BATCH CORRECTION OF TAXONOMIC DATA OF THE HUMAN GUT MICROBIOME FOR GENERALIZATION OF CASE-CONTROL CLASSIFICATION

By

Eric Antonius van der Toorn

in partial fulfilment of the requirements for the degree of

Master of Science

in Computer Science

at the Delft University of Technology,

to be defended publicly on Wednesday June 22nd, 2022 at 11:00 AM.

Supervisor:	Dr. T.E.P.M.F. Abeel	TU Delft, supervisor, committee chair
Thesis committee:	Dr. D. G. Weissbrodt	TU Delft
	Dr. Z. Erkin	TU Delft
	C. Peng	TU Delft, daily supervisor

Page intentionally left blank.

Preface

This report is the result of my thesis project to obtain the Master of Science degree for the Bioinformatics specialization of the Artificial Intelligence track in Computer Science, and I would like to thank the reader for their attention and interest. This project started from a discussion with Thomas at the end of last year, molding during my literature survey into something I believe neither of us had expected. It was really a first occasion for me in which I had to face the freedom of creating and directing my own project, which was both thrilling with the possibilities of where it could go, as well as heavy in terms of responsibility.

This document has two parts; firstly, a scientific paper written with the goal of publication, demonstrating the main results as well as the methodology used for obtaining these. This includes a supplementary section with figures and details related to and referenced by the paper. Secondly, I have included additional chapters that describe additional work that did not make it into the paper, including the development of an autoencoder that showed inadequate performance as well as the READMEs of the codebase developed for the project.

My thanks go first of all to Thomas Abeel, my supervisor, for his guidance throughout my project, from pushing me to do something I wanted to explore while giving critical feedback at every stage such that I could improve my results, and know where to go next. Chengyao Peng, my daily supervisor, was always available for questions and discussions, engaging me with my thesis through her constant positivity, and I am deeply thankful for her support. Furthermore, the AbeelLab and the rest of the PRB group made the work environment so much more fun and engaging that I ended up working in the offices, enjoying my coffee breaks with everyone that wanted to join. Finally, the care I got from the people close to me is amazing and overwhelming, marking me a happy guy.

I hope you enjoy,

Eric Antonius van der Toorn

Delft, June 2022

Abstract

Next-Generation Sequencing (NGS) has made it possible to perform metagenomic sequencing of environmental microbiome samples. Colorectal cancer (CRC) benefits from early detection, and many studies find correlations between disease presence and abundance of species in samples of the microbiome. However, these studies are hard to reproduce and even harder to build diagnostic tools from, and one of the major factors for this is the inherent bias in the datasets that were collected, the so-called batch effect.

To investigate the extent to which batch effect impacts the generalization of binary classifiers, we performed a benchmark of eleven batch correctors: four existing tools, three transformations and three encoders, assessing the subsequent performance of seven supervised binary classifiers using a leave-one-dataset-out (LODO) validation method. In addition, batch effect was measured through both visual (tSNE) and numeric (linear models) methods before and after applying each of the correctors, and the performance at different dataset counts was measured.

Batch effect was shown to be present in the shotgun metagenomic data, being reduced by some correction tools while being strengthened by others. Evaluations using AUROC showed that combining datasets without correction improved generalization, even at an equivalent number of samples. When combining batch correctors and different classifiers, the performance over the baseline did not improve significantly. Contrary to its popularity as batch corrector, the performance significantly worsened when using ComBat before training each of the binary classifiers.

Thus, even though batch correctors reduce batch effect within our taxonomic count data, they do not significantly improve classification performance when generalizing to separate datasets. We can thus advise against focusing on choosing a batch corrector when building tools for predicting diagnosis of CRC and instead aiming to improve the pool of datasets to learn from.

The code for reproducing the results and figures in this work have been made available at https://github.com/AbeelLab/ngs-batch-evaluation

1. Introduction

1.1. Machine learning for gut microbiome-based diagnostics

Predicting the diagnosis of a patient based on a sample of their gut microbiome is a challenging task due to the highly complex nature of the microbiome. The microbiome is highly variant, both in its composition and its function, both between patients, latitudinally, and over time within the same patient, longitudinally. Factors contributing to this difference include demographic and biological ones like gender, age and diet [1–3]. The increasing number of available microbiome studies and the increased interest in machine learning [4,5] have sparked more interest in the field as one of the first applications of personalized medicine as a combination of both [4].

One disease that especially affects the microbiome is colorectal cancer (CRC), the second most common cancer in women and third in men [6,7]. It is a disease that silently develops over the course of multiple years or decades, usually only showing symptoms after it has metastasized. If found early on, the disease is highly curable and can be removed entirely. Early detection of the disease can thus potentially save many lives [8].

Creating a diagnostic test for CRC can be done by looking at individual biomarkers for the disease and detecting their presence or by combining information from many features with machine learning models. While potentially losing explainability, these models are able to capture more complex patterns than singular biomarkers, which the complex nature of the microbiome may require.

Many machine learning approaches have been attempted, from the simplest logistic regression to highly complex deep learning models, showing varying amounts of success [4]. Unfortunately, results from studies of the microbiome are notoriously hard to reproduce, with independent replications of studies failing to obtain similar results [9]. This is both because of common pitfalls in the creation and evaluation of models as well as properties of the datasets used for microbiome data.

The problems facing datasets come both from how their data is obtained as well as their nature. Metagenomic datasets of the human microbiome are frequently small for case-control datasets, with a sample size of between 50 and 100 patients [4,10,11], with the exception of some large collaborations like the Human Microbiome Project (HMP) [12]. The number of features that can be obtained from the microbiome massively exceeds this number, rendering feature selection as one of the first steps in any analysis [4]. Additionally, the most frequently used features, count data of either the genes or the species, follow a non-normal distribution, making them less suited to many typical analyses [4,9].

To tackle the issue of small sample sizes and build more robust machine learning models, many studies have come to use multiple datasets, both for learning and for cross-validation [13–15]. Combining datasets from different studies comes with its own issues, however, of which a major one is the batch effect. Batch effect, the bias between each dataset that can inadvertently confound the biological signals, is the topic of this work.

1.2. Batch effects: a common challenge facing the integrative machine learning analysis for microbiome data

1.2.1. Batch effect

Batch effect has several different definitions within the literature, with the most common one being that of technical sources of variation between datasets [16–18]. Specific to next-generation sequencing data, it has also come to include other sources of variation between datasets that are undesired and unaccounted for, including both biological factors like age and diet as well as computational factors like the software used to analyze the raw reads [9,19]. As it is difficult to distinguish between technical and non-technical effects, and datasets rarely note down the same covariates with the same level of accuracy, making it harder to aggregate between them, in this work we use the more inclusive definition, wherein batch effect refers to any variation between batches.

It has been established that batch effects are present in metagenomic data [16,17,20–23] and that these effects can correlate closely with and confound biological results as to cause their validity to be strongly doubted [16]. A number of studies validate their results on a different dataset [4,24,25], but only recently has the correction of batch effect been methods been of interest for case control classification [26].

1.2.2. Batch correction

Batch correction methods try to remove batch effects from the raw data so that it can be analyzed as if all data were from the same batch. These methods range from simple standardization of the data to more complicated deep learning networks, adapted from similar data types like microarray and RNASeq data or developed uniquely for metagenomic data. However, some of these methods rely on complete information about the covariates of the data in order to perform their correction or have other limitations, rendering only a subset useful for predictive diagnosis [16,17,21,22,26,27].

Evaluating the impact that batch correction methods have on downstream analysis is not extensively studied. The effect of batch correction on metagenomic microbiome data has not been extensively studied, and as different correction methods influence the distribution of the datasets differently, the same downstream analysis may not be as effective between them. As such, benchmarking requires exhaustively testing and tuning combinations of batch correctors with binary classifiers.

1.3. Contribution

In this work, we aim to achieve a thorough analysis of batch effects in real-world gut microbiome data sets for CRC patients. First, we demonstrate how batch effects affect the accumulated data set we collected. Next, we measure the effectiveness of integrating multiple batches towards generalization on unseen batches. Last, we provide an evaluation of the most commonly used batch correction algorithms within the field of metagenomics for removing batch effects. In the end, we focus on delineating the impact of these algorithms on the generalization of supervised case-control classification, by evaluating the classification performance on unseen batches. Through this, we hope to provide new insights and guidance for the future machine learning research making use of this kind of data, and aid the development of a predictive model.

2. Methodology

2.1. Colorectal cancer microbiome datasets

The analyses in this work were performed on a set of eleven colorectal cancer (CRC) gut microbiome datasets, obtained from the CuratedMetagenomicData database [28]. CuratedMetagenomicData is a large-scale human microbiome database that provides uniformly processed human whole-genome shotgun metagenomic datasets. The metadata of each dataset was manually curated by contributors to the project. To date, CuratedMetagenomicData contains 20,533 human microbiome samples from 90 publicly available studies [28].

Among the CRC datasets available, we selected those with relatively large total size and balanced disease and healthy samples, after filtering out all repeated longitudinal samples. These longitudinal samples were filtered out to prevent contamination between training and test sets. The full procedure for obtaining and filtering our datasets can be found in Supplementary section 6.3.1.

The metadata of the datasets used can be found in Table 6-1. After filtering, most of the selected datasets had between 50 and 130 samples available, with the exception of Yachida et al. which has more than 500 samples.

Each of the raw metagenomic datasets was processed using the MetaPhlan3 package to obtain a species-level resolution of the number of reads associated with a clade, using around 100,000 microbial genomes [13]. The features were then filtered to only select species, as this has been shown to allow for the most accurate classification from a single data type [4]. The resulting dataset format is shown below in Table 2-1, outlining the eventual dataset that was used for the modelling.

Batch Label	*Disease Label*	*Species 1*	*Species 2*	...	*Species 934*
FengQ_2015	control	3914	0	...	0
FengQ_2015	CRC	1709	0	...	0
FengQ_2015	control	73699	0	...	0
⋮	⋮	⋮	⋮	⋱	⋮
YuJ_2015	control	0	51343	...	0
YuJ_2015	CRC	687589	44302	...	0
YuJ_2015	CRC	275081	232314	...	0

TABLE 2-1 SAMPLE OF DATASET USED, SHOWING BATCH, LABEL, AND FEATURE INFORMATION. BATCH REPRESENTS ONE OF THE ELEVEN DATASETS USED. DISEASE LABEL INDICATES WHETHER THE SAMPLE WAS FROM A CASE OR CONTROL SAMPLE. RANDOM REPRESENTATIVE SUBSET OF SPECIES WAS CHOSEN FOR FEATURES, WITH SHOWN DATA FROM *BACTEROIDES STERCORIS*, *BILOPHILA WADSWORTHIA* AND *ACTINOBACULUM MASSILIENSE* FROM LEFT TO RIGHT RESPECTIVELY.

2.2. Batch effect evaluation

To fully evaluate the presence and strength of batch effects, we relied on a few evaluation methods from different aspects, including visualization batch effects by dimension reduction, correlation analysis, and silhouette score analysis. Through dimension reduction of microbiome datasets from different studies, we visualized whether the batch effects are strong enough to cluster the datasets present [29] while correlation analysis allowed us to quantify which microbial features are influenced by the batch. The silhouette score then showed an objective measure of how well the batches cluster.

2.2.1. Visualizing batch effects

For visualizing high dimensional microbiome data, we used the T-distributed Stochastic Neighbour Embedding (tSNE), a dimension reduction method developed by van der Maaten [29], which aims to minimize the Kullback-Leibler divergence between the actual distribution of points in the original dimensionality and the points in the projection in two dimensions. This dimension reduction method was used to group samples locally and avoids overlapping points closely, as it is non-linear and performs different transformations on different regions. tSNE was chosen over the also commonly used principal component analysis (PCA) as PCA is known to break down in high-dimension cases [30].

2.2.2. Features significantly influenced by covariates

As an alternative to the visual analysis, a correlation analysis determined the presence of batch effects on a per-feature basis. The individual features are considered as the output variable for a model with the batch and disease labels as categorical input variables. After fitting this model, a likelihood ratio test was used to determine whether having the batch and disease label present as variables explains more of the feature's variance than not [32]. To correct for multiple testing, Bonferroni correction was applied to the p-values obtained from the test by the number of features present in a dataset. Each batch corrector was seen as an independent experiment, and correction was only applied with regards to the number of features its output possessed. In addition to a simple linear model, negative binomial model was also used, as it fits the distribution of the unprocessed individual features more closely [31].

Rather than the coefficients determined from the fits, we calculated the number of significantly corrected features that had either the batch information as a significant factor, the disease label as a significant factor, or both. As some transformations change the number of correlated feature, which made a direct comparison unreasonable, we divided the number of corrected features by the total number of features present in each transformed dataset to obtain proportions.

We expected the proportion of features that have batch as a significant factor to fall after batch correction, while those with disease label as a factor would remain be equivalent or increase if the biological information became less confounded.

2.2.3. Silhouette Scores

The silhouette score, also known as the average silhouette width, allowed for quantitatively evaluating the effectiveness of a clustering on a per-sample basis. It is the average silhouette coefficient for all the samples present in the dataset [22]. The silhouette coefficient compares the intra-cluster to inter-cluster distances for a sample following

$$\boldsymbol{s_i} = \frac{\boldsymbol{b_i} - \boldsymbol{a_i}}{\mathbf{max}(\boldsymbol{a_i}, \boldsymbol{b_i})}$$

EQUATION 2-1 FORMULA FOR CALCULATING THE SILHOUETTE WIDTH OF A SINGLE POINT s_i

Here, a_i is the average intra-cluster distance, while b_i is the average distance to the nearest cluster that sample s_i is not part of. To calculate distance, we use the cosine similarity, one of the most typical distance metrics for sparse high-dimensional data [32]. The coefficient s_i is bounded between -1 and 1, where positive values indicate that the clustering is correctly assigned to a cluster while negative values mean that another cluster should have been assigned instead.

While usually used to evaluate the performance of a clustering algorithm, this metric can also be used to assess the quality of a clustering task based on the known cluster labels of the samples [33].Here, the known cluster labels were either be the disease labels or the batches. When evaluating the quality of the batch clustering, the silhouette score should then be on or below 0, indicating that clusters overlapped and were not easy to match.

When samples are instead clustered by disease labels, a higher silhouette score indicates that the case and control samples are more clearly clustered together. As the reason for applying batch correction is to remove confounding effects on the biological signal, batch correction should have a positive effect on this score. This positive effect could be outweighed by correction inevitably removing some biological signal as a side effect [34], which would result in a net decrease of the silhouette score.

2.3. Batch correction methods

While creating a statistical model for predicting a treatment variable is a common analysis done with microbiome dataset, these usually require the target label as covariate [4,22,24–26]. To enact the approach of a diagnostic test, we look specifically at predicting the health of a subject from an entirely new batch.

The chosen batch correction methods were divided into three categories. The first category consists of existing tools that are commonly used for batch correction, including ComBat, ComBat-Seq, and Limma removeBatchEffect. The second category includes the commonly used transformations mapping features one-to-one to remove batch effect. Lastly there is the category of encodings that extract features of the count data in order to find some common representation within the data to get rid of the batch effects. The methods chosen are listed in Table 2-2, along with a short description and brief implementation details.

In addition to the default settings of the selected, we also included a number of adaptations deemed promising. Quantile transformation was added with both a uniform and normal distribution, as well as in combination with a centered log-ratio transform, because this is a frequently used step towards standardizing datasets [22,35]. Feature selection was also applied by thresholding based on the variance, but only after first performing CLR, due to the high variance of the negative binomially distributed data causing thresholding to remove few to no features for uncorrected data.

A diagnostic test would use some form of online batch correction, continuously working on new samples. As such, the test dataset were corrected separately from the training datasets to simulate this behavior. In the last column of the table, we categorized the procedure that was used for this transformation, which aimed to use information learned from the training set to enable or improve the transformation.

When methods perform their correction in a batch-wise manner, not using information aggregated over multiple batches, they could be directly applied to the test dataset. An example of this is batch mean centering, which standardizes each batch separately. While useful in this comparison, this method is not ideal considering that in an online, continually operating, setting, batches would consist of single samples or small groups at most, meaning that such corrections would become inapplicable.

Our single 'Reference' method is ComBat, which has an explicit parameter to accept a batch that it maps the rest of the batches towards. Here, the training set could simple be given as a single reference batch, mapping the test set towards the training set, without modifying the training set.

In contrast, when this option was not available for other tools while still requiring multiple batches in order to function, a copy of the training dataset was appended after it was already corrected and named as a single batch. As the training dataset is always much larger than the testing dataset, this made it more likely for the latter to be aligned with the corrected training dataset.

All other methods were capable of learning their transformation on the training dataset and then applying that on both the training and testing dataset directly, in the same manner as normally done in preprocessing before machine learning models are trained. These methods transform batches in the same manner regardless of batch size, rendering it the preferred choice for online batch correction.

	Method	*Description*	*Implementation*	*Test set transformation*
Baseline	Baseline	Doing nothing but adding pseudo counts	All counts + 1	Batch-wise*
Existing Tools	ComBat	Empirical bayes method for adjusting Location and Scale	'ComBat' from the SVA package (3.42.0) from CRAN [36]	Reference**
	ComBat-Seq	Adaptation to ComBat for RNA-seq	'ComBatSeq' from the SVA package from CRAN (3.42.0) [36]	Appends***

	ReComBat	Adaptation to ComBat using ElasticNet	'reComBat' python package (0.1.0) [37]	Appends***
	Limma	Creates a Linear Model of batch effect, then subtracted	The removeBatchEffect function of the Limma package on CRAN (3.50.0) [38,39]	Appends***
Transformations	Batch Mean Centering (BMC)	Standardizes feature-wise	Subtract the mean, divide by the variance per batch (feature-wise)	Batch-wise*
	Centered Log-Ratio (CLR)	Log transforms then subtracts the geometric mean of features batch-wise	Own implementation	Batch-wise*
	Normalize	Standardizes feature-wise with learned transformer	Learn mean and variance per feature of training set	Learned[X]
	Isometric Log-Ratio (ILR)	Applied CLR then maps to the orthonormal basis of the CLR plane	Compositions package (2.0.4) from the CRAN dataset	Learned[X]
	Quantile (uniform)	Feature-wise mapping of distribution to uniform distribution	QuantileTransformer of the scikit-learn (1.0.2) package with default settings	Learned[X]
	Quantile (normal)	Feature-wise mapping of distribution to normal distribution	QuantileTransformer of the scikit-learn (1.0.2) package with normal distribution as output [40]	Learned[X]
	CLR + Quantile (uniform)	First perform CLR, then Quantile (uniform)	Own implementation	Learned[X]
	ILR + Quantile (normal)	First perform ILR, then Quantile (normal)	Own implementation	Learned[X]
Encodings	PCA (20)	Linear transformation to find components that explain variance, with 20 components	PCA transformer of the scikit-learn package (1.0.2) [40]	Learned[X]
	PCA (100)	Same as PCA (20), except with 100 components	PCA transformer of the scikit-learn package (1.0.2) [40]	Learned[X]
	CLR + VarianceThreshold	First perform CLR, then drop features with a variance below 0.1	Own implementation for CLR, VarianceThreshold from the scikit-learn library (1.0.2)	Batch-wise*
Details	*Batch-wise: The corrector worked in a batch-wise manner, and usage was equivalent on training and test sets	**Reference: The corrector used the training set as a known reference.	***Appends: The corrector appended the corrected training datasets as a single dataset before correction	[X]Learned: The batch corrector learned a set of parameters on the training dataset which could then be used when correcting the test dataset

220

Table 2-2 Batch correction methods used in the benchmark. Names are listed along with descriptions and implementations used. In addition, the method used for subsequently transforming of the test set is given. Full implementation details in the code repository. Methods are grouped and sorted by three categories with the baseline model at the top.

2.4. Set-up Machine Learning experiments

2.4.1. Binary classification evaluation

For evaluating the algorithms, we used the area under the receiver operating characteristic curve (AUROC), which summarizes the information of the receiver operating characteristic curve (ROC curve). After obtaining the probabilities that a classifier assigns to a sample being the case or control, the ROC curve can be obtained by calculating the true positive rate (TPR) over the false positive rate (FPR) at various thresholds of sensitivity. They are defined below in Equation 2-2.

$$\boldsymbol{TPR} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}}, \boldsymbol{FPR} = \frac{\boldsymbol{FP}}{\boldsymbol{FP} + \boldsymbol{TN}}$$

EQUATION 2-2 TRUE AND FALSE POSITIVE RATES (TPR AND FPR).TP = TRUE POSITIVE, FP = FALSE POSITIVE, TN = TRUE NEGATIVE, FN = FALSE NEGATIVE.

Here, true positives (TP) and false negatives (FN) indicate the number of cases of diseased samples that are classified as diseased and healthy samples respectively. True negatives (TN) and false positives (FP) representing the samples from, respectively, healthy patients that were classified as healthy and diseased patients.

We use the AUROC metric because of its prevalence in the literature, being one of the most commonly reported metrics. There is extensive discussion on whether that position is deserved, with a primary concern on inaccurate representation of imbalanced data. This concern was mitigated in this study by the usage of balanced datasets.

2.4.2. Baseline evaluation of dataset integration

When establishing a baseline performance of binary classification on uncorrected data, we used the Random Forest classifier, an ensemble method that takes the majority vote of 'forest' of decision trees to be its classification. Through aggregating multiple decision trees, it becomes more robust than a single one, which is prone to overtraining quickly. This method is comprehensively studied and a common baseline for classification within metagenomics [4,41–43]. The default settings were used for the baseline, which have been shown to produce globally near-optimal results compared to extensive tuning [44]. This allowed for the complete experiments to be repeated up to 1000 times within a reasonable timeframe.

When comparing the performances of the different algorithms, a Mann-Whitney U rank test was performed, implemented by SciPy (v.1.0.2.) [45], a non-parametric version of the t-test, testing the null hypothesis that the underlying distributions of the two independent samples being tested are the same [46]. This test was chosen for its robustness at handling outliers. We report p-values after correction, which are said to be significant if they are found to be less than 0.05 after Bonferroni correction.

2.4.3. Evaluation of batch correction algorithms

For the evaluation the batch correction algorithms we used a more extensive set of classification algorithms. We chose the most commonly used classifiers in the field, Logistic Regression, Support Vector Machine (SVM), and Random Forest classifiers, wherein a Stochastic Gradient Descent implementation was used for the logistic regression [4]. In addition, a Bernoulli naïve Bayes classifier, K nearest neighbor classifier, and Gradient Boosting classifier that were used in a similar benchmark [26] were added. Lastly, the Multinomial naïve Bayes classifier that was developed for count data was added, as the distribution of the raw data resembles a multinomial one [21,47]. The methods are listed and described in Table 2-3. The scikit-learn library (version 1.0.2) [40] was used to implement all classification algorithms.

Each of the classifiers was tuned using a randomized grid search over the parameter space given in the table, wherein all other parameters were left at the defaults of the scikit-learn library. The tuning used 10 random parameter selections, selecting the best using 5 internal cross-validations, wherein the validation was modified to optimize for the leave one dataset out (LODO) validation with the AUROC metric. The parameters that were validated on are shown in the last column of Table 2-3. The search was performed with RandomizedSearchCV from the scikit-learn library.

To evaluate each classifier on the test set, we use the same LODO approach [14], wherein each study is left out as the test set once, with the others used to train the classifier. We perform the Wilcoxon signed rank test to test the hypothesis that the classification scores of two classifiers come from the same distribution, with the samples paired by iteration, as each of the iterations will have scores evaluated on the same test set.

The p-values of the tests are corrected for multiple testing with Bonferroni correction, multiplying by the amount of tests performed within each classifier's results. As each batch correction algorithm's performance is compared only to the baseline, the number of tests performed is equal to the number of batch correction algorithms. P-values are reported after correction, and those that are below 0.05 after correction are said to be significant.

Name	*Description*	*Parameters varied*
Random Forest Classifier	Aggregating the decisions of multiple decision trees by taking the majority vote when choosing a class for a sample	- Number of trees - Number of features to consider each split - Maximum depth of each tree - Minimum number of samples for splitting internal node - Minimum number of samples for being a leaf node
Bernoulli NB	Multivariate Bernoulli naïve Bayes classifier which binarizes all its input	- Additive smoothing parameter - Whether to learn priors first
Gradient Boosting Classifier	Builds an single regression tree in additive fashion.	- Number of boosting stages - Learning rate for each stage - Maximum depth of tree
KNeighbors Classifier	Uses the k-nearest neighbors to vote on which class a point belongs to	- Number of neighbors to consider - Weighting of the neighbors - Algorithm to use - Distance metric to use
Multinomial NB	Multinomial naïve Bayes classifier	- Additive smoothing parameter - Whether to learn priors first
Stochastic Gradient Descent Classifier	Linear Classifier, including logistic regression trained through Stochastic Gradient Descent.	- The loss function - The regularization penalty - The learning rate
Support Vector Machine/Classifier	Finds a hyperplane margin that best separates the classes.	- Regularization strength

TABLE 2-3 THE BINARY CLASSIFICATION METHODS USED IN THE BENCHMARK , WITH A DESCRIPTION AND THE PARAMETER SPACE THAT WAS SEARCHED USING A RANDOM GRID SEARCH WITH 10 ITERATIONS. PARAMETERS NOT NAMED IN THE PARAMETER SPACE WERE LEFT AT DEFAULT. FULL PARAMETER SPACE EXPLORED CAN BE FOUND IN SUPPLEMENTARY SECTION 6.1.2

3. Results

This work presents an evaluation of batch effects and their impact on generalization. Firstly, an investigation of batch effect is presented, both before and after correction. Then, the generalization of performance without any removal of batch effects is assessed on a baseline binary classifier trained on either a single or mixed datasets. We conclude with a comparison of leave one dataset out (LODO) [14] performance of ML algorithms trained on each of the different correctors data, establishing whether batch correction should be performed before learning on new datasets.

3.1. Batch effects are reduced by some batch correctors

Before comparing the performance of binary classifiers on batch corrected data, we established to what extent batch effect was present in the data and whether batch correctors could reduce this while avoiding the loss of biological signal. We hypothesized that batch effects are present in the datasets and that batch correctors would show varying degrees of reduction, wherein the best correctors would reduce the batch effect while increasing biological signal detected. To test this hypothesis we considered both a visual angle to establish a intuition and numerical angle to evaluate it. We found that batch effect was present, differing in its strength between our datasets. Some correctors, including ComBat, decreased this effect without seeming to lose label information, while others increased both the correlation with the batch and label effects, which was unexpected.

3.1.1. tSNE visualization shows different degrees of batch correction

To visualize the effect of batch correction, the tSNE dimensional reduction was performed for each of the transformers as well as the baseline. In Figure 3-1 a representative subset of the transformers and studies is shown, while the full set of tSNE and UMAP visualizations for both studies and transformers can be found in Supplementary sections 6.2.2 and 6.2.3.

The baseline without any correction applied showed clustering for some studies but not all, indicating some level of batch effect. Some of the transformations showed more clustering afterwards, like 'Quantile (0),' 'BMC,' and 'CLR,' while reduced the clustering, as can be seen in 'ComBat-integrated.' Increased mixing for these algorithms suggests that batch effect is reduced, with the nearest neighbors of samples in a batch more frequently being from a different batch. This suggests that the batch correctors that decreased clustering could be more effective for reducing batch effects.

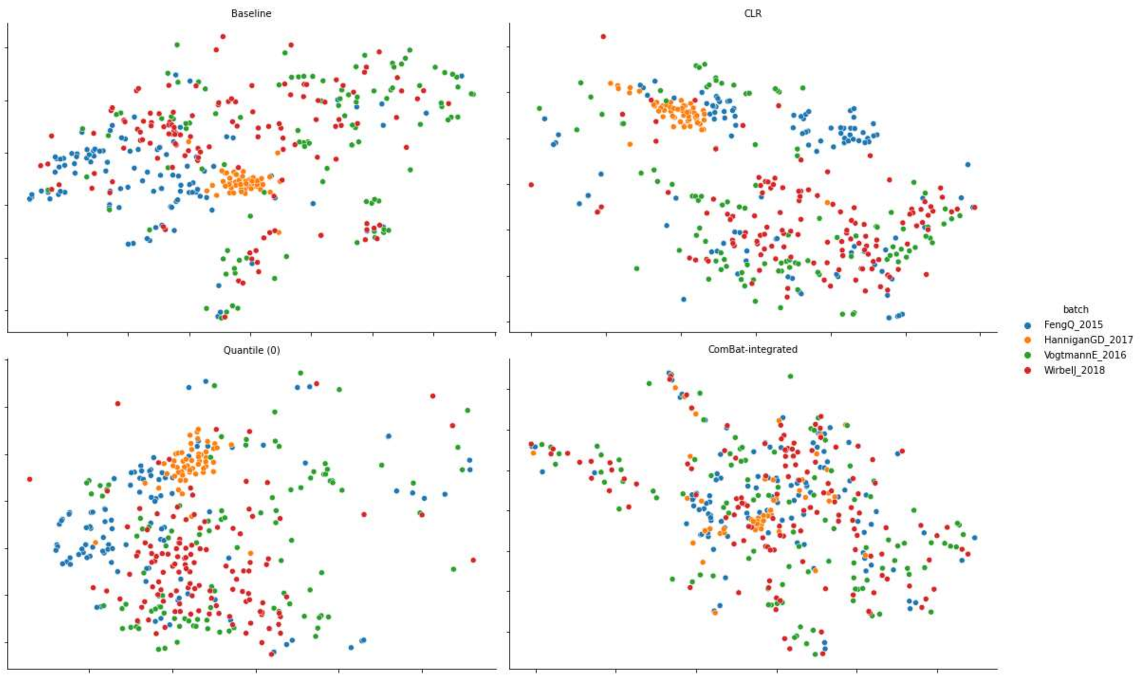

FIGURE 3-1 REPRESENTATIVE SUBSET OF TSNE PERFORMED ON ALL STUDIES AND BATCH CORRECTORS WITH DEFAULT SETTINGS. FROM TOP LEFT TO BOTTOM RIGHT THE BASELINE WITH NO TRANSFORMATION, THE CENTERED LOG-RATIO TRANSFORMATION, THE QUANTILE TRANSFORMATION AND THE COMBAT INTEGRATION ARE SHOWN.

3.1.2. Proportion of features correlated with batch identity is affected by batch correction

To quantitively evaluate batch effects on individual features, we determined the proportion of features significantly correlated with batch and disease labels, before and after batch correction, by fitting linear and negative binomial regression models. Also, we calculated the according silhouette scores based on batch and disease labels for the dataset corrected by different batch correction methods. The resulting proportions are shown in Table 3-1.

Without any processing, 18.2% of the dataset's features are significantly correlated with the batch after a Bonferroni correction, known to be especially conservative [48]. In contrast, only 1% of the features had the label as a significant factor, in both the linear and the negative binomial models. As some binary classification methods randomly pick features to consider, having only a few features with significant correlation with the label could increase the variance in performance between runs.

The results from batch correction tools all showed a decrease in the proportion of features that were significantly correlated with batch information, indicating a decrease in batch effect. ReComBat and Limma especially showed a large decrease, with the latter having no features where batch was a significant enough factor. Simultaneously, the proportion of label-correlated features stayed the same (ComBat-Seq and Limma) or increased marginally (ComBat and reComBat). This is interesting because batch correction tools are known to also remove at least some biological information [22], which should then reduce the proportion of features correlated with the batch. This is potentially because the linear model being better able to model the data after being normalized by the tools. As ComBat-seq outputs count data retaining the same proportion of label-correlated could thus be explained.

The transformations and encoders, in contrast, increased the proportion of features that had batch as a factor. This was especially clear in PCA (20), where only 1 of the 20 features did not have a significant correlation with the batch. PCA (20) had no features with a significant correlation to the label, indicating that the first 20 principal components of the data did not have relevant biological signals. CLR + VarianceThreshold had a higher proportion of correlated features with both the label and batch than only applying CLR, which is to be expected considering that the VarianceThreshold removes features with low variance.

	CORRECTOR	FEATURES WITH BATCH AS FACTOR , NB MODEL (%)	FEATURES WITH BATCH AS FACTOR, LINEAR MODEL (%)	FEATURES WITH LABEL AS FACTOR , NB MODEL (%)	FEATURES WITH LABEL AS FACTOR , LINEAR MODEL (%)
BASELINE	*Baseline*	*18.2*	*18.5*	*1.0*	*1.0*
TOOLS	ComBat	-	7.6	-	**1.3**
	ComBat-seq	9.9	10.1	1.0	1.0
	ReComBat	-	0.5	-	**1.8**
	Limma	-	0.0	-	1.0
TRANSFORMATIONS	Normalize	-	**20.7**	-	1.0
	BMC	-	**57.6**	-	1.0
	CLR	-	**37.4**	-	**5.7**
	ILR	-	**28.6**	-	**2.7**
	ILR + Quantile	**83.3**	**83.3**	**45.9**	**45.9**
	CLR + Quantile	**79.3**	**79.3**	**5.8**	**5.8**
	Quantile (normal)	**39.6**	**39.6**	**6.0**	**6.0**
	Quantile (uniform)	-	**39.3**	-	**5.4**
ENCODINGS	PCA (20)	-	**95.0**	-	-
	PCA (100)	-	**38.0**	-	**4.0**
	CLR + VarianceThreshold	-	**44.0**	-	**7.0**

TABLE 3-1 PROPORTIONS OF FEATURES CORRELATED WITH BATCH AND LABEL. FOR EACH CORRECTOR, IT LISTS THE PERCENTAGE OF FEATURES THAT WERE SIGNIFICANTLY (AFTER CORRECTION FOR MULTIPLE TESTING) CORRELATED WITH THE BATCH AND LABEL BOTH FOR A LINEAR MODEL AS WELL AS A NEGATIVE BINOMIAL (NB) MODEL. PROPORTIONS WERE CHOSEN AS THE TOTAL NUMBER OF FEATURES CHANGES FOR ENCODINGS. INCREASED PROPORTIONS WERE MARKED BY BOLDING VALUES.

To assess the sum effect of the batch correctors on the sample , we evaluated both a batch and label clustering with silhouette scores which average the dissimilarity of points to their cluster compared to the nearest other cluster. When the clustering aligns with the structuring of the data, the silhouette score will be positive and increase to 1, while seemingly random assignments will have a score of 0 for two clusters and more negative scores for multiple clusters. The silhouette scores are shown as boxplots in Figure 3-2. The silhouette score of the baseline varies for all but the baseline due to cross-validation

The batch clustering showed that most of the batch correctors increased the chance of a misclassification, decreasing the dataset's average silhouette coefficient. As there are eleven clusters present for batch clustering this is in line with the expectation that batch correctors correct for batches clumping together, with the odds of other clusters being closer to a random sample becoming higher as the clusters become more mixed. The largest change in performance is from ILR+Quantile which seems to have the most mixed batch corrected data. This seems to contradict the observation made from the feature-wise analysis, possibly due to the differences observed by the feature-wise model, while statistically significant, not large enough to impact the cosine similarity metric.

For the label clustering the range of differences is 30x smaller than that of the batch clustering, showing only slight changes in the silhouette scores between the baseline and the correctors. This was possibly due to the number of clusters being much smaller (two instead of eleven) while still overlapping, such that both clusters would have approximately the same distance to each distance on average.

ILR + Quantile showed the largest increase in performance, with a median performance increase of 0.009 compared to baseline ($p << 1e-10$ after correction). This contrasts with the observation from the batch clustering, aligning with the result of the feature-wise analysis, where this corrector was the only one which increased the proportion of correlated features to almost half. The other transformations similarly align with what was observed in the correlation analysis, with all the tools showing a smaller increase than the transformations, with the exception of BMC, ILR, and Normalize, just as in that analysis. This seems to confirm that they clarify biological signal, but not necessarily removing batch effect in the process.

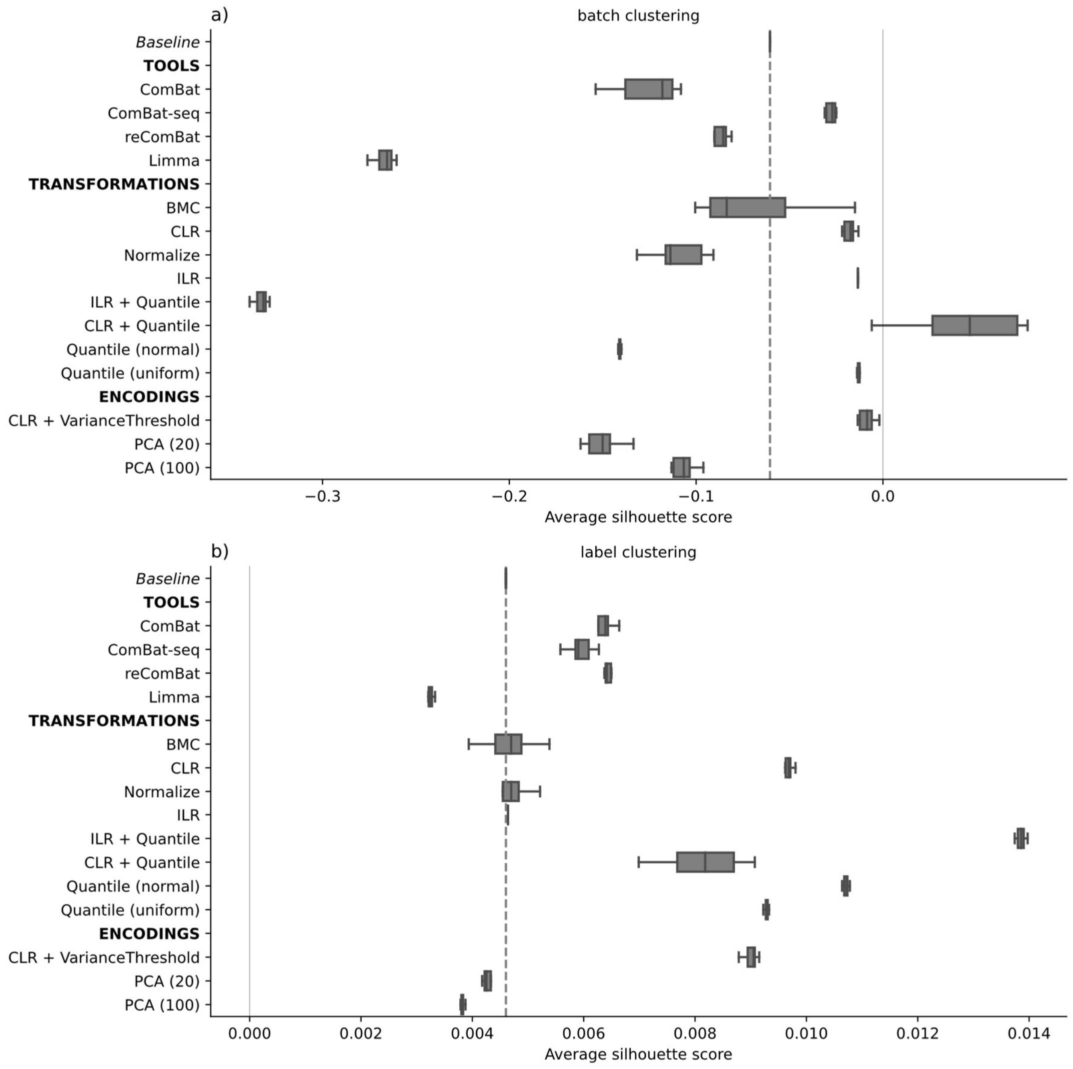

Figure 3-2 Boxplot with silhouette scores for Batch (a) and Label(b) clustering. Obtained by concatenating training and test set after batch correction, then averaging the silhouette widths across samples. The cosine metric was used to compute distance between points. The median of the baseline was drawn across the y-axis to allow for easier comparison with baseline. All correctors had significantly different silhouette scores with baseline ($p << 1E-6$) with exception of BMC for both label and batch clustering and Normalize for label clustering. Boxes show IQR, whiskers show furthest point within 1.5 IQR. Outliers are marked with diamonds. In order to show differences clearly, domain of subfigure A is (-0.35, 0.1) while domain of subfigure B is (0.0 , 0.015). Scores for label clustering center around 0 closely, while varying between -0.3 and 0.1 for batch clustering.

3.2. Learning from multiple batches increases generalization

In this section, we investigate the effect of learning from multiple datasets on binary classification performance on unseen batches. While having more data is usually considered to lead to better generalization [49], multiple datasets could destabilize results and prevent improvement with batch effects. We hypothesized that using multiple datasets would have a positive influence on the generalization of the classifiers by forcing them to be able to handle different batch effects already. We showed that datasets did not generalize well towards other testing datasets, with continually increasing performance when training on multiple datasets. Even when kept at the same size, using samples from different datasets improved scores over a single dataset when testing on an unseen dataset.

3.2.1. Single datasets have significantly better performance on their own test set

To assess the difference in classification performance between a test set of the same batch and that of different batches, we set up a new experiment with uncorrected data. We first split the datasets into training and testing sets with a stratified 80/20 split, train baseline classifiers on each of the separate batches, and then test their classification performance on both the test set of the same batch, or of all the other batches. Subfigure a of Figure 3-3 shows boxplots with the respective results.

The classification performance of the binary classifiers when tested on the remainder of their own dataset, shown in blue in the figure, is volatile, with most having a high spread in performance. This is largest for the studies by Thomas et al. [14] and Hannigan et al. [50], which vary from worse than random (0.5) to perfect classification (1.0). The dataset by Gupta et al. had median performance on its own dataset of close to perfect classification, with perfect classification falling within its IQR, indicating this dataset is easy to distinguish for the classifier. As the number of samples in the test datasets is small we consider it is most plausible that the choice of test dataset causes this variation in performance.

The performance of the classifiers on the rest of the datasets is hard to compare, as each of the classifiers leaves out a different dataset (its own) when testing. The variation in performances was a lot smaller for these performance tests, likely due to the test set not varying each iteration.

A comparison between the performance on the own test set and the remainder of the datasets shows that the performance on the own test set is significantly better. Eight out of the eleven datasets were significantly better, two (Yachida et al. and Vogtmann et al.) showed significantly worse performance, and one showed no significant difference. For the Yachida et al. dataset, we hypothesized that the increased test performance was due to the number of samples within this study being the equivalent of five of the other studies, which both makes for more samples to learn from and avoid overfitting on the smaller data.

To investigate this hypothesis, we reran the same experiment while limiting the training set size to a constant size, choosing the training set size of the smallest dataset (n=40), shown in subfigure b of Figure 3-3. The classification performance of most datasets dropped to 0.6, except whose performance remained around the same due to the sample size already being close to 40 in the 80/20 split. The performance of the dataset for Yachida et al. dropped drastically, with the median performance going from 0.77 to 0.60, though it still remained statistically higher ($p < 0.05$) than the score on its own dataset, which dropped to only slightly above random.

We conclude that the performance of classifiers trained on a single dataset perform better on that dataset's test split than on different, separate datasets. The bias in datasets is easily overtrained on it seems. The next section will look at whether training the same classifier on multiple of these datasets allows it take these biases into account for new datasets.

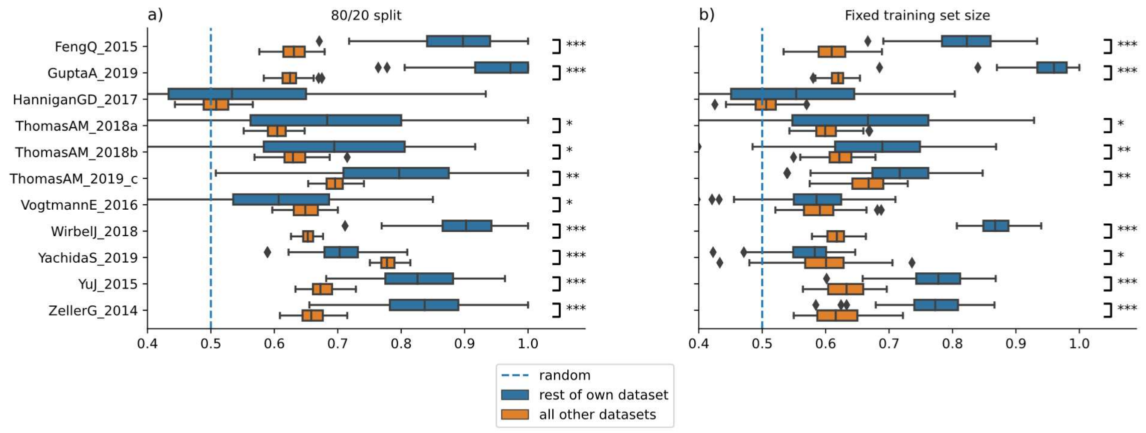

Figure 3-3 Boxplot comparing performance of random forest binary classifier with default settings for predicting disease in both a train-test split of its own study dataset (blue) as well as all other datasets (orange). Training size was either 80% of the study (subfigure a) or exactly 40 samples (subfigure b). Significant differences as tested with a MannWhitneyU test are marked with a bracket, number of stars indicating scale of p-value, with *p<0.05, **p< 1e-8, ***p<1e-30 (after Bonferroni correction). All significant differences have rest of own dataset as greater value, except for Yachida et. al. where the performance on all other datasets was significantly greater. Test split of own data shows equal or better performance for all studies for both types of splits .

3.2.2. Training on more batches increases generalization on new batches

To evaluate whether and to what extent using more datasets as training increased generalization performance, we measured the AUROC of the same baseline Random Forest model trained on an increasing number of uncorrected datasets, testing on two datasets that were excluded, randomly chosen each of the 1000 iterations. Performance showed a significant increase ($p < 0.05$ after correction) when combining more than two datasets. After more than seven datasets were combined the increase saturated, no longer significantly increasing. This shows that using multiple datasets will increase binary classification performance on new datasets, but does not isolate whether this is due to the number of samples or due to the increased diversity from multiple datasets.

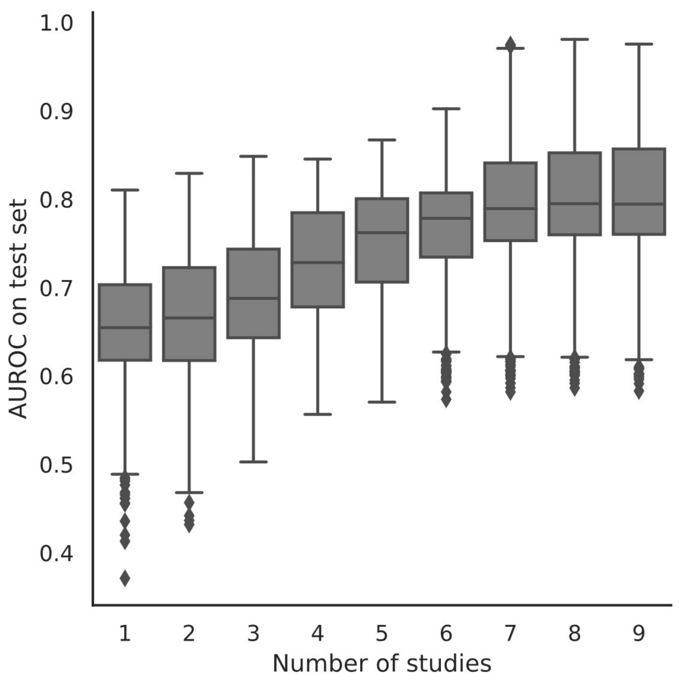

FIGURE 3-4 BINARY CLASSIFICATION PERFORMANCE MEASURED USING THE AUROC METRIC OF A RANDOM FOREST CLASSIFIER RE-TRAINED ON AN INCREASING NUMBER OF STUDIES, TESTED ON A SEPARATE TEST SET. BOXES SHOW QUARTILES OF THE DISTRIBUTION, WHISKERS SHOW LARGEST OBSERVED DATAPOINT WITHIN 1.5 IQR AND OUTLIERS ARE DRAWN SEPARATELY. ENTIRE EXPERIMENT WAS REPEATED 1,000 TIMES. SCORE INCREASES SIGNIFICANTLY AT ALL STEPS BETWEEN 2 AND 7, BUT DOES NOT SHOW AN INCREASE BETWEEN STEPS 7, 8, 9.

To assess whether the increased diversity by itself had an impact on generalization performance, we trained the same classifier on fixed numbers of samples of one dataset, Yachida et al., or that same number of samples of all other training datasets before testing it on two unseen test datasets, the result of which is shown in Figure 3-5. There was a significant improvement in performance at each sample size ($p < 0.0001$, increase in median between 0.046-0.0556 at each step), showing that the usage of multiple datasets is better than a single one for generalization, even with the same sample size.

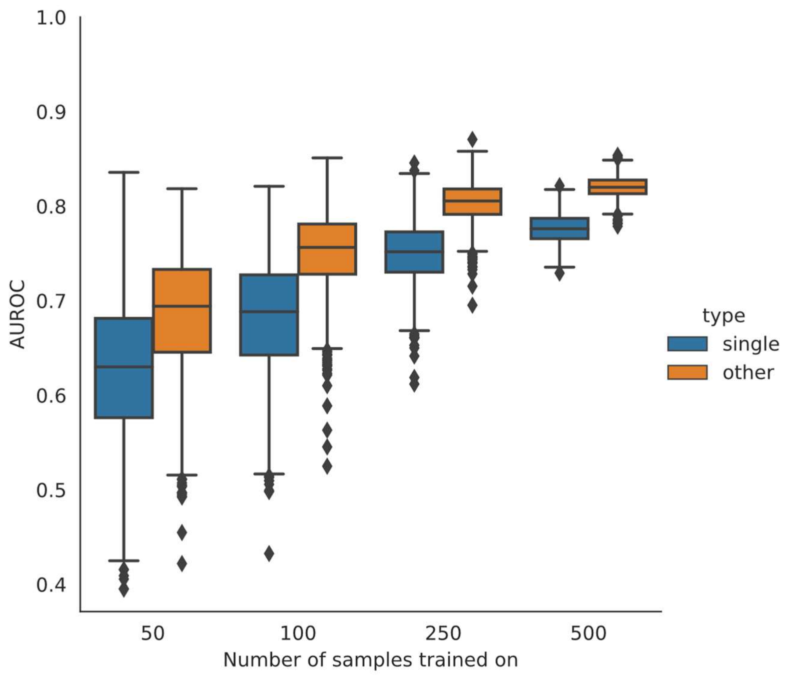

Figure 3-5 Boxplots of Binary classification performance of a random forest classifier on an unseen test set, trained on 50, 100, 250, and 500 samples taken from either a single study (Yachida et al.), shown in blue or all other training studies, shown in orange. Experiment was repeated 1000 times. Samples taken from multiple studies have a higher average score and show less variance than samples taken from the single study across all sample sizes, though there is still overlap. For the sample sizes 50, 100, 250, and 500 the change in median was 0.056, 0.050, 0.047 and 0.047 respectively, with all p << 1E-5

3.3. Current batch transformations have no significant impact on the generalization of classification performance

In this section we investigated what the ideal pair of batch corrector and binary classifier is when evaluated on unseen test datasets using the AUROC metric. We hypothesized that those correctors that showed an increase in the feature correlation and silhouette score would have an improved performance over a baseline classifier. The classifier that performed the best on average for the baseline, the random forest classifier, was not outperformed by any combination in a statistically significant manner, showing that batch effect did not have as much of an impact as was expected.

3.3.1. Pipeline for binary classification benchmark

To perform the benchmark that was proposed in an organized manner that remains reproducible, a more elaborate setup was required. To this end, we developed a pipeline which can be used to perform mass batch correction, training, and tuning with nested cross-validation, while remaining easy to setup and use. In addition to these goals, we also took into account some common pitfalls, which we describe below along with our steps for their mitigation. The problems explicitly addressed are those listed as common for microbiome research in a survey of more than one hundred studies, performed by the ML4Microbiome consortium in 2021 [4]. An overview of the pipeline is shown in Figure 3-6.

To avoid performing feature selection on the entire dataset, the first step of the pipeline is to split the dataset into a training and test set. This prevents features found significant in the test set from leaking towards the models.

To correct for the winner's curse, wherein the best algorithm can be unduly chosen because of random chance, the pipeline performs the entire cross-validation process ten times, cross-validating each time. In addition, balanced datasets are used which reduce the overestimation that could be produced by this curse.

To enforce appropriate splitting of the datasets, wherein a lack of stratification leads to imbalanced validation and testing datasets, all validation approaches leave entire datasets out. As each dataset was balanced, this maintained class distributions across folds.

To avoid handling repeat measurements, which violates the assumptions that samples are identically and independently distributed (i.i.d.) used by cross-validation, we filtered out all repeat measurements of samples in our datasets.

To simulate diagnostic tests, each test dataset was batch corrected with learned values if possible, else was combined with the training data in a separate correction procedure that would prevent information leakage.

We developed the pipeline with the Nextflow framework [51], a bioinformatics framework designed for reproducible omics workflows. Each of the jobs of the pipeline runs in isolation on a hand-crafter docker container with the capability of adjusting the resources allocated to it as well as the way it is processed through changing the label of the job. Jobs can be automatically queued on clusters using SLURM or similar job management software, and run using docker containers, either using Docker or Singularity when supported. With this implementation, a full evaluation of more than 10,000 classifiers could be achieved with Singularity and Nextflow as the only required dependencies on a local system.

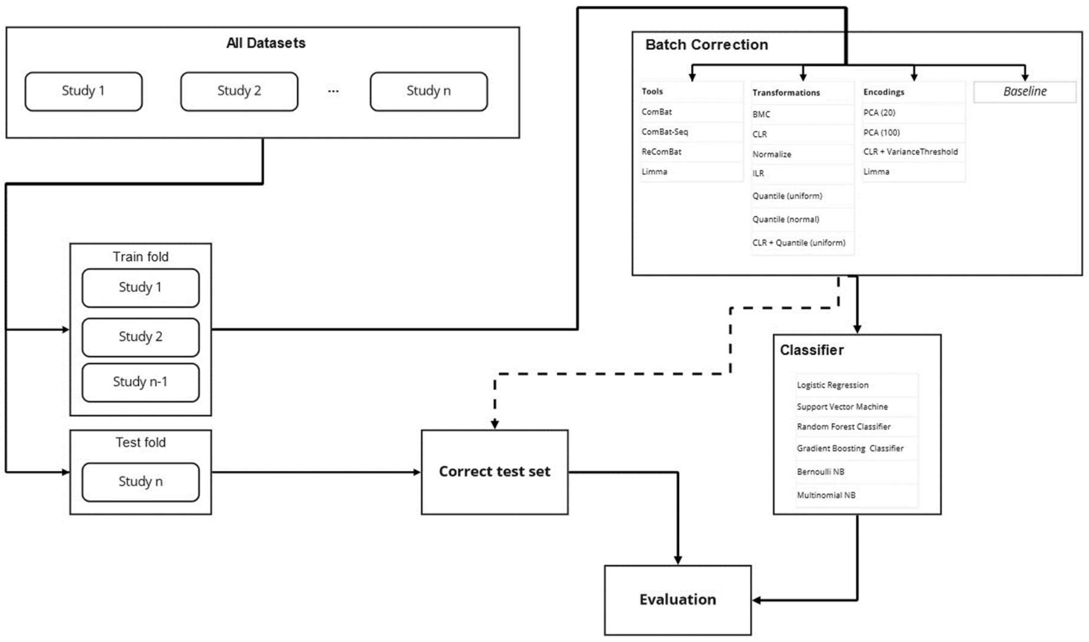

FIGURE 3-6 PIPELINE OF THE PROCESS USED TO ANALYZE BATCH CORRECTION AND ENCODING ALGORITHMS. THE DATA IS FIRST SPLIT INTO BATCHES, WITH EITHER 1 OR 2 BATCHES LEFT OUT FOR TESTING. THE TRAINING DATA IS THEN TRANSFORMED WITH ONE OF THE BATCH CORRECTION ALGORITHMS (SEE METHODS FOR MORE DETAILS FOR EACH ALGORITHM). THE TEST SET IS THEN TRANSFORMED, POTENTIALLY WITH THE TRANSFORMER USING PARAMETERS LEARNED FROM THE TRAINING DATA (INDICATED BY THE DOTTED ARROW). A CLASSIFIER IS TRAINED ON THE TRANSFORMED TRAINING DATA AND THEN TESTED ON THE TRANSFORMED TESTING DATA (EVALUATION). SEPARATELY, A BATCH EFFECT DETECTION PROCEDURE IS PERFORMED ON THE TRANSFORMED TRAINING DATA TO DETECT THE PRESENCE OF BATCH EFFECTS.

3.3.2. Batch correction algorithm does not have a significant impact

To investigate the impact of the batch correction algorithm on the model performance, we ran our pipeline ten times on each of the twelve batch correctors, and then trained and tuned seven machine learning algorithms with the corrected microbiome data. Comparing the performances based on their medians, the best performing classification algorithm for each of the batch correction method is displayed in Figure 3-7, with the complete set of scores for each combination of the binary classifiers and batch correctors plotted in Supplementary Figure 6-4.

The baseline shows a good performance on the test sets, with a median of 0.81 for its best-performing classifier, the RandomForest classifier. However, it has a large variance, having both scores worse than random choice and perfect classification. The perfect classification was the result of testing on the dataset of Gupta et al., which showed high performance in earlier testing as well (see section 3.2.1), likely as a result of easily distinguishable samples. The lower performance was not the result of any particular dataset, more likely a result of overtraining.

The other binary classifiers-corrector combinations do not significantly improve upon the baseline. Most median AUROCs were below baseline performance with the exception of ComBat-seq, CLR, Quantile (uniform) and CLR+VarianceThreshold, which only marginally outperformed it. The best classifier for ComBat had a significantly worse performance than the baseline ($p < 0.0001$), which is especially noteworthy considering the popularity of this tool for batch correction, and how the adaptation towards the test set used a built-in functionality.

The Quantile (normal) transformer in combination with the Bernoulli Naïve Bayes classifier did show one advantage in its consistency. With a lowest score of 0.59, it avoided the many outliers of the random forest classifier, although its 0.25th quantile was lower than that of the baseline’s random forest classifier. The more stable performance of this pair is likely because the Bernoulli Naïve Bayes is not as sensitive as the Random Forest, due to binarizing its input as a first step.

The two settings of PCA performed significantly worse than the baseline, showing that the biological signal is not completely encoded in the first principal components. The encoding with 100 components, PCA (100), outperforming PCA (20), which only has 20, indicating that the last 80 components contain biological signal that is otherwise lost. The variance of the high-dimensional data is likely so high that the first principal components capture more irrelevant noise than biological signal.

Random Forest classification was the best performing in six of the twelve correctors in terms of median, while the Gradient Boosting Classifier and Bernoulli NB models outperformed the rest in five and three occasions respectively. All classifiers showed high variance, with many badly performing outliers for even the best performing classifiers. The Bernoulli NB performed exactly as if guessing randomly for the baseline, likely because its internal binarization of each feature did not account for pseudo counts, but this is likely also the reason for its comparative lack of outliers. The most consistent performance was from the Multinomial Naïve Bayes, even considering it could only run on the baseline, Combat-seq, CLR + Quantile (uniform), and Quantile(uniform) outputs because of only accepting non-negative values, with no performance below random guessing at 0.5.

Overall, considering the best pair of batch corrector and classifier, no one combination is best. Performing no correction before using a random forest classifier will, on average, not lead to worse performance than that of any other corrector-classifier pair. However, to avoid worse-than-random performance the Quantile (normal) transformer can be used in combination with the Bernoulli Naïve Bayes classifier. But even then, the high variance in performance would not recommend these classifiers for aiding in diagnosis.

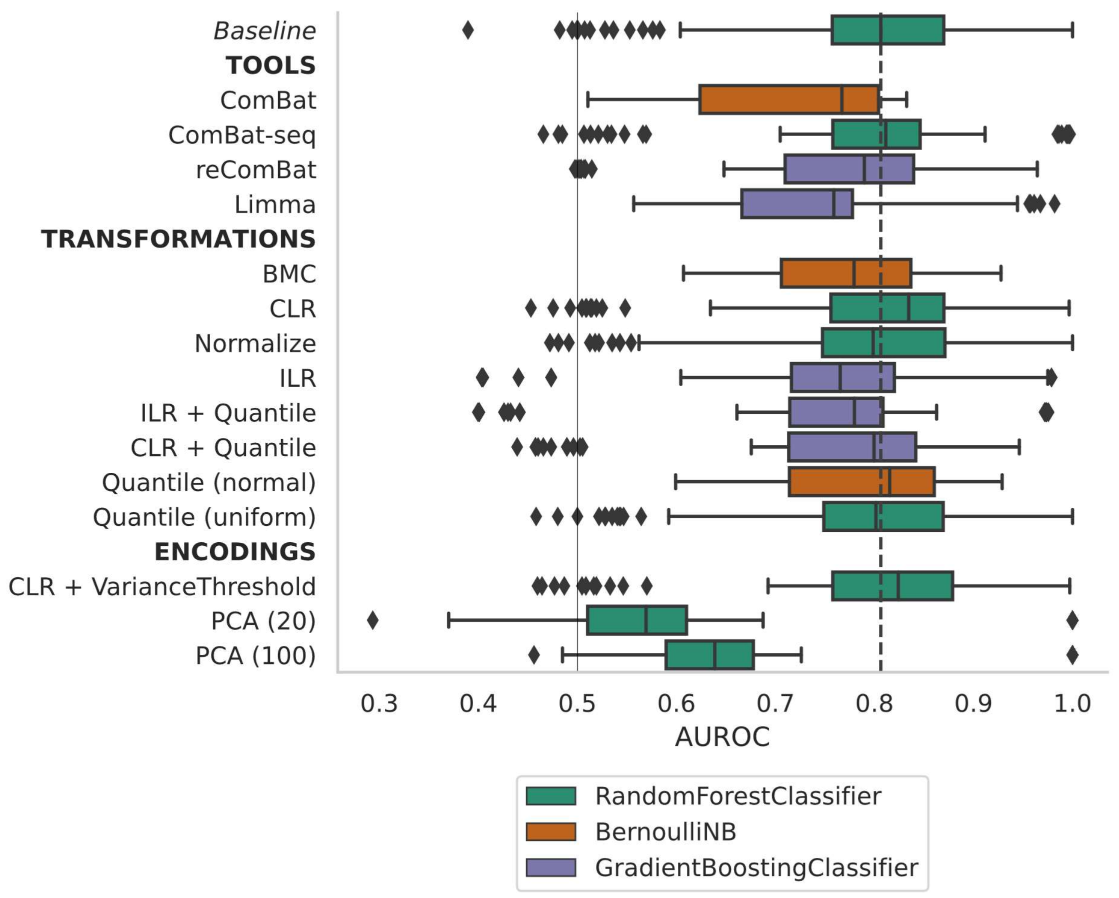

FIGURE 3-7 BINARY CLASSIFICATION SCORES OF BEST PERFORMING CLASSIFIERS FOR EACH OF 10 DIFFERENT (COMBINATIONS OF) BATCH CORRECTION METHODS, CATEGORIZED BASED ON THE TYPE OF BATCH CORRECTOR. PERFORMANCE MEASURED IN AUROC. BOXES INDICATE QUARTILES OF DISTRIBUTION, WITH WHISKERS AT FURTHEST POINTS WITHIN 1.5 TIMES IQR. COLOR INDICATES THE BINARY CLASSIFICATION MODEL THAT HAD THE HIGHEST MEDIAN FOR THE CORRESPONDING CORRECTOR AND WHOSE SCORES WAS USED IN THE FIGURE.

4. Conclusion

This work has investigated how best to design a binary classifier for an unseen dataset identifying patients with colorectal cancer. Taxonomic count data obtained from a shotgun metagenomic analysis of the gut microbiome was chosen as datatype, because it can be obtained non-invasively and is becoming more accessible, while allowing for highly accurate prediction. To account for batch effect when combining multiple datasets, combinations of batch correctors and binary classifiers were evaluated.

This was the first such benchmark performed on shotgun metagenomics of the human microbiome, with a comprehensive set of both batch correction methods as well as binary classifiers, tested in combinations. The pipeline that was designed allowed for the massive evaluation of more than 10,000 classifiers in a reproducible manner.

Batch effect was first mapped and analyzed, showing that it was indeed present, though not always strongly. Many of the batch correctors indeed reduced the correlation between features and the batch, though not always in both feature-wise correlations as well as clustering evaluation. Our metrics also indicated that some transformations improved the clarity of the biological signal, allowing the disease label to be more easily distinguishable.

Then, it was shown that using single dataset for training and then testing on a separate batch will have significantly different performance then testing on another part of the training dataset. Multiple datasets improved the generalization of binary classification models, even when the total number of samples was equivalent. This led us to conclude that with more diversity, the classifier can learn to ignore batch effect.

Lastly, combinations of batch correctors and binary classifiers trained and tested on new datasets in a manner approaching how diagnostic tests would be performed. We showed that no classifier could significantly outperform the baseline classifier, and that ComBat, one of the most commonly applied tools, though shown to remove batch effect detectably, caused subsequent binary classification performance to be significantly worse than the baseline. PCA encodings also decreased performance, showing that the biological signal for CRC was not encoded in the highest variance components.

All classifiers had a significant variance in their performance, causing many to have worse-than-random performance on occasion. A Quantile transformation to a normal distribution and then training with the Naïve Bayes classifier decreased the variance and could be a better choice to avoid outliers. Using these methods within a diagnostic setting would require

What then, is the best approach to deal with batch effects for new unseen datasets? We conclude that training the model on as many different datasets is key towards obtaining the best generalization. Batch correction will have little to no impact and could even reduce the classification performance, even though visibly reducing batch effect. Large datasets from different populations that are clearly labeled by disease will allow future research to create models that can accurately determine whether a patient is likely suffering from CRC.

While this benchmark was comprehensive, it was also limited in scope, exploring eleven datasets that were remarkably similar in composition, with balanced case-control sample amounts wherein all where shotgun metagenomes. While this made for a more controlled comparison, future research could broaden the scope of such a comparison to include different data sources like 16S rRNA pyrosequencing, or expand to different diseases like inflammatory bowel disease (IBD) or Autism Spectrum Disorder. Future research could also look into whether batch correction improves results when applied to a progressively smaller number of datasets. This will make this work more broadly applicable.

The code for the pipeline described in this work and to reproduce the figures can be found at https://github.com/AbeelLab/ngs-batch-evaluation along with a description on its usage.

5. References

1. Gibson GR, Probert HM, Loo JV, Rastall RA, Roberfroid MB. Dietary modulation of the human colonic microbiota: updating the concept of prebiotics. Nutrition Research Reviews. 2004;17: 259–275. doi:10.1079/NRR200479

2. Haro C, Rangel-Zúñiga OA, Alcalá-Díaz JF, Gómez-Delgado F, Pérez-Martínez P, Delgado-Lista J, et al. Intestinal Microbiota Is Influenced by Gender and Body Mass Index. PLOS ONE. 2016;11: e0154090. doi:10.1371/journal.pone.0154090

3. Kim D, Hofstaedter CE, Zhao C, Mattei L, Tanes C, Clarke E, et al. Optimizing methods and dodging pitfalls in microbiome research. Microbiome. 2017;5: 52. doi:10.1186/s40168-017-0267-5

4. Marcos-Zambrano LJ, Karaduzovic-Hadziabdic K, Loncar Turukalo T, Przymus P, Trajkovik V, Aasmets O, et al. Applications of Machine Learning in Human Microbiome Studies: A Review on Feature Selection, Biomarker Identification, Disease Prediction and Treatment. Frontiers in Microbiology. 2021;12: 313. doi:10.3389/fmicb.2021.634511

5. Tarca AL, Carey VJ, Chen X, Romero R, Drăghici S. Machine Learning and Its Applications to Biology. PLOS Computational Biology. 2007;3: e116. doi:10.1371/journal.pcbi.0030116

6. Flemer B, Lynch DB, Brown JMR, Jeffery IB, Ryan FJ, Claesson MJ, et al. Tumour-associated and non-tumour-associated microbiota in colorectal cancer. Gut. 2017;66: 633–643. doi:10.1136/GUTJNL-2015-309595

7. Sung H, Ferlay J, Siegel RL, Laversanne M, Soerjomataram I, Jemal A, et al. Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries. CA: A Cancer Journal for Clinicians. 2021;71: 209–249. doi:10.3322/caac.21660

8. Mahasneh A, Al-Shaheri F, Jamal E. Molecular biomarkers for an early diagnosis, effective treatment and prognosis of colorectal cancer: Current updates. Experimental and Molecular Pathology. 2017;102: 475–483. doi:10.1016/j.yexmp.2017.05.005

9. Wang Y, Lêcao KA. Managing batch effects in microbiome data. Briefings in Bioinformatics. 2020;21: 1954–1970. doi:10.1093/bib/bbz105

10. Gupta VK, Kim M, Bakshi U, Cunningham KY, Davis JM, Lazaridis KN, et al. A predictive index for health status using species-level gut microbiome profiling. Nature Communications. 2020;11. doi:10.1038/s41467-020-18476-8

11. Yu J, Feng Q, Wong SH, Zhang D, Liang QY, Qin Y, et al. Metagenomic analysis of faecal microbiome as a tool towards targeted non-invasive biomarkers for colorectal cancer. Gut. 2017;66: 70–78. doi:10.1136/gutjnl-2015-309800

12. Group JCHMPDGW. Evaluation of 16S rDNA-Based Community Profiling for Human Microbiome Research. PLOS ONE. 2012;7: e39315. doi:10.1371/JOURNAL.PONE.0039315

13. Beghini F, McIver LJ, Blanco-Míguez A, Dubois L, Asnicar F, Maharjan S, et al. Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with bioBakery 3. eLife. 2021;10: 2020.11.19.388223. doi:10.7554/eLife.65088

14. Thomas AM, Manghi P, Asnicar F, Pasolli E, Armanini F, Zolfo M, et al. Metagenomic analysis of colorectal cancer datasets identifies cross-cohort microbial diagnostic signatures and a link with choline degradation. Nat Med. 2019;25: 667–678. doi:10.1038/s41591-019-0405-7

15. Duvallet C, Gibbons SM, Gurry T, Irizarry RA, Alm EJ. Meta-analysis of gut microbiome studies identifies disease-specific and shared responses. Nature Communications. 2017;8: 1–10. doi:10.1038/s41467-017-01973-8

16. Leek JT, Scharpf RB, Bravo HC, Simcha D, Langmead B, Johnson WE, et al. Tackling the widespread and critical impact of batch effects in high-throughput data. Nat Rev Genet. 2010;11: 733–739. doi:10.1038/nrg2825

17. Gibbons SM, Duvallet C, Alm EJ. Correcting for batch effects in case-control microbiome studies. PLOS Computational Biology. 2018;14: e1006102. doi:10.1371/JOURNAL.PCBI.1006102

18. Goh WWB, Yong CH, Wong L. Are batch effects still relevant in the age of big data? Trends in Biotechnology. 2022 [cited 8 Apr 2022]. doi:10.1016/j.tibtech.2022.02.005

19. Lazar C, Meganck S, Taminau J, Steenhoff D, Coletta A, Molter C, et al. Batch effect removal methods for microarray gene expression data integration: a survey. Briefings in Bioinformatics. 2013;14: 469–490. doi:10.1093/bib/bbs037

20. Zhang Y, Jenkins DF, Manimaran S, Johnson WE. Alternative empirical Bayes models for adjusting for batch effects in genomic studies. BMC Bioinformatics. 2018;19: 1–15. doi:10.1186/S12859-018-2263-6/TABLES/2

21. Dai Z, Wong SH, Yu J, Wei Y. Batch effects correction for microbiome data with Dirichlet-multinomial regression. Bioinformatics. 2019;35: 807–814. doi:10.1093/bioinformatics/bty729

22. Wang Y, Lêcao KA. Managing batch effects in microbiome data. Briefings in Bioinformatics. 2020;21: 1954–1970. doi:10.1093/bib/bbz105

23. Li T, Zhang Y, Patil P, Johnson WE. Overcoming the impacts of two-step batch effect correction on gene expression estimation and inference. bioRxiv. 2021; 2021.01.24.428009. doi:10.1101/2021.01.24.428009

24. Hasic Telalovic J, Music A. Using data science for medical decision making case: role of gut microbiome in multiple sclerosis. BMC Med Inform Decis Mak. 2020;20: 262. doi:10.1186/s12911-020-01263-2

25. Ai L, Tian H, Chen Z, Chen H, Xu J, Fang J-Y. Systematic evaluation of supervised classifiers for fecal microbiota-based prediction of colorectal cancer. Oncotarget. 2017;8: 9546–9556. doi:10.18632/oncotarget.14488

26. Kubinski R, Djamen-Kepaou J-Y, Zhanabaev T, Hernandez-Garcia A, Bauer S, Hildebrand F, et al. Benchmark of Data Processing Methods and Machine Learning Models for Gut Microbiome-Based Diagnosis of Inflammatory Bowel Disease. Frontiers in Genetics. 2022;13. Available: https://www.frontiersin.org/article/10.3389/fgene.2022.784397

27. Tom JA, Reeder J, Forrest WF, Graham RR, Hunkapiller J, Behrens TW, et al. Identifying and mitigating batch effects in whole genome sequencing data. BMC Bioinformatics. 2017;18: 351. doi:10.1186/s12859-017-1756-z

28. Pasolli E, Schiffer L, Manghi P, Renson A, Obenchain V, Truong DT, et al. Accessible, curated metagenomic data through ExperimentHub. Nature methods. 2017;14: 1023. doi:10.1038/NMETH.4468

29. Van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008;9.

30. Feng J, Xu H, Yan S. Robust PCA in High-dimension: A Deterministic Approach. : 8.

31. Zhang Y, Parmigiani G, Johnson WE. ComBat-seq: batch effect adjustment for RNA-seq count data. NAR Genomics and Bioinformatics. 2020;2: lqaa078. doi:10.1093/nargab/lqaa078

32. Salton G, McGill MJ. Introduction to modern information retrieval. New York: McGraw-Hill; 1983.

33. Zhao S, Sun J, Shimizu K, Kadota K. Silhouette Scores for Arbitrary Defined Groups in Gene Expression Data and Insights into Differential Expression Results. Biological Procedures Online. 2018;20: 5. doi:10.1186/s12575-018-0067-8

34. Oytam Y, Sobhanmanesh F, Duesing K, Bowden JC, Osmond-McLeod M, Ross J. Risk-conscious correction of batch effects: maximising information extraction from high-throughput genomic datasets. BMC Bioinformatics. 2016;17: 332. doi:10.1186/s12859-016-1212-5

35. Gloor GB, Macklaim JM, Pawlowsky-Glahn V, Egozcue JJ. Microbiome datasets are compositional: And this is not optional. Frontiers in Microbiology. 2017;8: 2224. doi:10.3389/FMICB.2017.02224/BIBTEX

36. Leek JT, Johnson WE, Parker HS, Jaffe AE, Storey JD. The sva package for removing batch effects and other unwanted variation in high-throughput experiments. Bioinformatics. 2012;28: 882–883. doi:10.1093/bioinformatics/bts034

37. Adamer MF, Bruningk SC, Estermann F, Borgwardt KM. reComBat: Batch effect removal in large-scale, multi-source omics data integration. : 14.

38. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research. 2015;43: e47. doi:10.1093/nar/gkv007

39. Smyth GK. limma: Linear Models for Microarray Data. In: Gentleman R, Carey VJ, Huber W, Irizarry RA, Dudoit S, editors. Bioinformatics and Computational Biology Solutions Using R and Bioconductor. New York: Springer-Verlag; 2005. pp. 397–420. doi:10.1007/0-387-29362-0_23

40. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in {P}ython. Journal of Machine Learning Research. 2011;12: 2825–2830.

41. Breiman L. Random Forests. Machine Learning. 2001;45: 5–32. doi:10.1023/A:1010933404324

42. Couronné R, Probst P, Boulesteix A-L. Random forest versus logistic regression: a large-scale benchmark experiment. BMC Bioinformatics. 2018;19. doi:10.1186/S12859-018-2264-5

43. Verikas A, Gelzinis A, Bacauskiene M. Mining data with random forests: A survey and results of new tests. Pattern Recognition. 2011;44: 330–349. doi:10.1016/j.patcog.2010.08.011

44. Bernard S, Heutte L, Adam S. Influence of Hyperparameters on Random Forest Accuracy. In: Benediktsson JA, Kittler J, Roli F, editors. Multiple Classifier Systems. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009. pp. 171–180. doi:10.1007/978-3-642-02326-2_18

45. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods. 2020;17: 261–272. doi:10.1038/s41592-019-0686-2

46. McKnight PE, Najab J. Mann-Whitney U Test. The Corsini Encyclopedia of Psychology. John Wiley & Sons, Ltd; 2010. pp. 1–1. doi:10.1002/9780470479216.corpsy0524

47. Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. New York: Cambridge University Press; 2008. Available: http://www.amazon.com/Introduction-Information-Retrieval-Christopher-Manning/dp/0521865719/ref=sr_1_1?ie=UTF8&qid=1337379279&sr=8-1

48. Lee S, Lee DK. What is the proper way to apply the multiple comparison test? Korean J Anesthesiol. 2018;71: 353–360. doi:10.4097/kja.d.18.00242

49. Junqué de Fortuny E, Martens D, Provost F. Predictive Modeling With Big Data: Is Bigger Really Better? Big Data. 2013;1: 215–226. doi:10.1089/big.2013.0037

50. Hannigan GD, Duhaime MB, Ruffin MT, Koumpouras CC, Schloss PD. Diagnostic Potential and Interactive Dynamics of the Colorectal Cancer Virome. mBio. 2018;9: e02248-18. doi:10.1128/mBio.02248-18

51. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nat Biotechnol. 2017;35: 316–319. doi:10.1038/nbt.3820

52. Feng Q, Liang S, Jia H, Stadlmayr A, Tang L, Lan Z, et al. Gut microbiome development along the colorectal adenoma-carcinoma sequence. Nat Commun. 2015;6: 6528. doi:10.1038/ncomms7528

53. Gupta A, Dhakan DB, Maji A, Saxena R, P K VP, Mahajan S, et al. Association of Flavonifractor plautii, a Flavonoid-Degrading Bacterium, with the Gut Microbiome of Colorectal Cancer Patients in India. mSystems. 2019;4: e00438-19. doi:10.1128/mSystems.00438-19

54. Vogtmann E, Hua X, Zeller G, Sunagawa S, Voigt AY, Hercog R, et al. Colorectal Cancer and the Human Gut Microbiome: Reproducibility with Whole-Genome Shotgun Sequencing. PLoS One. 2016;11: e0155362. doi:10.1371/journal.pone.0155362

55. Wirbel J, Pyl PT, Kartal E, Zych K, Kashani A, Milanese A, et al. Meta-analysis of fecal metagenomes reveals global microbial signatures that are specific for colorectal cancer. Nat Med. 2019;25: 679–689. doi:10.1038/s41591-019-0406-6

56. Yachida S, Mizutani S, Shiroma H, Shiba S, Nakajima T, Sakamoto T, et al. Metagenomic and metabolomic analyses reveal distinct stage-specific phenotypes of the gut microbiota in colorectal cancer. Nature Medicine 2019 25:6. 2019;25: 968–976. doi:10.1038/S41591-019-0458-7

57. Zeller G, Tap J, Voigt AY, Sunagawa S, Kultima JR, Costea PI, et al. Potential of fecal microbiota for early-stage detection of colorectal cancer. Mol Syst Biol. 2014;10: 766. doi:10.15252/msb.20145645

58. Rajan SK, Lindqvist M, Brummer RJ, Schoultz I, Repsilber D. Phylogenetic microbiota profiling in fecal samples depends on combination of sequencing depth and choice of NGS analysis method. PLoS ONE. 2019;14. doi:10.1371/JOURNAL.PONE.0222171

59. McMurdie PJ, Holmes S. Waste Not, Want Not: Why Rarefying Microbiome Data Is Inadmissible. PLOS Computational Biology. 2014;10: e1003531. doi:10.1371/JOURNAL.PCBI.1003531

60. Filzmoser P, Hron K. Correlation Analysis for Compositional Data. Mathematical Geosciences 2008 41:8. 2008;41: 905–919. doi:10.1007/S11004-008-9196-Y

61. Gweon HS, Shaw LP, Swann J, De Maio N, AbuOun M, Niehus R, et al. The impact of sequencing depth on the inferred taxonomic composition and AMR gene content of metagenomic samples. Environmental Microbiome. 2019;14: 7. doi:10.1186/s40793-019-0347-1

62. Rong Z, Tan Q, Cao L, Zhang L, Deng K, Huang Y, et al. NormAE: Deep Adversarial Learning Model to Remove Batch Effects in Liquid Chromatography Mass Spectrometry-Based Metabolomics Data. Anal Chem. 2020;92: 5082–5090. doi:10.1021/acs.analchem.9b05460

63. Dincer AB, Janizek JD, Lee S-I. Adversarial deconfounding autoencoder for learning robust gene expression embeddings. Bioinformatics. 2020;36: i573–i582. doi:10.1093/bioinformatics/btaa796

64. Lotfollahi M, Wolf FA, Theis FJ. scGen predicts single-cell perturbation responses. Nat Methods. 2019;16: 715–721. doi:10.1038/s41592-019-0494-8

65. Arpit D, Aadyot, Bhatnagar, Wang H, Xiong C. Momentum Contrastive Autoencoder: Using Contrastive Learning for Latent Space Distribution Matching in WAE. arXiv:211010303 [cs]. 2021 [cited 10 May 2022]. Available: http://arxiv.org/abs/2110.10303

6. Supplementary Materials

6.1. Tables

6.1.1. Supplementary Table 1: Metadata for the CRC studies that were considered

Study	*# of Samples*	*Male (%)*	*Controls (%)*	*Mean Age (+/- std)*
FengQ_2015 [52]	107	59.81	57.01	67 (9)
GuptaA_2019 [53]	60	50.0	50.0	51 (16)
HanniganGD_2017 [50]	55	56.36	50.91	57 (10)
ThomasAM_2018a [14]	53	67.92	45.28	70 (8)
ThomasAM_2018b [14]	60	65.0	46.67	58 (8)
ThomasAM_2019_c [14]	80	56.25	50.0	61 (13)
VogtmannE_2016 [54]	104	71.15	50.0	62 (12)
WirbelJ_2018 [55]	125	58.4	52.0	60 (13)
YachidaS_2019 [56]	509	58.35	49.31	62 (11)
YuJ_2015 [11]	128	63.28	41.41	64 (9)
ZellerG_2014 [57]	114	50.0	53.51	63 (12)
Total	*1395*	*59.68*	*49.65*	*62 (12)*

TABLE 6-1 METADATA FOR STUDIES THAT WERE SELECTED. EACH OF THE DATASETS IS GIVEN IN REFERENCE TO THE STUDY IN WHICH IT WAS PUBLISHED, NAMED ACCORDING TO THE LEAD AUTHOR AND YEAR, WITH DUPLICATE AUTHORS SUFFIXED. THREE STUDIES FROM THOMAS ET AL. WERE USED, A COLLABORATION WITH 39 MEMBERS. PERCENTAGES WERE ROUNDED TO 2 DECIMAL PLACES, YEARS WERE ROUNDED TO WHOLE NUMBERS.

6.1.2. Supplementary Table 2: Parameter spaces

Name	*Parameter space*
Random Forest	"n_estimators": [int(x) for x in np.linspace(start=200, stop=1000, num=10)] "max_features": ["auto", "log2", 0.2, 0.4, 0.5], "max_depth": [int(x) for x in np.linspace(10, 110, num=11)], "min_samples_split": [2, 5, 10], "min_samples_leaf": [1, 2, 4]
BernoulliNB	"alpha": [0.0001, 0.001, 0.01, 0.1, 1] "fit_prior": [True, False]}
Gradient Boosting Classifier	"n_estimators": [int(x) for x in np.linspace(start=200, stop=1000, num=10)] "learning_rate": [0.1, 0.05, 0.01, 0.005] "max_depth": [int(x) for x in np.linspace(10, 110, num=11)]
KNeighbors Classifier	"n_neighbors": [int(x) for x in np.linspace(3, 20, num=5)], "weights": ["uniform", "distance"], "algorithm": ["ball_tree", "kd_tree", "brute"], "p": [1, 2], "metric": ["euclidean", ssd.braycurtis, ssd.cosine],
MultinomialNB	"alpha": [0.0001, 0.001, 0.01, 0.1, 1] "fit_prior": [True, False]}
SGD Classifier	"loss": ["hinge", "log", "modified_huber", "squared_hinge", "perceptron"], "penalty": ["l2", "l1", "elasticnet"], "alpha": [0.001, 0.01, 0.1, 1.0],
Support Vector Machine	"C": [0.001, 0.01, 0.1, 1, 10, 100, 1000]

TABLE 6-2 THE PARAMETER SPACES OF EACH BINARY CLASSIFIER THAT WAS USED IN BATCH CORRECTION EXPERIMENT. EACH PARAMETER SPACE WAS ARBITRARILY EXPLORED WITH A RANDOMIZED GRID SEARCH. PARAMETERS NOT MENTIONED WERE LEFT AT DEFAULTS.

6.1.3. Supplementary Table 3: Binary Classification Scores

Corrector	*Random Forest Classifier*	*BernoulliNB*	*Gradient Boosting Classifier*	*KNeighbors Classifier*	*MultinomialNB*	*SGD Classifier*	*SVC*
Baseline	0.81 (0.11)	0.50 (0.00)	0.78 (0.10)	0.53 (0.10)	0.65 (0.13)	0.60 (0.09)	0.57 (0.07)
TOOLS							-
ComBat	0.67 (0.19)	0.77 (0.18)	0.76 (0.18)	0.52 (0.11)	-	0.58 (0.13)	0.55 (0.13)
ComBat-seq	0.81 (0.09)	0.82 (0.15)	0.80 (0.08)	0.54 (0.07)	0.63 (0.11)	0.61 (0.09)	0.57 (0.14)
reComBat	0.78 (0.09)	0.74 (0.13)	0.79 (0.13)	0.54 (0.12)	-	0.62 (0.11)	0.57 (0.19)
Limma	0.73 (0.12)	0.66 (0.09)	0.76 (0.11)	0.53 (0.08)	-	0.59 (0.09)	0.54 (0.10)
TRANSFORMATIONS							-
BMC	0.69 (0.18)	0.78 (0.13)	0.70 (0.13)	0.55 (0.11)	-	0.53 (0.08)	0.62 (0.09)
CLR	0.83 (0.11)	0.82 (0.15)	0.82 (0.09)	0.59 (0.06)	-	0.69 (0.07)	0.73 (0.09)
Normalize	0.80 (0.12)	0.77 (0.14)	0.77 (0.11)	0.54 (0.12)	-	0.62 (0.12)	0.61 (0.08)
ILR	0.73 (0.13)	0.58 (0.17)	0.77 (0.10)	0.52 (0.07)	-	0.63 (0.10)	0.64 (0.14)
ILR + Quantile	0.73 (0.13)	0.55 (0.07)	0.78 (0.09)	0.58 (0.05)	-	0.63 (0.10)	0.67 (0.07)
CLR + Quantile	0.58 (0.13)	0.75 (0.16)	0.80 (0.13)	0.59 (0.06)	0.76 (0.15)	0.50 (0.05)	0.50 (0.00)
Quantile (normal)	0.79 (0.13)	0.82 (0.15)	0.78 (0.10)	0.59 (0.06)	-	0.68 (0.09)	0.72 (0.10)
Quantile (uniform)	0.80 (0.12)	0.82 (0.15)	0.78 (0.10)	0.57 (0.08)	0.80 (0.13)	0.68 (0.07)	0.72 (0.13)
ENCODINGS							-
CLR + VarianceThreshold	0.82 (0.12)	0.78 (0.16)	0.81 (0.11)	0.59 (0.06)	-	0.68 (0.09)	0.74 (0.09)
PCA (20)	0.57 (0.10)	0.54 (0.04)	0.54 (0.09)	0.52 (0.11)	-	0.55 (0.07)	0.52 (0.08)
PCA (100)	0.64 (0.09)	0.63 (0.08)	0.63 (0.06)	0.54 (0.05)	-	0.60 (0.09)	0.56 (0.04)

TABLE 6-3 BINARY CLASSIFICATION SCORE OF EACH COMBINATION OF CORRECTOR (ROW) AND BINARY CLASSIFIER (COLUMN), GIVEN AS MEDIAN (IQR) OF PERFORMANCE MEASURED WITH AUROC SCORE MEASURED OVER 11 LEAVE ONE OUT CROSS-VALIDATION RUNS. SCORES ARE VISUALIZED IN SUPPLEMENTARY FIGURE 6-4. FOR EACH CLASSIFIER, SCORES THAT WERE SIGNIFICANTLY HIGHER THAN THE BASELINE ARE BOLDED, WHILE THOSE THAT ARE SIGNIFICANTLY SMALLER ARE ITALICIZED. SIGNIFICANCE WAS TESTED USING SIGNED WILCOXON RANKED SUM TEST, AND P-VALUES WERE CORRECTED WITH BONFERRONI CORRECTION.

6.1.4. Supplementary Table 3: Binary classification significance

Batch correction model	*Median (IQR)*	*Statistic*	*p-value*
Baseline	*0.81 (0.11)*	-	-
TOOLS			
ComBat	**0.77 (0.18)**	**1575**	**1.58E-04**
ComBat-seq	0.81 (0.09)	2916	1.03E+01
reComBat	0.79 (0.13)	2520	1.68E+00
Limma	**0.76 (0.11)**	**1975**	**1.97E-02**
Normalize	0.80 (0.12)	2816	1.43E+01
TRANSFORMATIONS			
BMC	0.78 (0.13)	2626	3.05E+00
CLR	0.83 (0.11)	2726	4.95E+00
ILR	0.77 (0.10)	2077	5.43E-02
ILR + Quantile	**0.78 (0.09)**	**1997**	**2.47E-02**
CLR + Quantile	0.80 (0.13)	2538	1.87E+00
Quantile (normal)	0.80 (0.12)	2830	1.48E+01
Quantile (uniform)	0.82 (0.15)	3027	1.41E+01
ENCODINGS			
CLR + VarianceThreshold	0.82 (0.12)	2754	5.60E+00
PCA (20)	**0.57 (0.10)**	**385**	**1.07E-13**
PCA (100)	**0.64 (0.09)**	**835**	**2.59E-09**

TABLE 6-4 TESTING RESULTS FOR COMPARISON OF THE BEST CLASSIFIERS. A TWO-SIDED WILCOXON RANK SUM TEST WAS PERFORMED, REPORTING THE P-VALUE AND U STATISTIC. P-VALUES ARE REPORTED AFTER CORRECTION (MULTIPLICATION BY THE NUMBER OF CORRECTORS-1). SIGNIFICANT P-VALUES ARE BOLDED FOR COMBAT-INTEGRATED, ILR, PCA (20), AND PCA(100). STATISTIC=THE SUM OF THE RANKS OF THE DIFFERENCES ABOVE OR BELOW ZERO, WHICHEVER IS SMALLER.

6.2. Figures

6.2.1. Supplementary Figure 1: Datasets in the curatedMetagenomicData package

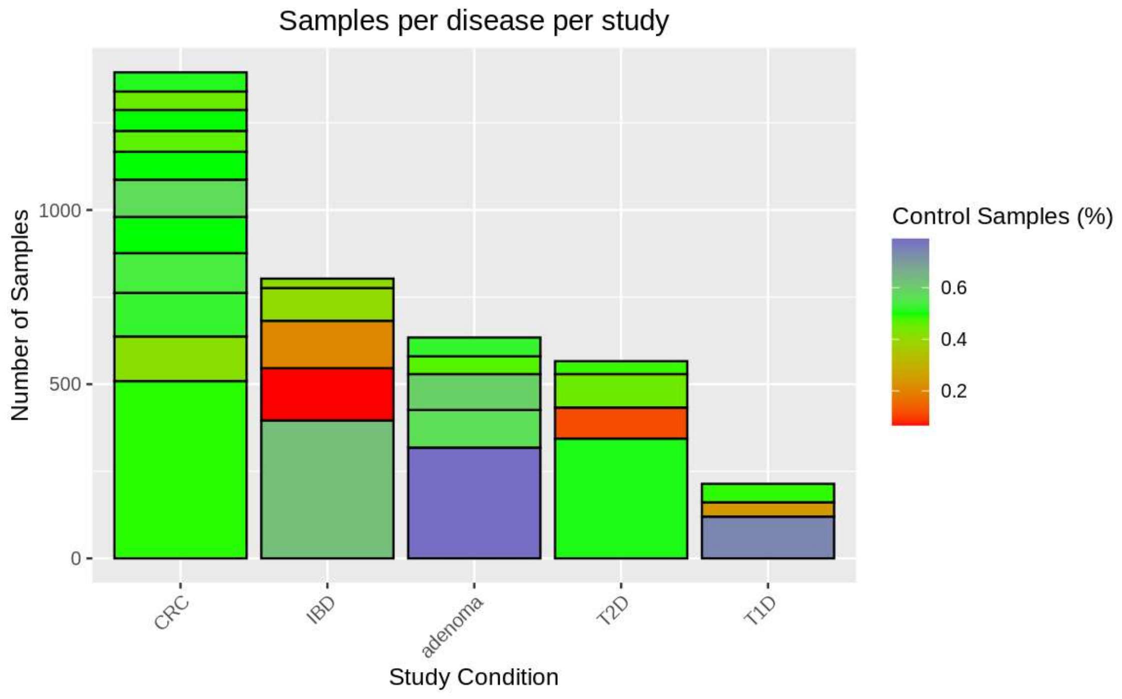

Figure 6-1 Number of samples (from unique patients) for the most populous conditions, as found in 'curatedMetagenomicData.' Each block represents one study with both control and case samples. Color indicates what percentage of samples were from control patients, with a gradient from 0% (only case samples) as red to 100% (only control samples) as blue, with a peek in green at 50%. CRC = Colorectal Cancer, IBD = Inflammatory Bowel Disease, Adenoma = first stage of CRC, T2D = Type 2 Diabetes, T1D = Type 1 Diabetes.

6.2.2. tSNE reductions for each batch corrector

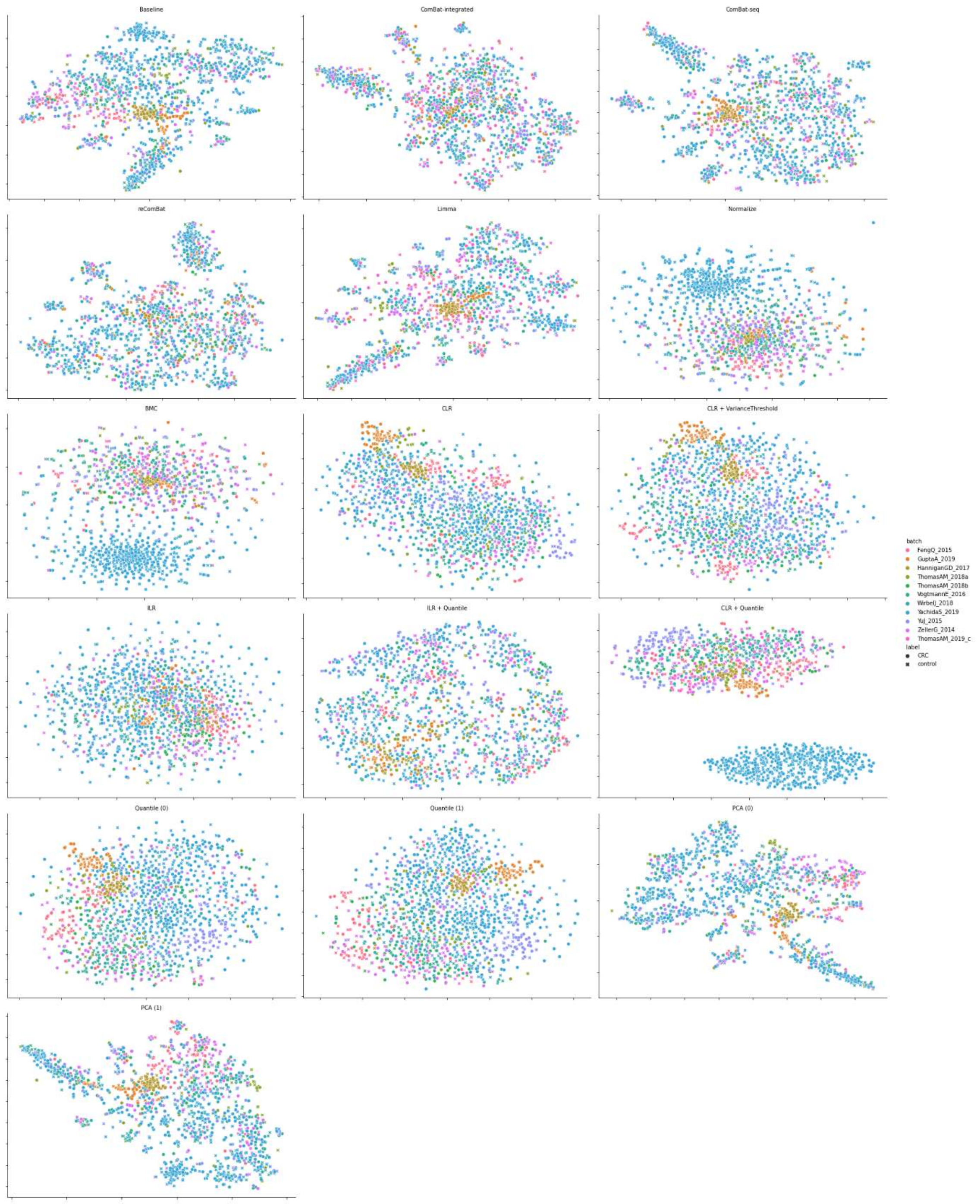

FIGURE 6-2 TSNE REDUCTIONS FOR THE FIRST ITERATION OF THE LODO CROSS-VALIDATION FOR EACH OF THE BATCH CORRECTORS USED IN THE BENCHMARK. SUBFIGURES ARE LABELED ACCORDING TO THE BATCH CORRECTOR WHOSE OUTPUT WAS TRANSFORMED WITH POINTS COLORED BY THEIR ORIGINATING DATRASET AND STYLED ACCORDING TO THEIR LABEL. ELEVEN DATASETS ARE PRESENT IN EACH SUBFIGURE WITH YACHIDA ET AL. MOST PROMINENT, BEING THE LARGEST (N>500).

6.2.3. UMAP reductions for each batch corrector

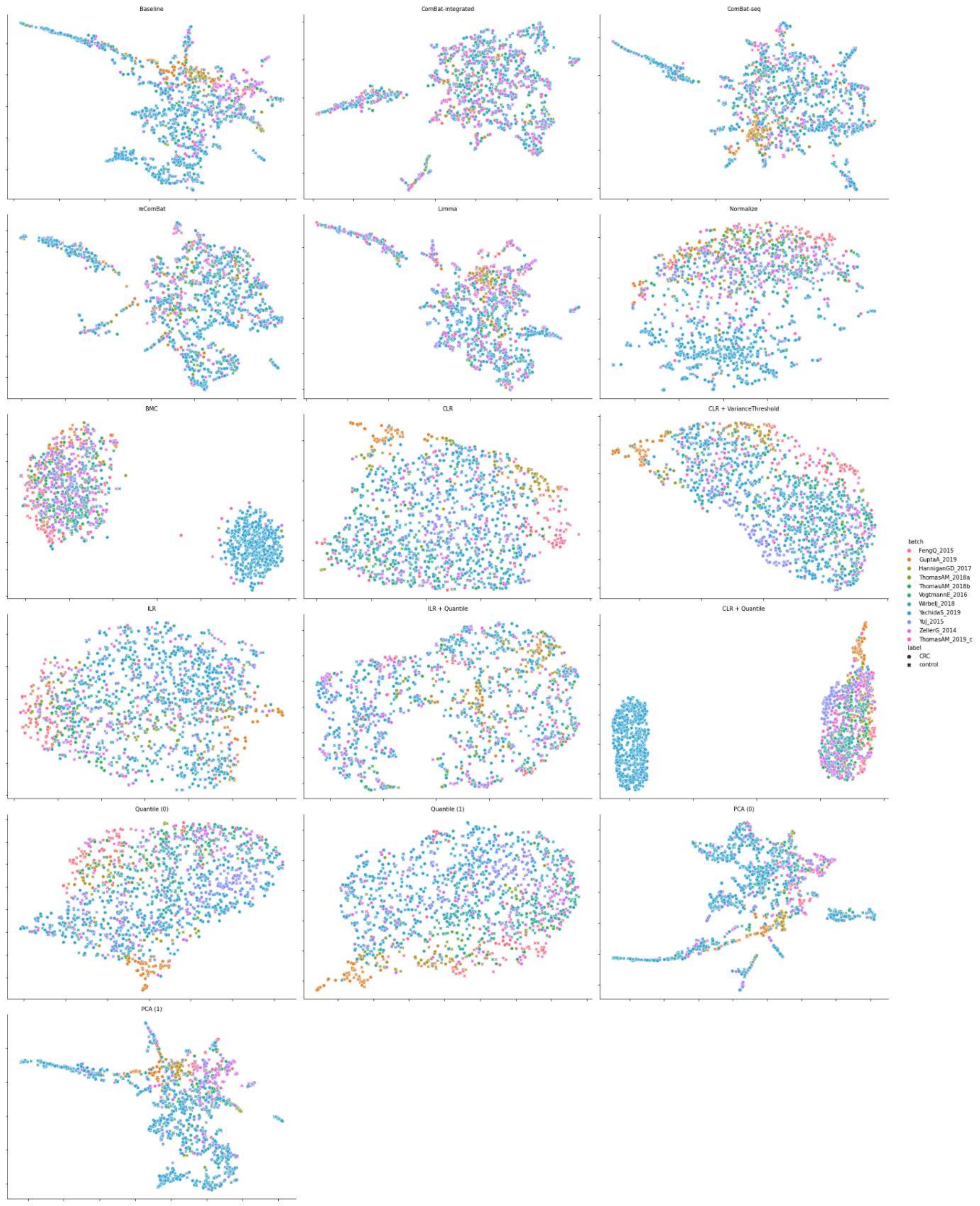

FIGURE 6-3 UMAP REDUCTIONS FOR EACH OF THE TRANSFORMATIONS USED IN THE BENCHMARK. EACH AXIS CONTAINS THE UMAP TRANSFORMATION FOR ONE OF THE TRANSFORMATIONS, WITH POINTS COLORED BY THEIR STUDY AND STYLED ACCORDING TO THEIR LABEL. SOME TRANSFORMATIONS CLUSTER SOME STUDIES TOGETHER, AS CAN BE CLEARLY SEEN FOR 'BMC,' 'NORMALIZE,' AND 'CLR + QUANTILE', WHERE THE STUDY 'YACHIDAS_2019' FORMS A SEPARATE CLUSTER.

6.2.4. Binary classification performances for each batch corrector

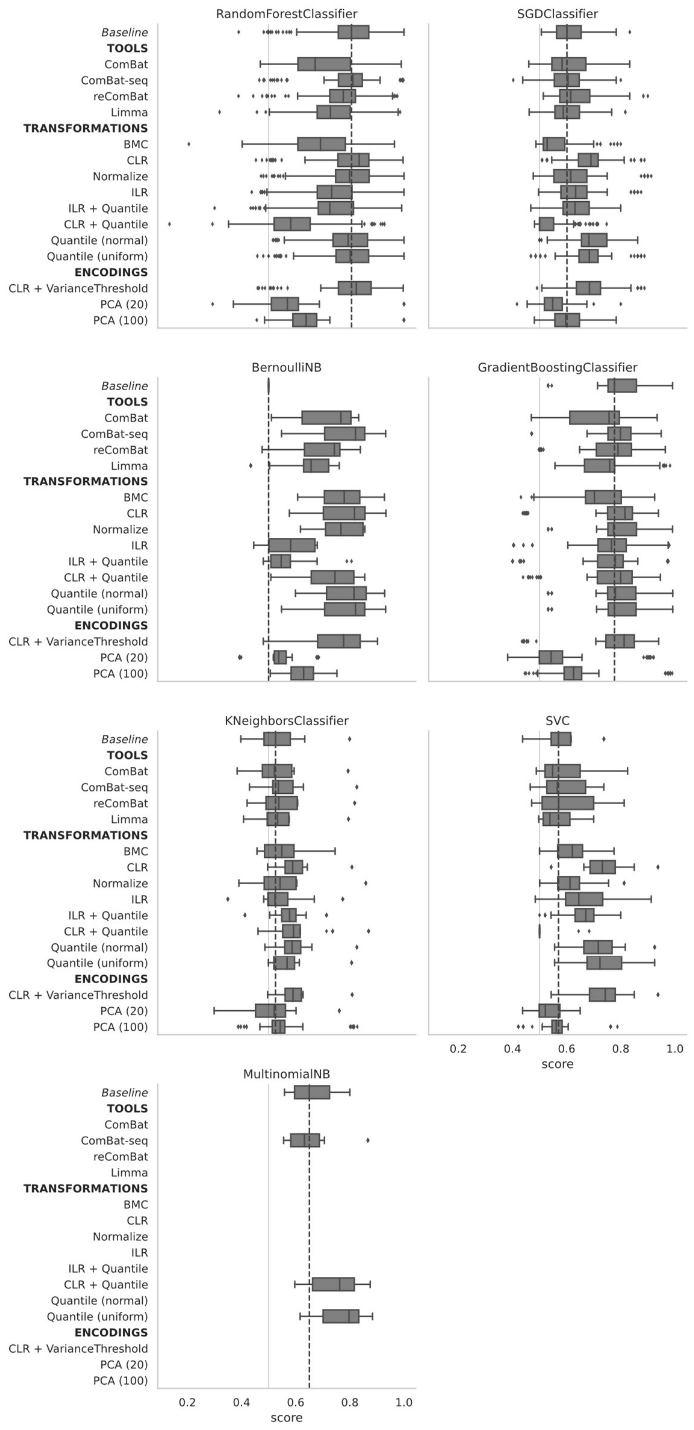

FIGURE 6-4 COMPARISON OF BINARY CLASSIFICATION PERFORMANCE WITH LEAVE ONE DATASET OUT (LODO) PRINCIPLE. ALL AXES ARE THE SAME, SETTING OUT BATCH CORRECTOR OVER THE AUROC SCORE THAT WAS OBTAINED. BOX PLOT OF QUARTILES IS PLOTTED, WITH WHISKERS EXTENDING TO NEAREST POINT WITHIN 1.5 IQR. FLIERS ARE DRAWN FOR OUTLIERS. EACH OF THE PLOTS IS TITLED ACCORDING TO THE BINARY CLASSIFIER THAT WAS EVALUATED, WHILE BATCH CORRECTORS ARE KEPT IN THE SAME ORDERING AND GROUPING AS IN THE REST OF THE PAPER . MULTINOMIAL NB CRASHED ON HANDLING NEGATIVE VALUES, REDUCING THE NUMBER OF BATCH CORRECTORS WHOSE OUTPUT IT COULD HANDLE .

6.3. Datasets

6.3.1. Selecting data

In this supplementary section we outline the procedure that was used to select the datasets. For this, we used the ‘curatedMetagenomicData’ package that was made available on CRAN, which had a selection of 20,283 samples taken from 86 studies of shotgun metagenomics data. In addition, this package used the same procedure on each of the raw reads by running MetaPhlan3 with default settings, and had curated the metadata of each of the datasets, which is notoriously rare within the field.

After having obtained access to all the datasets available of the curatedMetagenomicData dataset and performing some initial exploration, we filtered out all but the first (and only) sample of a patient, to prevent longitudinal samples from influencing our results. Then, we calculated the percentage of samples within each study that were diseased and selected the studies that had at least 5 percent control/healthy samples and 5 percent case/diseased samples, and had at least 40 samples. We then grouped these studies by the diseases that they investigated and graphed each study in a stacked bar plot showing the number of samples for each disease, which can be viewed in Supplementary Figure 1.

From this graph it was clear to us that Colorectal Cancer (CRC) studies were both most balanced as well as the largest in total within the scope that we had selected. We thus selected these datasets to perform our analysis on and downloaded them, again making sure that we only kept a single sample per patient. Apart from taxonomic data, the package also had many other datatypes available, like gene families and pathway coverage. We elected to focus solely on the taxonomic abundance and specifically the abundance of particular species, both to simplify the project and because this had been found to perform best for classification for many cases [4].

The code for analyzing the curatedMetagenomicDataset is provided in our rough work repository: https://github.com/AbeelLab/ngs-batch-evaluation-rough

7. Additional materials

7.1. Additional Data Analysis

Before trying to understand the advantages and drawbacks of batch correction methods, it is appropriate to consider what the input looks like. Taxonomic count data obtained from metagenomic sequencing has a number of characteristics that make it harder to apply traditional techniques.

Read counts are correlated. While traditional batch correction techniques assume that microbial species are independent, sequence data represents the abundance of corresponding microbial communities [21]. This can create problems as while the sum of the counts is some constant value, standard statistical methods assume no such constraint and result in spurious values [9].

Uneven sequencing depths can have an unmapped technical influence, as some low abundance bacteria are not measured at lower sequencing depths [58]. Statistical comparison of samples can be hindered by these differences in depth [59].

Sparsity and overdispersion. The many zeroes in taxonomical microbiome data have two possible sources. They may come from an actual absence within the sample, a 'structural' zero, or they may come from under sampling of the sample, a 'sampling' zero. In addition, the counts within data are widespread in their value, making conventional methods of batch correction less suited [9].

Compositionality. When sequencing the microbiome, the samples are by necessity a small subset of the entire microbiome, and cannot inform on the absolute abundances of the bacterial population. Instead, relative abundances are obtained in the form of counts, which make it harder to perform many statistical analyses [35,60,61].

To confirm that the taxonomic OTU counts suffer from these same issues, we visualized the dispersion of its data in Supplementary Figure 7-1, wherein it is clear that the variance of the count data is nowhere near its mean, indicating overdispersion. This means that we cannot apply some of the typical count model data analysis, which relies on undispersed data.

To show the sparsity of the data, we counted the percentage of features that were non-zero for each of the samples in all the studies, which we show in Figure 7-2. It can be seen that only 10-20% of all counts is non-zero.

On a per-sample basis, we visualized the saturation of features by going over each sample in random order 10 times and counting how many features had been non-zero (cumulatively). The resulting graph is shown in **Error! Reference source not found.** The sharpness of the slope when it cuts can be seen as an indication of how many samples the data needs to be able to learn fully. The slope being sharp for the smaller studies indicates they lack the amount of samples to be able to fully learn about its features.

7.1.1. Overdispersion of taxonomic count data

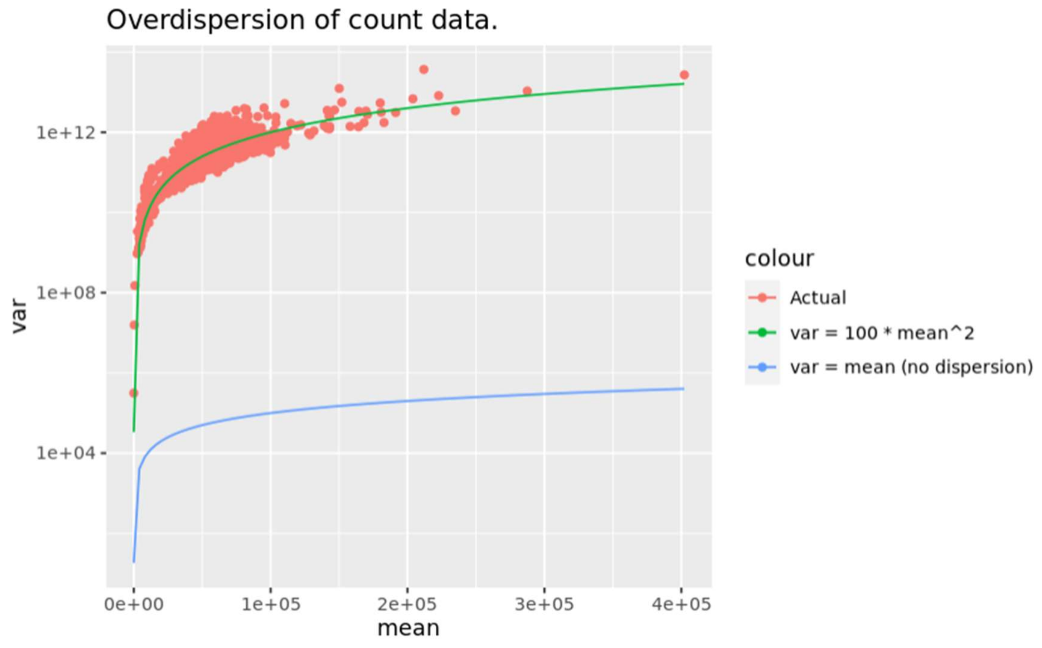

Figure 7-1 Dispersion of the datasets shown by plotting the variance over the mean for the features in the dataset. Each point represents the mean and variance of a single feature in the raw taxonomic count data. Both axes are logarithmic base 10. For illustrative purposes two functions for the variance are plotted, one linear and one quadratic.

7.1.2. Sparsity of the datasets

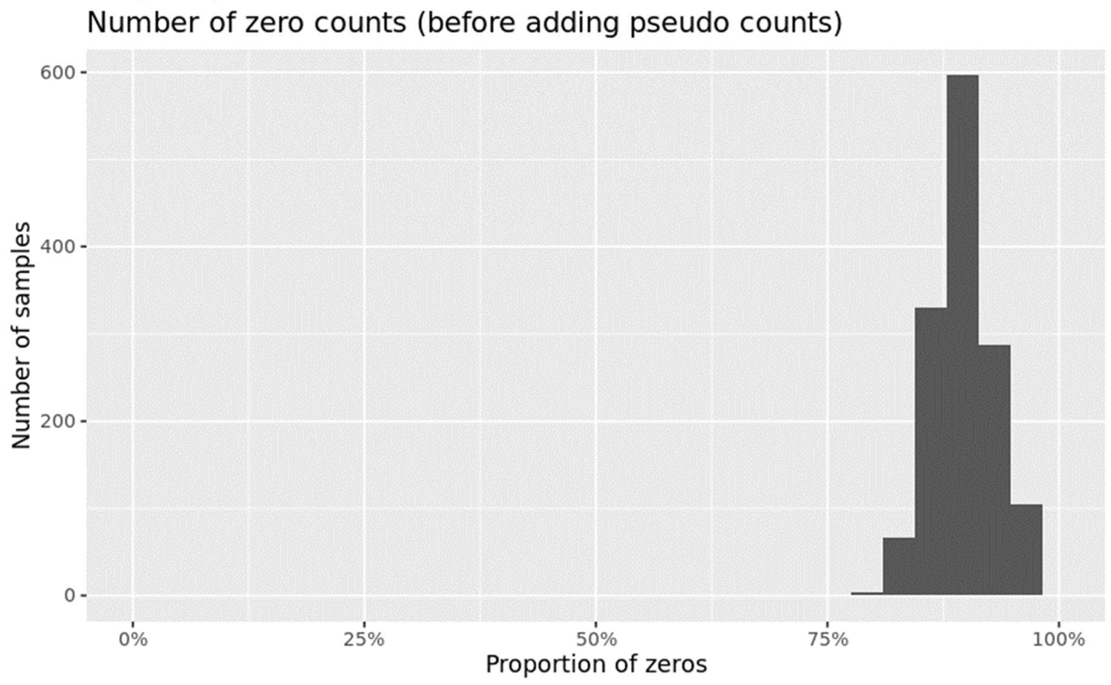

Figure 7-2 Sparsity of the datasets visualized through a histogram counting the number of samples that had a given proportion of features which were zero. None of the samples had a proportion of zeroes that was below 75%, so for each sample at most 25% of the features had a non-zero value. All samples had at least a few features that were non-zero. Data was generated without adding pseudocounts.

7.1.3. ECDF of species for each study

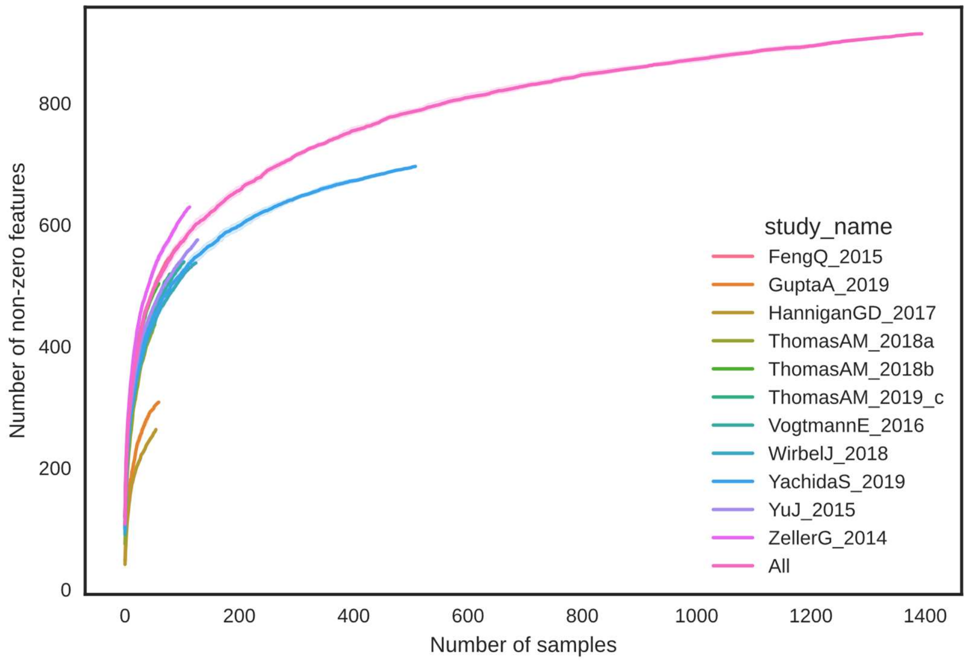

Figure 7-3 Graph showing saturation of features. For each study, and the combination of all studies ('All'), in 10 random shuffles, the number of features that were non-zero was counted cumulatively. The average number of features encountered at the ith sample for a study was plotted on the line, with the shaded area showing the confidence interval. Yachida et al. has a total of 509 samples with 696 nonzero features. Line that goes farthest is for all studies together. Slope of line when it reaches its last sample is sharp for smaller studies, smaller for Yachida et. al. and smaller still for all studies combined .

7.2. Alternative data sources

In addition to the curatedMetagenomicData package, we also explored using MGnify and downloading and processing datasets ourselves.

MGnify is a microbiome analysis resource provided by the European Bioinformatics Institute (EMBL-EBI) to which metagenomics data is submitted and automatically processed using their pipeline. With 4,294 studies submitted at the time of writing, 928 of which are related to the digestive system of the human microbiome, this was an avenue worth exploring. The increasing usage of this and other similar all-in-one platforms like MGRast, Qiime2, and Galaxy it is increasingly relevant to explore the results that these produce.

To investigate the usage of this platform we queried its API for the studies that were relevant, eventually narrowing it down to approximately 50 studies that fit our criteria of having both case and control samples of the human gut microbiome, which we then manually combed through for the treatment variables, for each of the selected studies.

After this process, we aggregated the data, counting the number of samples for each treatment (control-case) within each of the studies. By selecting those treatment covariates that had more than 50 samples, we filtered the number of usable studies to 20 as shown in Figure 7-4, which unfortunately did not show anywhere close to the balance that we desired for our classification task. As such, we elected to put the effort on hold and instead focus on the studies made available through the curatedMetagenomic Data package.

7.2.1. MGnify study data

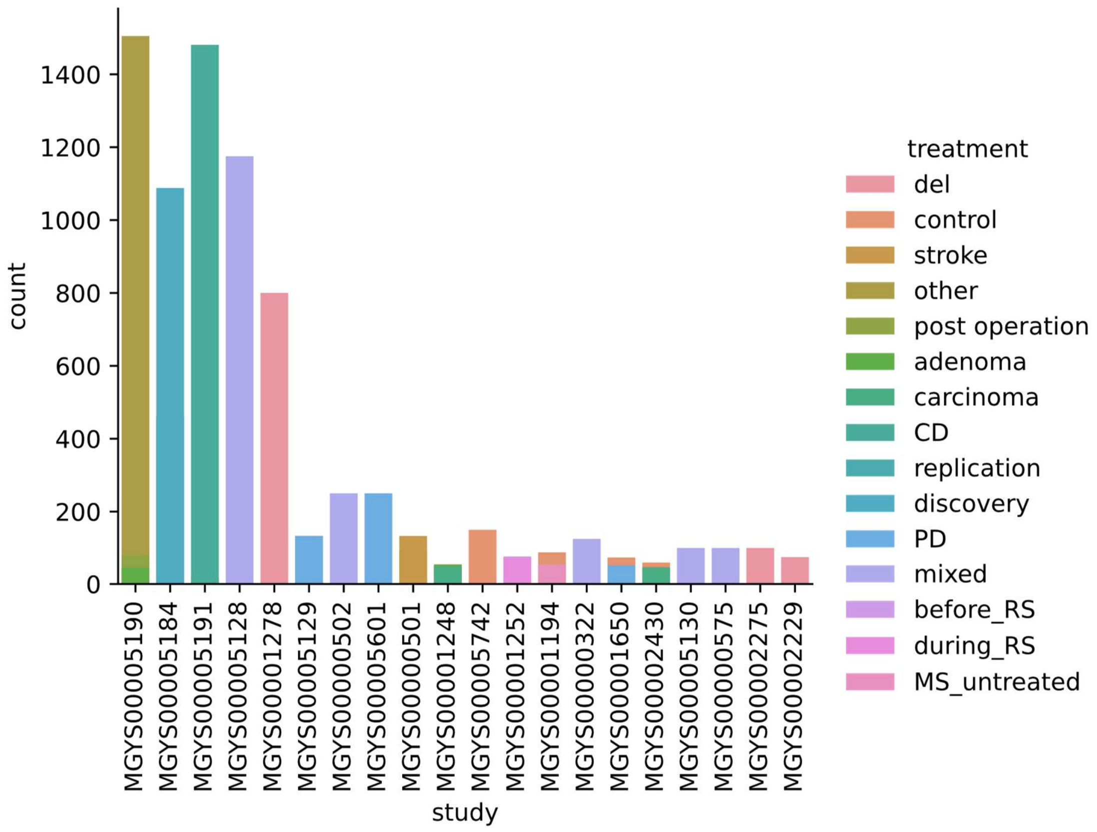

FIGURE 7-4 THE NUMBER OF SAMPLES FOUND FOR A NUMBER OF SELECTED STUDIES FROM THE MGNIFY DATABASE, COLORED ACCORDING TO THE TREATMENT VARIABLE ASSIGNED AS METADATA. METADATA FOR EACH OF THE STUDIES WAS DOWNLOADED AND PROCESSED MANUALLY TO CLEAN UP THE LABELS. MIXED, OTHER, AND DEL INDICATE THE METADATA PROVIDED BY THE STUDY WAS INSUFFICIENT TO BE ABLE TO DISTINGUISH CONTROL AND CASE SAMPLES. DISCOVERY AND REPLICATION DID NOT DISTINGUISH FURTHER TO CONTROL AND CASE. PD = PARKINSON'S DISEASE, MS = MULTIPLE SCLEROSIS, RS = RESISTANCE STARCH SUPPLEMENT. CD = CROHN'S DISEASE. POST OPERATION INDICATES SAMPLES FROM PATIENTS THAT WERE TREATED FOR THEIR CARCINOMA.

7.3. Performing own taxonomic read assignment

In tandem with our efforts to build the pipeline for the analysis we also worked on downloading and processing a number of datasets, starting with the datasets found through the curated package, we extracted the ECBI ids that were available, downloading the data for each using the prefetch and fasterq-dump tools provided by the SRA toolkit. The data was then trimmed using the Trimmomatic package and kraken2 was run on the data, using the standard dataset.

We elected not to use the data thus obtained for several reasons; firstly, not all studies had accession IDs available, and some of the samples referenced multiple accessions making it unclear which one would have to be used. Lastly, the Kraken tool is used for general analysis and does not provide a human gut microbiome specific dataset while MetaPhlan3 is the latest of an excellent line of tools that is specialized in human gut microbiome, which makes the dataset more fitting for the analysis of the samples that we were focused on.

7.4. Building an AutoEncoder

In addition to trying out many of the algorithms already available and proven to work to some extent on the data we had available, we also investigated the potential of developing our own algorithm. As autoencoders have been shown to work for metabolomic and single cell RNA sequencing [62–64], they were chosen as avenue for exploration. In addition we were fascinated by the recent development of contrastive autoencoders, also known as Wasserstein Autoencoders (WAE) [65].

This algorithm works by learning to distinguish pairs of samples from each other that are either from the same batch or from different batches, and have either the same or different labels. It first encodes both samples using the same hidden layers towards two parts, z_1 and z_2, then calculates a label loss aimed at minimizing the distance between parts with the same label and maximizing the distance between samples with different labels. Then, a domain loss is calculated simply aimed at minimizing the distance between samples of the same batch. Lastly, the parts are decoded and the reconstruction loss is added.

Unfortunately we did not manage to get the encoder to converge on anywhere close to the performance desired within the timeframe allocated towards the endeavor, which led to the decision to leave its results out of the main results of the paper.

The code for creating the samples, setting up the autoencoder, and training it on the metagenomic data available is freely available on Deepnote[1].

[1] https://deepnote.com/workspace/pluriscient-9a8d4768-9ead-49ef-a014-ce66d9dcda06/project/Armans-AutoEncoder-b2a5f14d-9e6c-4748-809c-310f1065eaa8/%2Fnotebook.ipynb

7.5. Code repository

The codebase that has been developed over the course of the past months can be accessed through https://github.com/AbeelLab/ngs-batch-evaluation, of which I have attached the root README as well as the one that can be found in the src/pipeline directory, to illustrate the ease with which the results can be reproduced.

7.5.1. NGS Batch evaluation

Welcome to the repository associated with the thesis "BATCH CORRECTION OF TAXONOMIC DATA OF THE HUMAN GUT MICROBIOME FOR GENERALIZATION OF CASE-CONTROL CLASSIFICATION"

This repository contains all the code and data necessary to perform a full reproduction of the figures shown in the paper.

It has the following structure

- src
  - `pipeline`: the pipeline used to perform the batch correction and train each of the classifiers
    - `README.md`: Details on how to configure and run the pipeline
  - `visualizations`: The notebooks used to create each of the visualizations
    - `common.py`: Common functions used by each of the sections, including the ordering of the batch correctors
    - `section-1.ipynb`: The code for reproducing the figures in subsection 1 of the results
    - `section-2.ipynb`: The code for reproducing the figures in subsection 2 of the results
    - `section-3.ipynb`: The code for reproducing the figures in subsection 3 of the results
- input
  - `CRC_studies.csv`: the input data, taxonomic read counts obtained from the curatedMetagenomicData database
  - `scores.csv` and `feature.*.csv`: output data of the results of the pipeline

7.5.2. NGS batch correction evaluation pipeline

Requirements

- Java 11
- Nextflow
- Docker or Singularity
- Conda for some of the postprocessing

Quickstart

To quickly run the pipeline with its current configuration on your local machine:

```
    nextflow run ./flow.nf -profile docker --base_dataset
$(pwd)/../../input/CRC_studies.csv
```

And start waiting!

Configuration

The configuration of the pipeline can be found in the `nextflow.config` file. Most configuration can be done through editing the `params` block:

- Params
    - label_column: column with label of interest for the classifiers (can only for binary classification)
    - batching_column: column with batch assignment for each sample
    - feature_column_prefix: prefix that ALL feature columns should have
    - base_dataset: location of the input dataset
    - out_dir: output directory
    - random_seed: *Not completely implemented* seed for randomness
    - split_modes: How to perform the cross-validation, currently either "L1SO" = "Leave one dataset out" or L2SO = "Leave two datasets out"

    - iterations: iterations of cross-validation, should be 1..Nk where N where is number of batches and k the number of iterations of cross-validation
    - correctors: batch correctors to implement
    - models: binary classifiers to train
    - pseudo_count: what count to add to all numbers
    - max_cpus: no. of cpus available in total
    - max_mem: size of the memory available

Binary classifiers

Most any binary classifier from the scikit learn library can easily be used, and any other that implements the same interface. The hyperparameter space does not need to be configured within the `bin/sklearn_trainer.py` file, wherein you need to make sure that

- The classifier is present in the `CLASSIFIERS` list as callable
- The classifier is present in the `PARAMETER_GRID` dictionary as dictionary holding parameters to the callable that can be varied between

Batch correctors

To configure batch correctors individually you can find them in the `bin/ba(r)_*.(py|r) files, named consistently. To add a new one

- Create the appropriate file in the bin directory
- Chmod it to be executable
- Within `flow.nf`, Add it to the massive switch of the `corrector_code_both` function (usually copy pasting the previous entry and changing the target is enough)
- Add any additional requirements to `env.yaml` or `containers/main.Dockerfile`

Post processing

After the nextflow run has completed, a number of steps still need to be completed so that the figures can be created.

1. From the pipeline directory, run `python postbin/run-r-collections.py` and wait for completion

2. Run `python postbin/run-r-collections-collect.py` and wait for completion
3. Run `collect_dec.py`
4. Move the files in the root of the results directory to the `input` folder for the figure analysis.