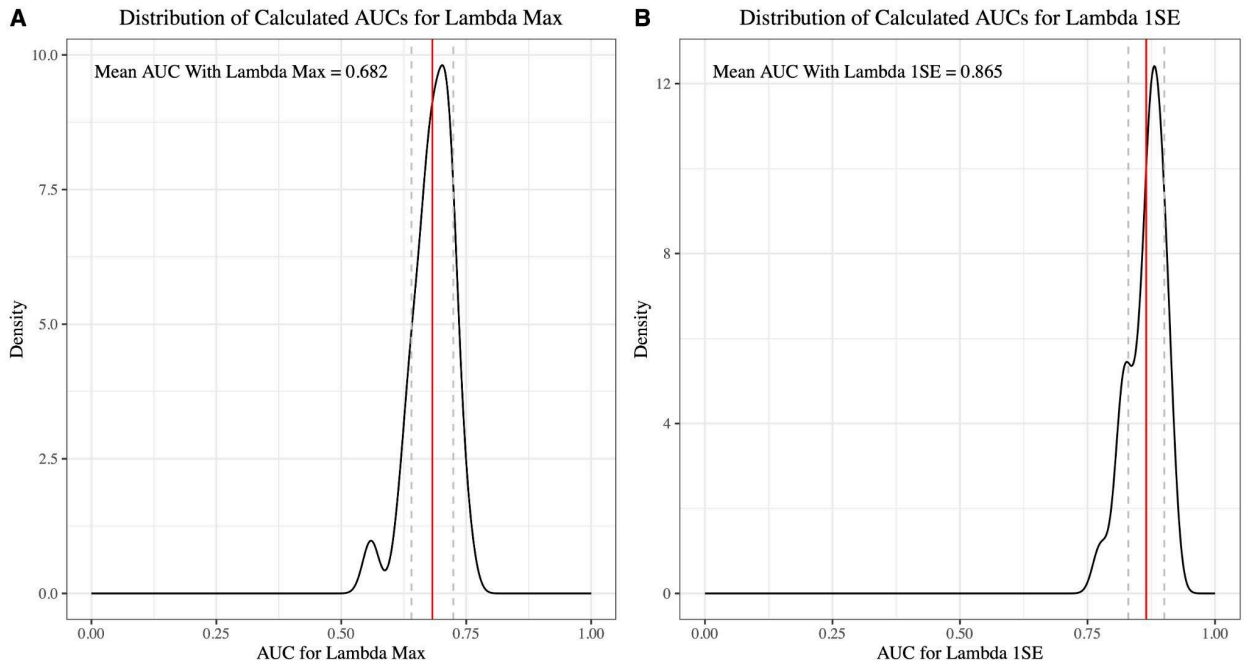

Figure 5 Performance of LCL framework on colorectal cancer (CRC) case study dataset using λ_{max} and λ_{1SE} (optimal regularization parameter derived through inner-cross validation as described in Methods). The red line marks the mean calculated AUC, and the dotted lines designate the values one standard deviation above and below the mean AUC.

As shown in Figure 7A, the median performance of the LCL framework regularized using λ_{1SE} is slightly greater than that of the ridgeEnsemble `DiCoVar` model as reported in Hinton & Mucha (2021) (AUCs of 0.801 and 0.80 respectively). Though rigorous statistical comparison is not possible due to the differences in

cross-validation parameters and seeds between Hinton & Mucha's model construction and that performed here, it is reasonable to claim that there is no significant difference in model performance between LCL and `DiCoVar` on the CRC case study data.

The second case study dataset explored the onset of necrotizing enterocolitis (NEC) in infants and was composed of 903 independent observations with $F = 135$. The performance of LCL with each of λ_{max} and λ_{1SE} is shown in Figure 6 and demonstrates that, in accordance with all previous analysis, the LCL framework regularized with λ_{1SE} (mean AUC = 0.865) performs much better than that with λ_{max} (mean AUC = 0.682).

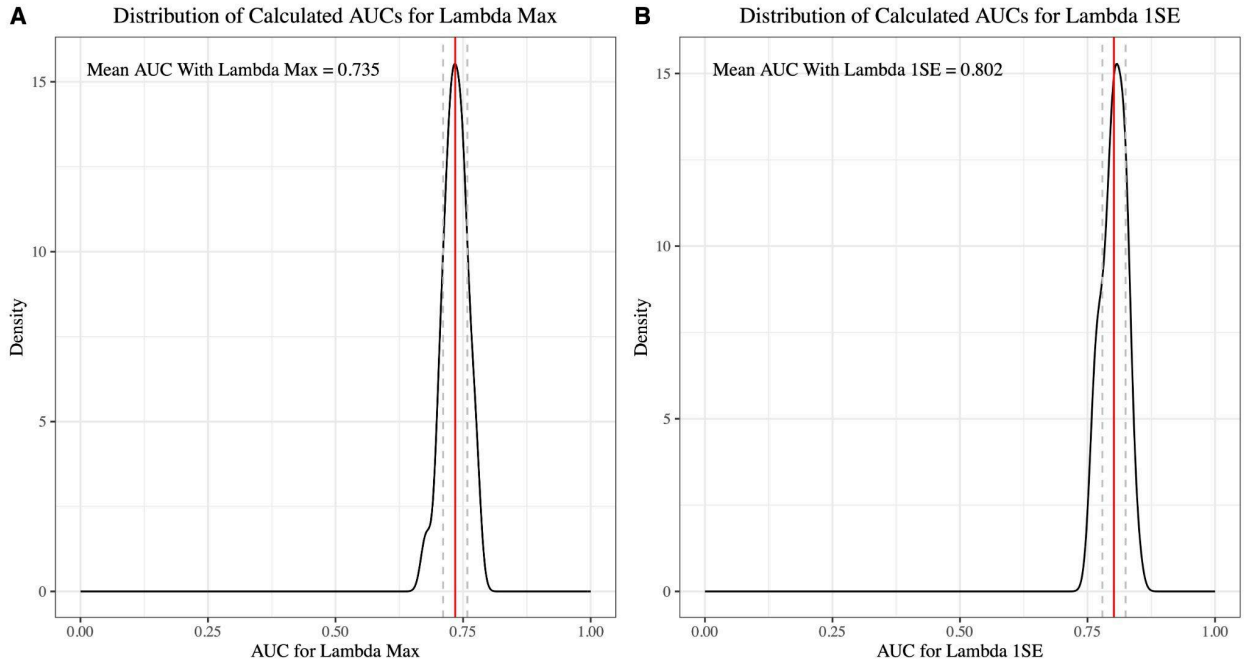

Figure 6 Performance of LCL framework on necrotizing enterocolitis (NEC) case study dataset using λ_{max} and λ_{1SE} (optimal regularization parameter derived through inner-cross validation as described in Methods). The red line marks the mean calculated AUC, and the dotted lines designate the values one standard deviation above and below the mean AUC.

As shown in Figure 7B, the LCL framework performs well above the ridgeRegression `DiCoVar` framework as reported in Hinton & Mucha (2021). While the median AUC of all models constructed for the NEC dataset using `DiCoVar` is 0.80, that of LCL is 0.873. Again, a rigorous statistical comparison of results is not possible due to the differences in cross-validation parameters and seeds between Hinton & Mucha's model construction and that performed here. However, it is reasonable to state that the LCL framework performs substantially better than the ridgeRegression `DiCoVar` framework on NEC case study data.

Overall, analysis of the performance of the LCL framework relative to that of `DiCoVar` in these two case studies reveals that for robust datasets with large *n*, LCL performs at a level comparable to or above `DiCoVar` depending upon the model used within `DiCoVar` (i.e. ridge regression versus ensemble learning) and the nature of the dataset.

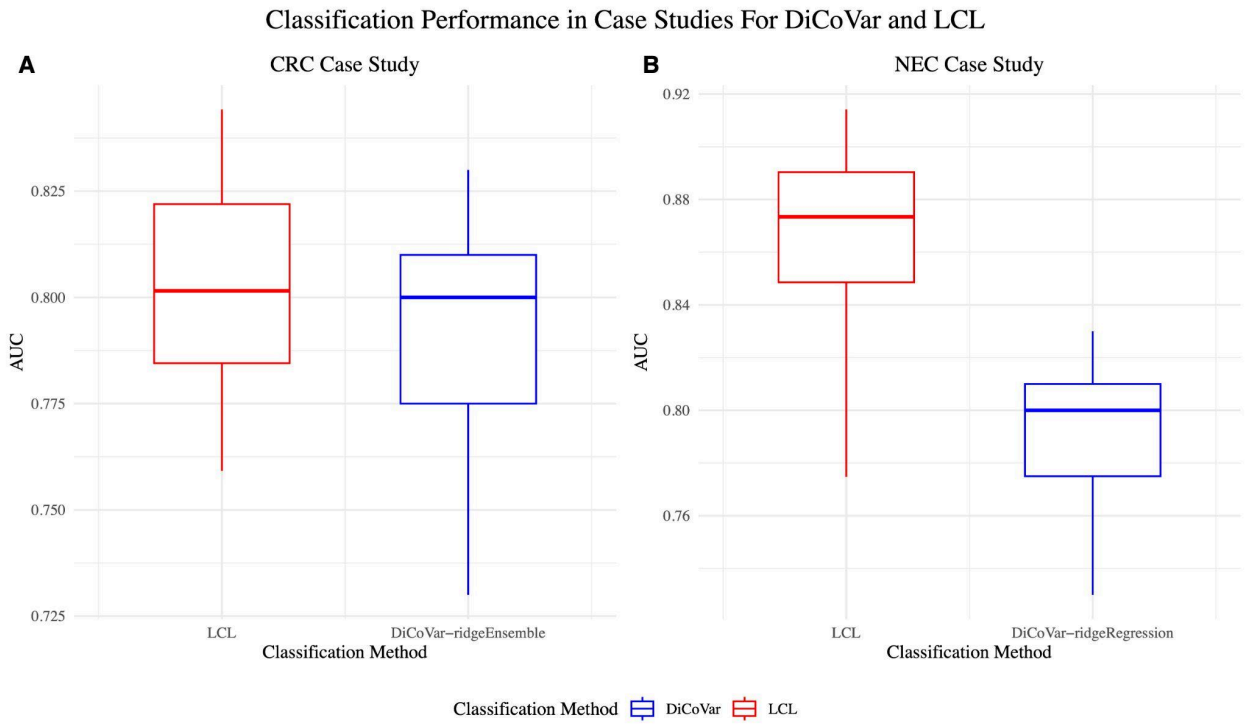

Figure 7 Each pair of boxplots shows the performance, measured using the AUC, of the LCL framework using λ_{1SE} (red) and the `DiCoVar` framework (blue). (A) shows these results for the CRC case study, and compares the LCL framework performance with the performance of the ridgeEnsemble version of the `DiCoVar` framework. This reveals comparable performance (almost exactly the same median) on the CRC dataset between the LCL framework and the `DiCoVar` framework incorporating ensemble learning. (B) reveals that the LCL framework substantially outperforms the ridgeRegression version of the `DiCoVar` framework (which does *not* use ensemble learning) on the NEC case study dataset.

Dataset Name	Number of Features (F)	Number of Observations (n)	F/n
CRC Case Study	934	1395	0.670
NEC Case Study	135	903	0.150

Table 2 This table includes the number of features F, the number of observations n, and the ratio between the number of features and number of observations (F/n) for the two case study datasets examined in Figures 5, 6 and 7.

Discussion

Analysis of Experimental Data Results

Performance of LCL and DiCoVar on Publicly Available Datasets

Because the DiCoVar framework used to develop the models used for classification for the publicly available datasets analyzed above uses ensemble learning, we expect that this framework will, on average, perform better than the LCL framework. The usage of ensemble learning methodologies mitigates the limitations of creating a singular model and can reduce bias and variance by combining multiple models, so it is only logical that the performance of an ensemble learning model like DiCoVar would exceed that of a single estimator like LCL.

With the exception of the CDI and HIV datasets (Figures 4D and 4G), this expected outcome occurs – though the LCL framework generally produces a subjectively 'good' mean AUC, the mean AUC of DiCoVar models is higher. For the CDI dataset, however, LCL does *much* worse than DiCoVar (mean AUCs of 0.876 and 0.980 respectively). In contrast, the HIV dataset is the sole dataset for which the mean LCL performance exceeds the mean DiCoVar performance (mean AUCs of 0.805 and 0.710 respectively).

What characteristics of the CDI and HIV dataset, given the methods of DiCoVar and the results of our synthetic data analysis, might produce such results? (Note: these properties are listed in Table 1). First, let's consider the dimensionality of the CDI and HIV datasets. The CDI dataset has $n = 336$ and $F = 259$; unlike some of the other publicly available datasets examined there are more observations than features. The same is true of the HIV dataset, which contains 155 observations of 60 features ($n = 155$ and $F = 60$). Notably, however, the ratio between F and n is greater in the CDI dataset ($F/n = 0.771$) than in the HIV dataset ($F/n = 0.387$). This preliminary comparison suggests that when compared to the performance of DiCoVar framework, the LCL framework does relatively better with datasets with smaller F/n.

However, other analyses of publicly available data do not support this conclusion. If it were strictly the case that LCL performs much better than DiCoVar when F/n is smaller, then we would expect the mean AUC from the LCL model to be significantly greater than that of DiCoVar for the Crohn's disease dataset, which has the smallest F/n ratio (0.0492) (see Table 1). This is not the case – while both models perform relatively well, the mean AUC generated by DiCoVar exceeds that of LCL by 0.041. Similarly, if it were strictly the case that LCL performs much worse than DiCoVar when F/n is larger, then we would expect the mean AUC from the LCL model to be significantly smaller than that of DiCoVar for the Zeller CRC dataset, which has the largest F/n ratio at 4.179 (many more features than observations) (Table 1). While LCL certainly does not perform as well (mean AUC = 0.76 while the DiCoVar mean AUC = 0.84), the magnitude of difference between LCL and DiCoVar performance is smaller than that of the CDI dataset.

Therefore, it is reasonable to conclude that it is at least not *only* the F/n ratio that is responsible for these seemingly aberrant results from the HIV and CDI publicly available datasets. From the results of the synthetic data analysis in which the signal-to-noise ratio of features X1 and X2 is manipulated, we also see that the performance of the LCL model is, as expected, very sensitive to the SNR (Figure 4C). While it makes sense

that a higher SNR would also adversely affect the performance of the DiCoVar model, it is possible that the LCL model displays higher sensitivity. If the SNR of selected features in the HIV dataset is significantly higher than that of the CDI dataset, this may then explain why LCL performs much worse than DiCoVar on the CDI dataset but slightly better on the HIV dataset.

Performance of LCL and DiCoVar on Case Study Datasets

As displayed in Figure 7, the LCL framework performs almost equivalently to DiCoVar on the CRC dataset but exceeds DiCoVar performance on the NEC dataset by a substantial margin (mean AUC from the LCL model is 0.073 greater than that of the DiCoVar model). So, why does LCL exhibit such better performance relative to DiCoVar on the NEC data compared with the CRC data?

Though it is possible that differences in dataset properties between the NEC and CRC case studies may contribute to this difference, the primary reason lies in the version of DiCoVar used to model the NEC data versus the CRC data. The CRC data is modeled with ridgeEnsemble – an ensemble learning framework – while NEC is modeled with ridgeRegression – a single estimator.

We inherently expect an ensemble learning model to outperform a single regression model. In general, ensemble learning reduces variance – that is, the model's sensitivity to specific training data – and therefore overfitting, and by combining models exploits specific models' strengths while mitigating individual weaknesses. In the specific case of the ridgeEnsemble DiCoVar model, individual linear regression models regularized with an L2 penalty are fit to PLR-transformed data, and these base models are then combined to improve predictive performance. Thus, it is reasonable to assume that a DiCoVar model with ridgeEnsemble constructed for the NEC dataset would exhibit better performance than that displayed in Figure 7. However, it remains unclear whether such a model would perform at a level below, comparable to, or above the LCL framework.

In the NEC DiCoVar analysis using a single ridge regression model, the model has access to the same feature space as the LCL analysis on this dataset because ensemble learning is *not* employed in either framework. Therefore, we would expect to see comparable performance between these two models. However, the relatively better performance of the LCL framework suggests that this is indeed a better model than DiCoVar.

Using the same line of reasoning, we would expect the DiCoVar ridgeEnsemble analysis of the CRC data to significantly outperform the LCL model constructed on the same data. The ensemble space accessed by the ridgeEnsemble version of DiCoVar provides this framework with much more freedom to fit an accurate, generalizable model. However, the comparable performance of DiCoVar when it *does* have access to this expanded feature space and LCL when it *does not* further demonstrates the superior performance of the LCL framework.

Cross-Dataset Comparison of Relative Performance of LCL and DiCoVar

For all but one experiment using publicly available data, the DiCoVar framework performs better than the LCL framework as demonstrated by a higher mean AUC. However, this is not the case in the CRC and NEC case studies, in which LCL performs comparably or better than DiCoVar depending upon the model used within DiCoVar (ridge regression versus ensemble learning). Per the discussion in the preceding section, it does not make sense to compare the relative performance of DiCoVar and LCL on the NEC case study with the

relative performance of `DiCoVar` and LCL on the publicly available datasets because the DiCoVar model used by Hinton & Mucha (2021) in the NEC case study is a ridge regression framework rather than an ensemble learning framework as is used in all of the publicly available dataset analyses. As such, this discussion will primarily focus on the difference between the relative performance of `DiCoVar` and LCL on the CRC case study data and the publicly available datasets.

First, we note that the CRC case study dataset contains 1395 observations of 934 features, giving it an F/n ratio of 0.670 (Table 2). However, examination of relative performance of `DiCoVar` and LCL on the CRC data versus on publicly available datasets with similar F/n confirms the conclusion in the preceding section that the feature number to sample size ratio does not (or at least does not *solely*) affect `DiCoVar` vs. LCL performance comparison. In fact, the CDI dataset – the publicly available dataset for which the LCL framework performs *most* below `DiCoVar` – is the one with the *most similar* F/n ratio (0.771) to that of the CRC case study dataset.

Next, we consider how class distribution may impact the relative performance of `DiCoVar` and LCL. For the CRC case study data, $p = 0.497$, representing a remarkably equal class distribution (unlike many of the publicly available datasets that, when examined, are better modeled by `DiCoVar` than LCL). As shown through our preceding analysis of the synthetically generated data, the performance of the LCL framework regularized with λ_{max} is somewhat dependent upon class distribution (Figure 4D). That is, LCL performance on synthetic data decreases on average when p is close to 0 or 1 (very few or majority control observations) while the framework performs optimally when $p = 0.5$. While this effect is isolated to LCL regularized with λ_{max}, which notably is *not* the regularization parameter used in any analysis referenced here, this demonstrates that the framework overall does demonstrate some sensitivity to class distribution, perhaps above and beyond `DiCoVar`. The synthetic data is largely dissimilar from the high-dimensional data analyzed here (in the experiments examining the effect of manipulating p, the parameters are $F = 5$, $n = 200$ and there are only two relevant features). Therefore, the conclusion reached in the simulated data experiments that the performance of LCL regularized with λ_{1SE} does not vary with p may not generalize to higher-dimensional cases. Indeed, it makes logical sense that constructing a model with data equally distributed between classes would result in better performance, and this inherent difference in the CRC dataset may be at least partially responsible for the equivalent performance of LCL and `DiCoVar` on this data.

Additionally, it is possible that signal-to-noise ratio differences between the CRC case study data and the publicly available datasets may account for the relative improvement of the LCL framework relative to `DiCoVar`. As shown in Figure 4C and described in the preceding section, the LCL framework is highly sensitive to SNR – perhaps more so than `DiCoVar`. If this is the case, and if the CRC case study data includes relatively less noise in predictive features (smaller σ^2) than many of the publicly available datasets, then it would make sense that we would see more comparable performance of the LCL and `DiCoVar` frameworks on the CRC case study data than on the publicly available datasets with higher SNRs.

While we also initially considered sample size (n) as a possible contributor to relatively better performance of LCL when compared with `DiCoVar` on the CRC case study data, it is unlikely that sample size alone is a primary cause for the effects seen. First, this performance improvement of LCL relative to `DiCoVar` is *not* seen in models constructed for the Crohn's disease publicly available dataset, which is comparable in size to

the CRC case study dataset (Crohn's n = 975). Second, analysis of synthetic data with varied n demonstrated that, at least for data with few predictive and noisy features, the number of observations has no statistically significant impact on LCL AUC (Figure 4A). Though this synthetic data is not necessarily representative of higher-dimensional metabolomic or metagenomic data with more predictive features as described above, this finding combined with the stark performance difference between the LCL and `DiCoVar` frameworks on the Crohn's disease data suggests that sample size does not majorly account for differences in relative performance.

Despite this, it is possible that the aforementioned sensitivity to SNR in the LCL framework relative to `DiCoVar` is more pronounced when sample size is small. It makes sense that the confluence of a small training set and observations with large variance would dually increase the risk of overfitting in LCL models, meaning that LCL may be more likely to underperform relative to `DiCoVar` in cases where the SNR is large and the number of samples is small. We know that the publicly available datasets are generally much smaller than the CRC case study dataset, and it is possible that they may have (on average) a higher SNR. If this is the case, and it is true that LCL's sensitivity to signal to noise ratio is more prominent in datasets with fewer samples, then this would explain the improved performance of LCL relative to `DiCoVar` on the CRC dataset.

Computational Requirements of LCL Modeling

Relative to the computational demands of `DiCoVar`, the LCL framework is much more efficient. All experiments for synthetic, publicly available, and case study data used 25 outer loops and 20 inner loops, with each inner loop run 102 times (one for each value of λ tested). This represents a total of 51,000 individual models generated in each experiment. Each experiment is run on 14 cores and the inner loop function institutes parallel computing using the `Parallel` R package.

With these parameters and parallelization used, each synthetic dataset requires approximately 30-60 minutes to run, each publicly available dataset requires approximately 40 minutes to run, and the case studies require an average of 4.5 hours to run. Figure 8, which displays runtimes to create full models on synthetic datasets with varied number of noisy features (F) and varied number of observations (n), reveals that individual runtimes (which are a proxy for computational expense) depend upon F and n.

In the synthetic data used in Figure 8, we see that the runtime increases by only 3.820 minutes (from 38.566 minutes to 40.860 minutes) when the number of observations increases from 50 to 500 (with number of features held constant at 5), representing a 9.91% increase in runtime with a tenfold increase in n.

Additionally, we see that the runtime increases from 40.860 minutes to 56.369 minutes when the number of noisy features increases from 1 to 250, a difference of 15.509 minutes. With a tenfold increase in F from 5 to 50, we see an increase in runtime from 39.740 minutes to 43.623 minutes, which is a 9.77% increase. With a tenfold increase in F from 10 to 100, we see an increase in runtime from 41.500 minutes to 44.639 minutes, which is a 7.56% increase.

Because similar magnitudes of change in n and F result in similar % change in runtime, we can conclude that runtime (which can be interpreted as a proxy for computational expense) depends approximately equally on the number of observations and the number of noisy features.

Importantly, it is likely that LCL model runtime depends linearly on F and n when either/both of these values are high. With the synthetic data examined in Figure 8, we are looking at a relatively small system with significant "startup costs" where F and n do not appear to be directly proportional to runtime. However, they are likely proportional to runtime at their limits, which the synthetic data experiments do not approach.

Though Hinton & Mucha (2021) do not directly cite the computational requirements of the `DiCoVar` model in their writeup, it is known that the computational resources used to construct a `DiCoVar` model in comparable settings typically involves computations taking days on clusters with 12–48 cores, far exceeding those required by the LCL framework. The significantly shorter runtimes for LCL compared to `DiCoVar` make sense given that `DiCoVar` is computing a much more complex problem in the $\binom{F}{2}$–dimensional feature space.

Because of the quadratic increase in the number of features seen in `DiCoVar` as the result of the pairwise log ratio transformation, we expect that, at the limit, runtime increases quadratically with the number of features. However, it is clear from Figure 8 that runtime is *not* increasing quadratically with the number of features (which makes sense because the LCL model works exclusively in a F-dimensional feature space and therefore increases linearly as described previously), which is further evidence for the computational improvements of LCL over `DiCoVar`.

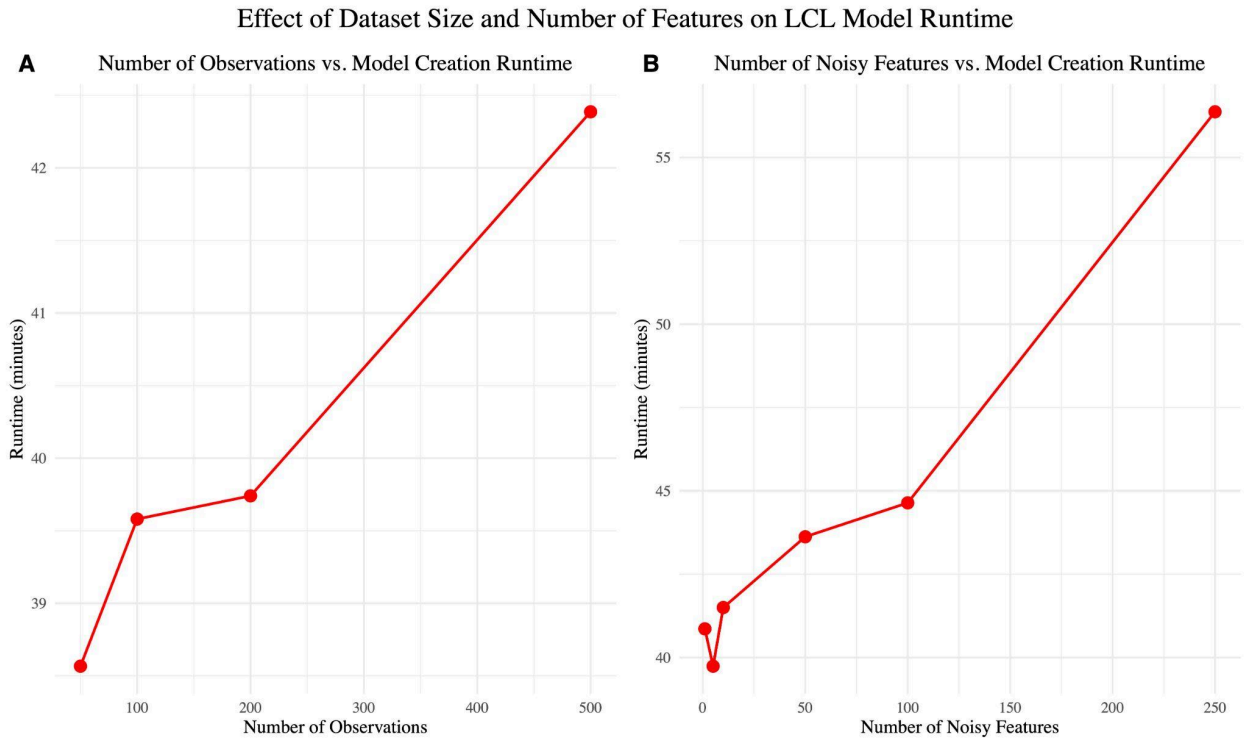

Figure 8 Time to compute LCL model on synthetic data with varied dataset size and number of noisy features. (A) shows how the runtime varies with the number of observations n on synthetic data generated with all other parameters held constant ($J = 1$, $p = 0.5$, $F = 5$). This figure reveals that runtime increases slightly as n increases. (B) shows how the runtime varies with the number of noisy features F on synthetic data generated with all other parameters held constant ($J = 1$, $n = 200$, $p = 0.5$). Overall, the figure shows that runtime is highly positively correlated with F.

Examining Model Construction Decisions

Throughout the development of the LCL framework, numerous design decisions were made surrounding methods of imputation, cross-validation, parameter selection, and writing code. While each choice occurred intentionally, it is likely that these decisions do affect LCL model performance – though the nature and magnitude of this impact is unknown at this time. Additional research into alternate design choices is needed to fully understand how different methods may improve model efficacy.

First, in preliminary analyses of both synthetic and publicly available data, we found that CVXR using the MOSEK optimization solver failed seemingly at random. This did not occur often (further exploration revealed that this happened at most twice in a model with 25 outer splits and 20 inner splits), but we were unable to determine any definitive cause for this failure. We considered the possibility that data with very unequal class distributions might cause this error, as there could be training and splits that contained none of a certain class of observation. However, upon further examination of the R `caret` documentation (the package we were using to create random train/test splits), we realized that this algorithm would not permit single-class training or testing splits for sufficiently large data (at least three observations of each class). Since all datasets met this criteria, this was not the reason for the CVXR failure.

Notably, these failures were uncommon – in many datasets, the solver never failed. For the few in which it did, it occurred in a maximum of two of the 25 outer loops in a maximum of two of the inner splits and for at most one value of λ in the given inner loop. This is equivalent to a maximum of four failures of 51,000 models constructed. Therefore, we chose to implement R's `tryCatch` function to simply skip a run through the inner loop and calculate the mean and standard error AUC for the given λ without data from this iteration. Thus, further exploration is needed to determine the reason for this solver failure as well as how the omission of these iterations may impact LCL model results.

Next, alternative decisions could have been made surrounding the zero imputation. As described in the Methods section, zeros in metabolic and metagenomic input data represent the absence of a given substance or bacteria/virus/fungus – but these observations become $-\infty$ when log-transformed. In the LCL framework, these values are imputed by replacing them with the smallest nonzero value in the (already log-transformed) training set. Zero values in the outer loop test set are *also* replaced by the smallest nonzero value in the training set. This is effectively the same as using the smallest nonzero value as a 'limit of detection' – that is, any zero obtained might be because it was below this limit, so it makes reasonable sense to replace it with this value.

However, there are alternative methods of imputing zeros – for example, we could have chosen an arbitrarily small number before any cross-validation occurred and replaced all zeros with this value. Or, zeros in the test set could have been replaced with the smallest value in the test set split rather than the training split. While this is an important consideration when working with older metabolomic datasets, note that many newer mass spectrometry machines produce output data without zeros. They give estimates of the limit of detection for a given substance (based upon the machine's capabilities and the properties of the material being analyzed) and fill in this value in every place that it did not show up in the reading. This eliminates the need for zero imputation, since the log of this data can be computed as-is without producing undefined values.

Finally, we made the choice to use a train/test split of 0.8/0.2 for both the inner and outer cross-validation loops. This was informed by preliminary analyses on the publicly available datasets. When the proportion of data included in the training set for the inner cross-validation was larger (0.9, 0.95), the smaller test set prevented the algorithm from effectively finding the best λ since accurate performance estimates could not be obtained from the small testing set – which sometimes included only one or two observations of a given class for data with uneven class distribution. When the proportion of data included in the training set was smaller (0.6, 0.7), there was not sufficient data to train an effective model and it struggled with overfitting.

Motivated by the understanding that insufficient training data yields a poor model but insufficient test data can give inaccurate estimates of model performance, we also explored the use of leave-one-one cross validation in the outer loop. We hypothesized that it would provide the inner loop hyperparameter tuning with the maximal amount of data with which to find the best λ and perhaps yield better results. Using the HIV dataset to test this, we found that this method was ineffective – though it yielded approximately 80% accuracy, this was merely because it was assigning all observations to the class constituting 80% of the data.

Though these preliminary results reveal the importance of train/test split proportions in producing an effective model, no rigorous testing on synthetic data nor on different datasets with different characteristics was performed. Additional rigorous experimentation is needed to determine how nuanced changes in train/test splits in both the inner and outer loops would affect LCL performance and thus to refine the values used in the LCL model.

Next Steps – Ensemble Learning and LCL

Based upon the success of the ensemble learning methods employed in the `DiCoVar` framework, the logical next step in creating a more powerful LCL framework to improve its predictive performance is introducing ensemble learning. While there are many possible ways to do this, a first step would be to use each unique log contrast function learned in the nested cross validation algorithm in an ensemble learning framework such as a random forest. A random forest classifier builds a large collection of decision trees, each trained on different random subsets of the data and/or its features, and combines the prediction of each tree to produce a more robust classification. The combination of multiple log contrasts (many candidate models) in an ensemble learning framework allows for performance above and beyond each model's performance on its own.

Another possible extension of the existing LCL model to ensemble learning is to consider the space of all possible pairwise log ratios between the features extracted in the preliminary model. For datasets in which many predictive features are extracted, this would negate the computational efficiency of LCL relative to `DiCoVar`. However, for datasets with fewer extracted features – such as the NEC case study data (see Figure 5 in Hinton & Mucha, 2021) – using all possible pairwise log ratios between these extracted features may yield a computationally reasonable method of refining the LCL results. These log ratios can then be considered in an ensemble learning model, such as a random forest as described above.

While this study represents a preliminary comparison of LCL with existing methods and additional rigorous experimentation is required to how the results shown here generalize to other datasets, our initial findings suggest that LCL – especially once integrated with ensemble learning methods – has the potential to

revolutionize standard practice in analysis of metagenomic and metabolomic data for biomarker discovery and disease classification.

Bibliography

Aitchison, J. (1982). The Statistical Analysis of Compositional Data. *Journal of the Royal Statistical Society: Series B (Methodological)*, *44*(2), 139–160. https://doi.org/10.1111/j.2517-6161.1982.tb01195.x

Aitchison, J., & Bacon-Shone, J. (1984). Log Contrast Models for Experiments with Mixtures. *Biometrika*, *71*(2), 323–330. https://doi.org/10.2307/2336249

Bates, S., & Tibshirani, R. (2019). Log-Ratio Lasso: Scalable, Sparse Estimation for Log-Ratio Models. Biometrics, 75(2), 613–624. https://doi.org/10.1111/biom.12995

Friedman, J. H., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, *33*, 1–22. https://doi.org/10.18637/jss.v033.i01

Fu, A., Narasimhan, B., & Boyd, S. (2020). CVXR: An R Package for Disciplined Convex Optimization. *Journal of Statistical Software*, *94*, 1–34. https://doi.org/10.18637/jss.v094.i14

Greenacre, M. (2019). Variable Selection in Compositional Data Analysis Using Pairwise Logratios. *Mathematical Geosciences*, *51*(5), 649–682. https://doi.org/10.1007/s11004-018-9754-x

Greenacre, M., Grunsky, E., Bacon-Shone, J., Erb, I., & Quinn, T. (2023). Aitchison's Compositional Data Analysis 40 Years on: A Reappraisal. Statistical Science, 38(3), 386–410. https://doi.org/10.1214/22-STS880

Hastie, T., Qian, J., & Tay, K. (2023, March 23). *An Introduction to glmnet*. glmnet. https://glmnet.stanford.edu/articles/glmnet.html

Hinton, A.L., & Mucha, P.J. (2021). Differential Compositional Variation Feature Selection: A Machine Learning Framework with Log Ratios for Compositional Metagenomic Data. BioRxiv. https://doi.org/10.1101/2021.12.08.471758

Hron, K., Filzmoser, P., Donevska, S., & Fišerová, E. (2013). Covariance-Based Variable Selection for Compositional Data. *Mathematical Geosciences*, *45*(4), 487–498. https://doi.org/10.1007/s11004-013-9450-9

Kuhn, Max (2008). Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 28(5), 1–26. https://doi.org/10.18637/jss.v028.i05

Lin, W., Shi, P., Feng, R., & Li, H. (2014). Variable selection in regression with compositional covariates. Biometrika, 101(4), 785–797. https://doi.org/10.1093/biomet/asu031

Lu, J., Shi, P., & Li, H. (2019). Generalized linear models with linear constraints for microbiome compositional data. *Biometrics*, *75*(1), 235–244. https://doi.org/10.1111/biom.12956

MOSEK ApS. (2025). *Mosek RMOSEK package 11.0.18*. MOSEK. https://docs.mosek.com/latest/rmosek/index.html

Pasolli, E., Schiffer, L., Manghi, P., Renson, A., Obenchain, V., Truong, D., Beghini, F., Malik, F., Ramos, M., Dowd, J., Huttenhower, C., Morgan, M., Segata, N., Waldron, L. (2017). "Accessible, curated metagenomic data through ExperimentHub." *Nat. Methods*, 14(11), 1023–1024. ISSN 1548-7091, 1548-7105, https://doi.org/10.1038/nmeth.4468

Rivera-Pinto, J., Egozcue, J. J., Pawlowsky-Glahn, V., Paredes, R., Noguera-Julian, M., & Calle, M. L. (2018). Balances: A New Perspective for Microbiome Analysis. *mSystems*, *3*(4), e00053-18.

https://doi.org/10.1128/mSystems.00053-18

Rubel, M. A., Abbas, A., Taylor, L. J., Connell, A., Tanes, C., Bittinger, K., Ndze, V. N., Fonsah, J. Y., Ngwang, E., Essiane, A., Fokunang, C., Njamnshi, A. K., Bushman, F. D., & Tishkoff, S. A. (2020). Lifestyle and the presence of helminths is associated with gut microbiome composition in Cameroonians. *Genome Biology*, *21*(1), 122. https://doi.org/10.1186/s13059-020-02020-4

Schubert, A. M., Rogers, M. A. M., Ring, C., Mogle, J., Petrosino, J. P., Young, V. B., Aronoff, D. M., & Schloss, P. D. (2014). Microbiome Data Distinguish Patients with Clostridium difficile Infection and Non-C. difficile-Associated Diarrhea from Healthy Controls. *mBio*, *5*(3), e01021-14. https://doi.org/10.1128/mBio.01021-14

Sharpton, S. R., Yong, G. J. M., Terrault, N. A., & Lynch, S. V. (2018). Gut Microbial Metabolism and Nonalcoholic Fatty Liver Disease. *Hepatology Communications*, *3*(1), 29–43. https://doi.org/10.1002/hep4.1284

Shi, P., Zhang, A., & Li, H. (2016). Regression analysis for microbiome compositional data. The Annals of Applied Statistics, 10(2), 1019–1040.

Susin, A., Wang, Y., Lê Cao, K.-A., & Calle, M. L. (2020). Variable selection in microbiome compositional data analysis. *NAR Genomics and Bioinformatics*, *2*(2), lqaa029. https://doi.org/10.1093/nargab/lqaa029

Zeller, G., Tap, J., Voigt, A. Y., Sunagawa, S., Kultima, J. R., Costea, P. I., Amiot, A., Böhm, J., Brunetti, F., Habermann, N., Hercog, R., Koch, M., Luciani, A., Mende, D. R., Schneider, M. A., Schrotz-King, P., Tournigand, C., Tran Van Nhieu, J., Yamada, T., … Bork, P. (2014). Potential of fecal microbiota for early-stage detection of colorectal cancer. *Molecular Systems Biology*, *10*(11), 766. https://doi.org/10.15252/msb.20145645

Zhu, F., Ju, Y., Wang, W., Wang, Q., Guo, R., Ma, Q., Sun, Q., Fan, Y., Xie, Y., Yang, Z., Jie, Z., Zhao, B., Xiao, L., Yang, L., Zhang, T., Feng, J., Guo, L., He, X., Chen, Y., … Ma, X. (2020). Metagenome-wide association of gut microbiome features for schizophrenia. *Nature Communications*, *11*(1), 1612. https://doi.org/10.1038/s41467-020-15457-9