Jennifer Neumaier

Institute of Genomics, University of Tartu

Master Thesis

*Benchmarking machine learning performances with compositional* *data*

Table of Contents

[Introduction 2](#_Toc103902902)

[Goal 2](#_Toc103902903)

[Background 2](#_Toc103902904)

[Characteristics of Compositional Data 3](#_Toc103902905)

[The Simplex Space 5](#_Toc103902906)

[Mapping the Simplex Space into Euclidean Space 7](#_Toc103902907)

[Log-Ratio Transformations 8](#_Toc103902908)

[Compositional Data in Machine learning 11](#_Toc103902909)

[CODACORE 12](#_Toc103902910)

[Implementation 13](#_Toc103902911)

[Methodology 16](#_Toc103902912)

[Pre-Processing 16](#_Toc103902913)

[Cleaning and filtering 16](#_Toc103902914)

[Imputation 16](#_Toc103902915)

[Transformations 17](#_Toc103902916)

[Machine LEarning models 17](#_Toc103902917)

[Data Leakage in Transformations 19](#_Toc103902918)

[Results 20](#_Toc103902919)

[Holdout vs 80/20 20](#_Toc103902920)

[Leaky Preprocessing 20](#_Toc103902921)

[Colorectal Cancer Dataset (CRC) 22](#_Toc103902922)

[GLMNET 22](#_Toc103902923)

[Polycystic Ovary Syndrome Dataset (PCOS) 24](#_Toc103902924)

[GLMNET 24](#_Toc103902925)

[xgboost 26](#_Toc103902926)

[Estonian Biobank Dataset (EstMB) 27](#_Toc103902927)

[Discussion 28](#_Toc103902928)

[Publication bibliography 30](#_Toc103902929)

[Supplementary 32](#_Toc103902930)

[Influence of Transformation on Holdout vs. 80/20 32](#_Toc103902931)

# Introduction

## Abstract

Machine learning in microbiome studies is widely used and the interest is growing. However, there is no universal understanding of the algorithmic approaches that can best utilize the information present in the microbiome data. Thus, this is an interesting and widely discussed topic that can have a great impact on the potential applications leveraging microbiome data. A key topic in microbiome research is the sample space of the input data. The sequencing data appears as count data, but, only relative abundance of the microbial features can be observed, commonly called “compositional data”. Thus, transforming the read counts to relative abundances is usually the first step and machine learning methods are usually applied on relative abundances. However, relative abundances raise several limitations, which can have an impact on the performance of the prediction models. Therefore, log-ratio transformations are a proposition made by several studies now, however their impact on machine learning performances has never been tested in large-scale studies. The goal of this benchmarking project is to rectify that and conduct several machine learning models under several log-ratio transformations in comparison to *CoDaCoRe*, an algorithm specifically made with microbiome analysis in mind. This way it will become clearer if a scientist should make the effort in learning about machine learning methods, when automated algorithms perform well enough, and no heavy prior machine learning knowledge is necessary.

## Background

Working with mathematical concepts is always a bit out of the comfort zone for most biologists. Unfortunately, with technical improvements and big data encroaching in our field, and statistical methodology being an essential part in data analysis, ignoring mathematics is just not an option. One concept that became increasingly more important in sequencing data analysis is the concept of “compositional data”. Several papers (Greenacre et al. 2021b; Gloor et al. 2017; Quinn et al. 2018) made it abundantly clear that sequencing data is of compositional nature, which means it has different mathematical characteristics than other data types. Furthermore, as machine learning concepts become more widespread and useful, their performance in combination with compositional data and its necessary transformations have not been fully analyzed.

The goal of this master thesis is an attempt in making the information around compositional data more approachable, summarizing the achieved solutions, and in a practical part, trying to assess if these achievements are also applicable when combining compositional data and machine learning concepts.

As this master thesis uses microbiome sequencing data and was created in a microbiome research group, this text will mostly focus on this field and its papers. However, all results are applicable to other high-throughput sequencing data, as well as any data that is in some way confined by an arbitrary sum. Such data is found for example in geochemistry, ecology, sociology, political sciences, etc., and therefore ultimately spans the problematic into various different fields (Greenacre et al. 2021a).

### Characteristics of Compositional Data

In order to define and illustrate the concept (and problems) of compositional data, let’s assume a classical biological example. The following Figure (1A) shows two different ecological fields: A and B. In field A, four rabbits, seven birds, eight bees and one wolf have been counted, whereas field B contains two rabbits, four birds, four bees and one wolf. It becomes clear that, as similar as the diversity may be, the fact that field B seems to have only half of the population of field A, is already valuable information in itself. The total counts per field can be preserved in our data collection and therefore, the absolute count of each organism in this field matters.



Figure : Information Loss of Normalized Data

(A) Illustration of the number of animals found in two different samples. Field A contains four rabbits, eight bees, seven birds and one wolf, whereas field B contains two rabbits, four bees, four birds and one wolf. In (B) the absolute counts have been plotted as a stacked bar plot, with each animal in a different color. (C) shows the stacked bar plot as normalized counts, e.g., percentages.

When using absolute counts, the difference between both fields is easily visible (1B). However, when we really want to compare both fields, we need to transform the samples to a common scale. This is called normalization and we can see the effect in (1C). As soon as the data is normalized, the particular information of absolute counts gets lost. When collecting ecological data ourselves, we can preserve the fact that field B only contained 11 individuals and field A contained 20, by saving that number somewhere on our Excel sheet. However, the problem with sequencing data is: we get the data in the form of 1C.

To demonstrate how the method of using a sequencing machine cannot preserve absolute counts, imagine the following situation: We want to sample field A multiple times a day, but in order to be more efficient, we buy a machine to do the counting for us. Three times a day this machine transmits the number of all the different animals coming to this field. However, this machine has one flaw: it can only count to 20. As soon as the 21st animal on this day comes to the field, it is just simply not counted.

This ultimately means, that the overall number of 20 carries no meaning. Every sample has this exact total number, so it carries no valuable information. Of course, a limit of 20 is weird for us to understand, but sequencing machines do the exact same thing. They are limited in their capacity on the flow cells and even the biggest sequencing machines could never fully sequence the entirety of the organisms RNA contents. And not only the sequencing machines, but the whole RNA-Seq procedure limits the total number of sequences measured. The total number of sequences measured by sequencing machines ultimately depends on the *chemistry of the assay*, not the input material (Quinn et al. 2018).

The consequence of this sampling problem is, that we have to accept the fact that the sum of counts in sequencing data are irrelevant. This leads us to the concept of “Compositional Data”, first introduced by John Aitchison (Aitchison 1982), roughly 40 years ago. Because thankfully, we can still use sequencing data. We just have to adjust for the fact that the absolute counts are non-informative (Quinn et al. 2018; Greenacre et al. 2021b). We can instead use relative abundances, or the proportions between features in a sample.

### The Simplex Space

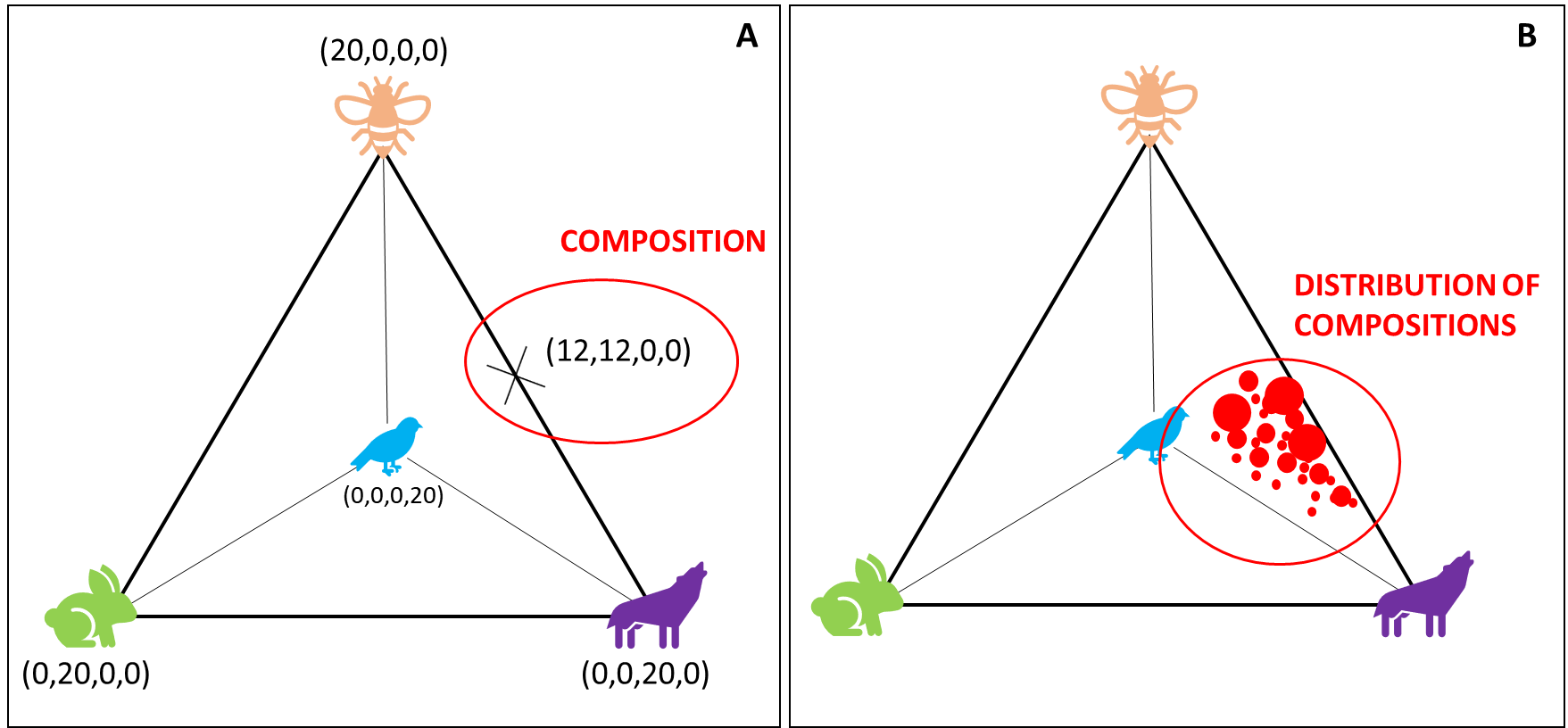
As mentioned, absolute counts in compositional data are irrelevant and only relative abundances are of interest. This puts the data in the so-called “simplex space”, instead of the for us more common Euclidean space. The following Figure 2 shows how the data from field A would look like in the simplex space:

Figure : Biological Example in the Simplex Space

Assuming the collected ecological data from Figure 1 is compositional, puts it in a S3-Simplex space. Geometrically, a tetrahedron is created with all different components (here animals) placed on the four corners of the polytope. A composition is one possible combination of components confined in the simplex space.

We stick with our ecological example and place all our animals as one corner in a geometrical space. With four features, we are able to create a 3-Simplex and a geometric figure called tetrahedron (otherwise called a pyramid).

We use our flawed machine, and one day, we sample 20 rabbits in field A. This would lead to a point in the simplex space that sits directly in the left corner, with the coordinates (0,20,0,0), because we only have rabbits, no other animal. Another day, we sample only 20 bees, and no other animals, then we would find our data point where the bee is, very at the top. Marked in red is a sample where the machine counted 12 bees, 12 wolves and no bees and no rabbits. Every sample round produces one “composition” and the examples show, that the distance between any two variables is sensitive to the presence or absence of other components (Quinn et al. 2018). If a composition is moved from one corner of the animal-simplex, it directly influences the other values in the composition. Consequently, that makes all variables *mutually dependent* on one another and leads, amongst other things, to problems in our assumptions about statistical testing. In statistical literature this data is also called “spurious” because it appears as if the data points have a causal relationship. When a composition is moved from bees in the direction of wolves it seems like there is a causal relationship because the increase in the number of wolves, directly decreases the number of bees.

To describe this a bit more mathematically, the problem described above is formally known as “the negative bias problem” (other names are also the constant-sum problem, the closure problem, or the null correlation difficulty) (Aitchison 2003), which is the main reason why we have a dependency problem. When the sum of a component is constant, then it can be mathematically proven that the covariance between any two compositions equals 0. This has the consequence that some variances would be negative, which is problematic, as variances are always positive. Therefore, negative covariances are presupposed by the limitation of the sum, instead of produced by stochastic factors (Pawlowsky-Glahn and Egozcue 2016; Aitchison 2003).

Additionally, it is commonly assumed - and all experiments are created to accommodate these assumptions – that data is collected IID: independent and identically distributed. The IID assumption is important for e.g., the central limit theorem, Markov sequence, hypothesis testing in general and of course machine learning. Having such an obvious violation in compositional data can have serious consequences on the reproducibility of results. In life sciences, count data are usually modelled using the Poisson distribution or negative binomial, because using anything else would imply that negative and non-integer counts would exist, which is biologically not feasible (Quinn et al. 2018).

Using any form of statistical test or machine learning tool seems redundant, as a type two error is almost preconditioned, and we easily would make false assumptions about the correlation of the data. Thus, a correct handling of compositional data and the simplex space is not optional (Gloor et al. 2017).

### Mapping the Simplex Space into Euclidean Space

The difficulty of confined data points has already been commented on by Pearson (1897) in the context of spurious correlations and has been taken up by Aitchison 1982 in an attempt to overcome the “bounded sum problem”.

In our ecology sampling, we could easily overcome the simplex by e.g., normalizing to a field size from the start, to preserve indirectly an information about the total number of animals.

Similarly, it has been tried for sequencing data to calculate an “effective library size” and to recover this way the original scale of data. For that, normalization methods like trimmed mean of M-values (TMM) have been introduced, as well as RPKM and TPM (Quinn et al. 2018). However, all of those methods involve rescaling counts by the library size and these normalizations come with the drawback that some of these methods are sensitive to the removal of low abundant counts, as well as to data symmetry (Quinn et al. 2018).

Furthermore, Aitchison already criticized very early that there is no “magic to open up closed data” (Aitchison 2003), which is what normalization tries to do. Moreover, since information provided in compositional data is essentially about ratios of the components, it seems logical to also think in terms of ratios. Thus, the only way forward is to transform the data in a way that allows us to use it with Euclidean space rules, first by Aitchison 1982 with several logistic transformations proposed to produce “transformed-normal” models, and later with the definition of the Aitchison geometry (Pawlowsky-Glahn and Egozcue 2001). The general idea is, that the simplex space is endowed with a Euclidean space structure, which has several mathematical advantages: if one can map the simplex space into Euclidean space, then all advantages of the Euclidean space can be accessed, i.e., orthogonal projections are possible, the concepts of linear combination, linear dependence, Euclidean distances, as well as all the typical geometrical elements are available (Pawlowsky-Glahn and Egozcue 2016).

Building on top of the Aitchison geometry, methods of analyzing compositional data were proposed by Mateu-Figueras et al., (2011) with the “staying-in-the-simplex” approach or Greenacres (2017) “pragmatic approach”. In this master thesis, mentioning these methods is as far as I will go here, because they require a technical understanding of the algebraic-geometric structure of the simplex. Here, I will focus more on log-ratio transformations, as they have been more heavily favored in the last decades due to their practicability (Greenacre et al. 2022).

# Log-Ratio Transformations

There are several types of log-ratios, which were proposed of the last 40 years, and I want to take the time and introduce them. Some more in detail than others, as not every log-ratio transforms the data perfectly and it is important to point out here, that there are still ongoing discussions about which log-ratio transformation is preferrable over the other in terms of accuracy, complexity, and interpretability (Greenacre et al. 2022; Quinn and Erb 2020; Rivera-Pinto et al. 2018).

In general, all log-ratio transformations capture the relationship between the features in the data set and taking the logarithm of these ratios makes the data symmetric and linearly related. It moves the simplex into real space and imparts key properties on the data set: scale invariance (compositions do not change with e.g., sequencing depth), perturbation invariance (i.e., converting a composition between equivalent units will not change the results), and permutation invariance (i.e., changing the order of the components within a composition will not change the results).

Two more important properties exist that are transformation-specific: sub-compositional coherence (i.e., scientists A and B get identical results for components when these components are included in compositions) and sub-compositional dominance (i.e., using a subset of a complete composition carries less information than using the whole) (Quinn et al. 2018; Greenacre et al. 2021a; Greenacre et al. 2022). From a scientific standpoint, it seems to be a no-brainer to try to adhere to both of these properties, as they are the gold standard of reproducibility. Not following sub-compositional coherence would mean that two sequencing runs from the same patient (and the same bioinformatics pipeline) couldn’t be compared and ignoring sub-compositional dominance would mean we couldn’t filter data before using.

The log-ratio transformation that imparts all those properties is called isometric log-ratio (ILR). ILRs are considered the “gold standard” of log-ratio transformations, as they engender exactly the same multivariate geometric structure of the sample points as that of the formerly mentioned Aitchison geometry (Greenacre et al. 2021b). The ILR maps a composition in the D-part Aitchison-simplex isometrically to a D-1 dimensional Euclidian vector, which is not just confusing to understand but makes it also difficult to interpret (Greenacre et al. 2021a; Greenacre et al. 2022). Additionally, they are also particularly problematic when the numbers of components are high (Greenacre et al. 2021b), which is a quality worth considering as sequencing data is usually very high-dimensional.

Text, whiteboard

Description automatically generatedThankfully, we have more types of log-ratio transformation, that are easier to use and interpret. They do not fully impart sub-compositional coherence, but interestingly, it has been shown recently that quasi-coherence is sufficient in practice, as well as quasi-isometry (Greenacre et al. 2022), especially in high-dimensional data sets. As a result of this, it was decided to use two types of log-ratio transformation in this master thesis: ALR (additive log-ratio) and CLR (centered log-ratio).

Figure : Equation for CLR

The equation describes the calculation of CLR, with xj as vector of sample features, Dj the total number of features, and g(x) the geometric mean of sample vector x. Log-ratio transformations are applied within a sample (i.e., row-wise).

The CLR uses the geometric mean of the whole composition as the reference feature (Gloor et al. 2017). It has the advantage that it is computationally easy to do, which becomes more important with high-dimensional data sets. Furthermore, it reproduces the log-ratio geometry perfectly, but is not sub-compositionally coherent, because the whole composition (i.e., sample) is used to calculate the geometric mean and every sample will therefore use a different geometric mean. Unfortunately, it is not very easy to interpret and it is not very useful in sparse data containing a lot of 0s (Gloor et al. 2017).

A picture containing text, watch

Description automatically generatedThe second log-ratio transformation is ALR. Here, the log-ratio is taken of each measurement within a composition and divided by a chosen reference feature.

Figure 5: Equation ALR

The equation describes the calculation of ALR, with xj as vector of sample features, D the total number of features, and xref the reference feature. Log-ratio transformations are applied within a feature (i.e., column-wise).

Thus, the interpretation of ALR log-ratios is very straight-forward and it is also sub-compositional coherent, which is traded for a small loss of isometry. The biggest problem with ALR has always been the choice of reference. When choosing a reference, Greenacre et al. 2021 proposed to use three criteria to find a good reference: (i) the reference component should maximize the Procrustes correlation between the additive log-ratio geometry and the exact log-ratio geometry, (ii) the reference should minimize the variance the relative abundances of log-transformed components, and (iii) it should be a well populated component. Using these guidelines produces additive log-ratios close to being isometric, which would make them a favorable log-transformation. The obvious drawback is the computational complexity (if a Procrustes analysis is used beforehand), which increases especially in higher-dimensional data.

In general, log-ratio transformation do not normalize the data (does not “open it”), but makes the interpretation of the transformed data dependent on the reference used and aim for a straight-forward interpretation of the data (Quinn et al. 2018). For machine learning purposes, it is still unclear if any log-ratio transformation improves the performance in a prediction task. This will be one of the core goals of this benchmarking project and previous studies and results will be described in the next section.

Add section for chosen log ratio transformations

Furthermore, four log-ratio transformations will be compared: CLR (centered log-ratio), and three ALR methods (worst, random and optimal). The performances will be directly compared to TSS-transformed data (total sum scaling transformation) and *CoDaCoRe* in the following conceptual pipeline:

## Transformations

As mentioned in the introduction, choosing a log-ratio is not an easy decision. In order to stay with the goal of improving quality management and reducing human error, ILR will left out, as it is the most difficult one to work with and interpret. Similarly, pair-wise log-ratio transformations will also not be tested, as they are very computationally taxing. It has been decided to use TSS (total sum scaling transformation), which is standard relative abundance data, and compare it to CLR and ALR transformed data.

As ALR would be the most promising log-ratio transformation in terms of interpretability and its closeness to ILR, we will compare ALR transformation in three ways: (i) a random reference will be picked as denominator, to assess the average performance of machine learning models for ALR, (ii) find the most optimal denominator and (iii) worst ALR denominator via Greenacre et al. (2021) proposed way of finding a reference. Included in the package “easyCODA” is the function ALR() that assesses the abundances and variances of features in a data matrix, followed by a Procrustes analysis to assess their geometry. This leads to a list of possible good denominators for the respective data set if the top results are chosen or worst denominators, if the bottom results are selected. Similarly, “easyCODA” also contains the function CLR() to compute the centered log-ratio.

# Machine Learning

## Supervised Machine Learning

Predictive methods such as random forests (RF), artificial neural networks (ANN), deep learning (DL) or support-vector machines (SVM) and other methods have become in the last years increasingly popular (Tolosana-Delgado et al. 2019). Vital for good machine learning conclusions is the balancing of predictive power with the explainability, similarly to log-ratio transformations.

In terms of statistical analysis, machine learning models are of great interest for microbiome analysis, as they allow predictions of biomarkers, phenotypes or microbial taxa, as well as other interesting tasks, that are not possible with the standard microbiome tool kit (Marcos-Zambrano et al. 2021). Therefore, a correct application of machine learning models is key to reproducible and interpretable research results. Several studies (Zhang and Shi 2019; Coenders and Greenacre 2021) showed log-ratio transformations in machine learning models with mixed performances. In 2019, Zhang and Shi compared several machine learning algorithms on geological compositional data and showed that overall, RF was the best performing model and that ILR and CLR were superior to ALR (Zhang and Shi 2019). Tolosana-Delgado et al. (2019) concluded that ridge regression and SVM both need ILR. More observations were also made by Quinn et al. 2020. They performed linear discriminant analysis (LDR) on ILR-transformed data and partial least squares (PLS) to CLR-transformed data and showed good predictive results (Quinn and Erb 2020). Neural Networks require further research, but does not seem to be equivariant (Tolosana-Delgado et al. 2019), i.e. not any log-ratio transformation works similarly well.

These observations demonstrate the current predicament between compositional data and machine learning. Log-ratio transformation in linear and generalized linear models are not easily chosen and depend heavily on the observations at hand. In general, log-ratio transformations seem to outperform raw proportions for classification tasks, but it is not clear how log-ratio transformations relate to the changes in predictive performance. Furthermore, employing log-ratio transformations leads to an increase in complexity in the correct application of machine learning models. Thus, it is of increasing importance to create a practical guide for all scientists who want to employ such analysis.

The question arises if machine learning models are “worth the hassle” considering microbiome-specific algorithms like *CoDaCoRe* exist, that are faster and do not require a lot of background knowledge to use. The next section describes this algorithm more in detail and their potential effectiveness.

Add section for chosen machine learning models

Therefore, the focus will be on standard machine learning models, that are already incorporated in easy-to-use packages in R: generalized linear models (GLMs) and support vector machines (SVM) as non-linear approach.

## CODACORE

The following section is a summary of the paper *“Learning Sparse Log-Ratios for High-Throughput Sequencing Data”* published by Gordon-Rodriguez et al in 2021, where they first introduce *CoDaCoRe.* *CoDaCoRe* is a novel learning algorithm for finding balances (Compositional Data via Continuous Relaxations). Balances are defined as the log-ratios between geometric means of two features of the input variables. Translated, CoDaCoRe finds ratios between two features that are explanatory for the given classification task. Such ratios are commonly used as biomarkers of gut health e.g., the Firmicutes-to-Bacteroidetes ratio (Crovesy et al., 2020; Magne et al.; 2020).

Balances are essentially pairwise log-ratios; however, they allow the aggregation of more than one variable in the numerator and denominator of the log-ratio. This leads to a richer set of features and therefore more flexible models. Usually, pairwise log-ratios are computationally very taxing, which is why they are not separately included in the master thesis. However, in *CoDaCoRe* Gordon-Rodrigues et al. use a deep learning technology called “continuous relaxation” and only approximate the optimization problem, which has the advantage of greatly reducing the runtime.

In its basic formulation, *CoDaCoRe* learns a regression function, which uses balances as weights. The goal of *CoDaCoRe* is to find the balance that is maximally associated with the response variable by minimizing the cross-entropy loss. The continuous relaxation approximates the geometric averages over subsets of the inputs, by weighted geometric averages of all components. This makes the relaxation and balances differentiable and allows the use of gradient descent. This has the advantage of a linearly scaling computational cost instead of exponential, which reduces the runtime drastically.

At this step, weighted geometric averages or not easily interpretable. Therefore, *CoDaCoRe* implements a discretization procedure, i.e., fitting a linear model to assess if the previously found balance is impactful. This step can be regularized by influencing lambda in the model creation, which becomes a regularization hyperparameter that can be tuned. In practice, lower lambda is more useful when the emphasis is on predictive accuracy rather than interpretability or sparsity.

In summary, in the full *CoDaCoRe* algorithm, multiple regressors are trained in a stage-wise additive fashion and afterwards each successive balance is fitted on the residual from the current model. Thus, *CoDaCoRe* identifies a sequence of balances, in decreasing order of importance, each of which is sparse and interpretable.

*CoDaCoRe* is a promising algorithm that is created to also work efficiently on big data sets with a lot of features. In their paper, the authors compare *CoDaCoRe* against several machine learning models (Lasso, RF and XGBoost) and show that their algorithm does not sacrifice interpretability nor predictive accuracy.

# Framework

Recent years made it clear that machine learning is a tool that should be available to all biologists, but comes with high complexity and its own pitfalls, even without the addition of mathematical characteristics of compositional data and log-ratio transformations. The above-described examples already mention that machine learning models in combination with log-ratio transformations do not show clear-cut results. The goal of this project focuses on collecting insights on the performances of machine learning models, but also practicality.

The following sections describe the framework and pipeline in which the chosen machine learning models and log-ratio transformations are puzzled in.

Diagram

Description automatically generated

Figure 5: Used Pipeline

The graph shows the proposed pipeline for the benchmarking project. Data sets will be collected by their characteristics large/small, high/low correlations and continuous/discrete variables. Afterwards, data sets will be pre-processed by zero-imputation methods and filtering. Microbiome-native methods will be employed and compared to the data being log-transformed and used in machine learning models.

The general pipeline will be constructed of the following building blocks: Pre-processing, Imputation, Transformation, and Machine Learning Models/Microbiome Approaches. The core idea is to observe statistical fluctuations in two chosen machine learning models, given the same training data set. After the data split, a repeated cross-validation is used to find the best model and its performance is saved for plotting. It will be explored in further detail if filtering affects model performances, as well as the different transformations itself.

## Data sets

As several authors pointed out (Quinn and Erb 2020; Gloor et al. 2017), machine learning performance is influenced by data size. Therefore, three data sets were chosen accordingly to include direct comparison of performances of small and large data sets, as well as high and low known correlations microbiome and host, as well as continuous and discrete predictive variables.

First is a Colorectal Cancer (CRC) set. The CRC data set was first used and described by Wirbel et al. 2019 in their meta-analysis for colorectal cancer. This data set is well known and contains 7727 features with 695 samples. It shows clear correlations between gut microbiota and colorectal cancer and is therefore helpful to show the behavior of transformations and machine learning algorithms on small but highly specific data sets. The second data set is the Polycystic Ovary Syndrome (PCOS) data set described by Kreete et al. (2020). It observed 312 individuals, with two-thirds of them being healthy, and 72738 features. It is a valuable addition as it is a small data set (sample-wise) that shows no correlation between the disease and microbiome structure. Lastly, is the Estonian Biobank microbiome cohort (EstMB). This data set includes 2509 individuals with several phenotypical markers collected over time and 17180 features overall, which makes it by far the largest of all three sample-wise. All data sets contain at least one discrete and one continuous response variable.

## Pre-Processing

### Cleaning and filtering

First, all data sets were cleaned in order to remove NAs in predictor columns or patients that have no sequencing data. In EstMB data set, 21 rows removed in metadata due to NA and 21 patients respectively cut out of abundance table. This leads to 2485 final sample-size. In CRC, 128 rows were removed due to NA in feature “BMI”, leading to 567 samples overall. Additionally, the column “X.1” has been removed as it is only a sum of all abundances per row. In PCOS, 6 rows removed in abundance table because no matching patient has been found in metadata, reducing the number of samples to 304.

As microbiome data usually has a lot of features, the computational work can be taxing. Therefore, filters were applied to all three data sets. In this benchmarking project, taxa with ≥10% abundance in samples will be discarded. Additionally, a filter of ≥50% abundance in samples will be applied, as well as a mean relative abundance filter for 0.001. For 10% abundance filters EstMB keeps 9738 features, CRC 650 features and PCOS 1154 features. Respectively, for 50% abundance filters EstMB keeps 5233 features, CRC 189 features and PCOS 120 features. For EstMB data, 90% abundance was used, as the data was otherwise not practically usable without heavy computational power. EstMB keeps 3062 features after filtering.

### Imputation

One of the main problems of microbiome data is its sparse nature. When working with relative abundances this is annoying but doesn’t have any mathematical consequences. On the other hand in log-ratio transformation zeros lead to problems, as log(0) is undefined. Therefore, one of the first steps after filtering and before log-ratio transformation is zero-imputation. Introduced by (Palarea-Albaladejo and Martín-Fernández 2015) is pseudocount. It has been frequently used for statistical analysis of microbiome data. It adds a pseudo-count of 1E-05 to avoid non-finite values resulting from log(0). All three data sets were imputed with Geometric Bayesian Multiplicative (GBM) and output form “p-counts”.

As a side quest, these three data sets give the option to explore several other concepts important in machine learning and the potential influence of log-ratio transformations. As the study design for the CRC data set had holdout data sets in mind, this opens the possibility of comparing model performances for Holdout sets versus a classic 80/20 split as is planned for the general pipeline.

An additional point of interest is the impact of data leakage on transformations and machine learning models. It is considered good standard-of-practice to split the data before any pre-processing or imputation is conducted. As transformations fall under the category “pre-processing” and as ALR is conducted column-wise in nature, it should be tested how machine learning models are impacted under data leakage and non-data-leakage circumstances.

We hypothesize that

# Methodology

For the data analysis and model pipelines, the script language R (v4.1.3) in combination with RStudio (v2022.02.1+461) has been used. For data cleaning and filtering the main libraries is “tidyverse” (1.3.1). Imputation was conducted with “zCompositions” (1.4.0.1), and transformations were mostly done with “easyCODA” (0.34.3). Models were constructed with “mikropml” (1.2.2), “tidymodels” (0.2.0) and “codacore” (0.0.3).

Additionally, own scripts were created for convenience purposes. All scripts can be found on Github JenniferNeumaier/ml\_coda.

## Transformations

As mentioned in the introduction, choosing a log-ratio is not an easy decision. In order to stay with the goal of improving quality management and reducing human error, ILR will left out, as it is the most difficult one to work with and interpret. Similarly, pair-wise log-ratio transformations will also not be tested, as they are very computationally taxing. It has been decided to use TSS (total sum scaling transformation), which is standard relative abundance data, and compare it to CLR and ALR transformed data.

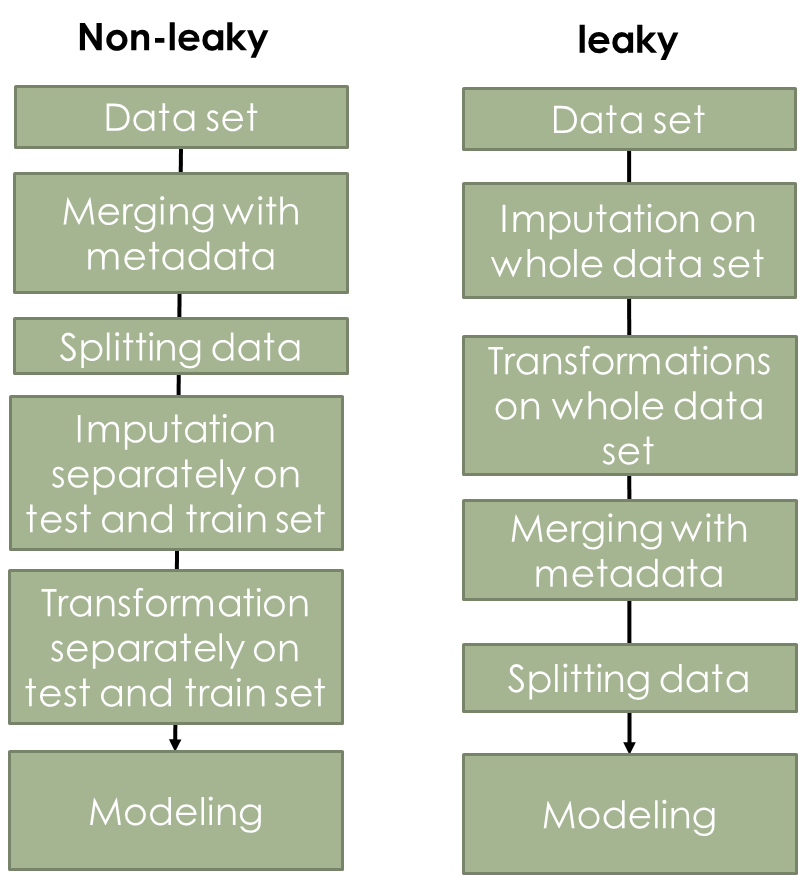
As ALR would be the most promising log-ratio transformation in terms of interpretability and its closeness to ILR, we will compare ALR transformation in three ways: (i) a random reference will be picked as denominator, to assess the average performance of machine learning models for ALR, (ii) find the most optimal denominator and (iii) worst ALR denominator via Greenacre et al. (2021) proposed way of finding a reference. Included in the package “easyCODA” is the function ALR() that assesses the abundances and variances of features in a data matrix, followed by a Procrustes analysis to assess their geometry. This leads to a list of possible good denominators for the respective data set if the top results are chosen or worst denominators, if the bottom results are selected. Similarly, “easyCODA” also contains the function CLR() to compute the centered log-ratio.

## Machine LEarning models

Figure .. shows the pipeline for comparing the performance of machine learning models. First, the preprocessed (and imputed) data set was transformed using TSS, CLR, and various ALR options. Afterwards, the data set was split 80/20 with stratification with the package “tidymodels”. The train set given to the function run\_ml() from “mikropml” by. This package nicely compacts the use of standard machine learning models to a few lines of code and supports the use of GLMs (glmnet), as well as XGBoost (xgbTree). As shown in the pipeline, it was of interest to control the initial split into test and train data, which is also allowed by mikropml. The training set is then split into 5 folds and the best model assessed via 10-fold cross validation and the final test and training scores of the best model are saved for plotting. This procedure is repeated 10x for each model and each data set to assess statistical fluctuations of model performances and accuracies.

As it was of interest to compare the machine learning model performances to codacore, the pipeline includes codacore directly. The first initial data split is fed to the function codacore(). Two codacore models are trained, one with lambda = 0 and the other with lambda = 1. For both models, AUC values were predicted with all balances taken into consideration and one were only the best balance was used. This creates four performance scores per repetition and saved for further plotting. The codacore function is also repeated 10x to catch statistical fluctuations under the same data split. For discrete response variables, AUC is chosen as performance score, and for continuous response variables RMSE.

## Data Leakage in Transformations

In order to assess the influence of transformation on the concept of “data leakage” (machine learning best practice paper), a small test was conducted. In this, the CRC and PCOS data sets were used, and it was compared how the test and training set performances behave with imputation and transformation before data merging vs. after data merging (see figure).

In the non-leaky procedure, the processed data set has been first merged with the metadata to include the predictor column. Afterwards, the data is split into train and test set and imputation, as well as three transformations (TSS, CLR and optimal ALR) are performed separately on both. Finally, train and test set are fed into a glmnnet model.

In the leaky procedure, imputation and transformations are conducted on the whole data set and afterwards the data is merged with the metadata and split into train and test set and fed into the glmnet model. Both procedures were repeated 10x to observe statistical fluctuations. Only 10% abundance filter data sets have been used.

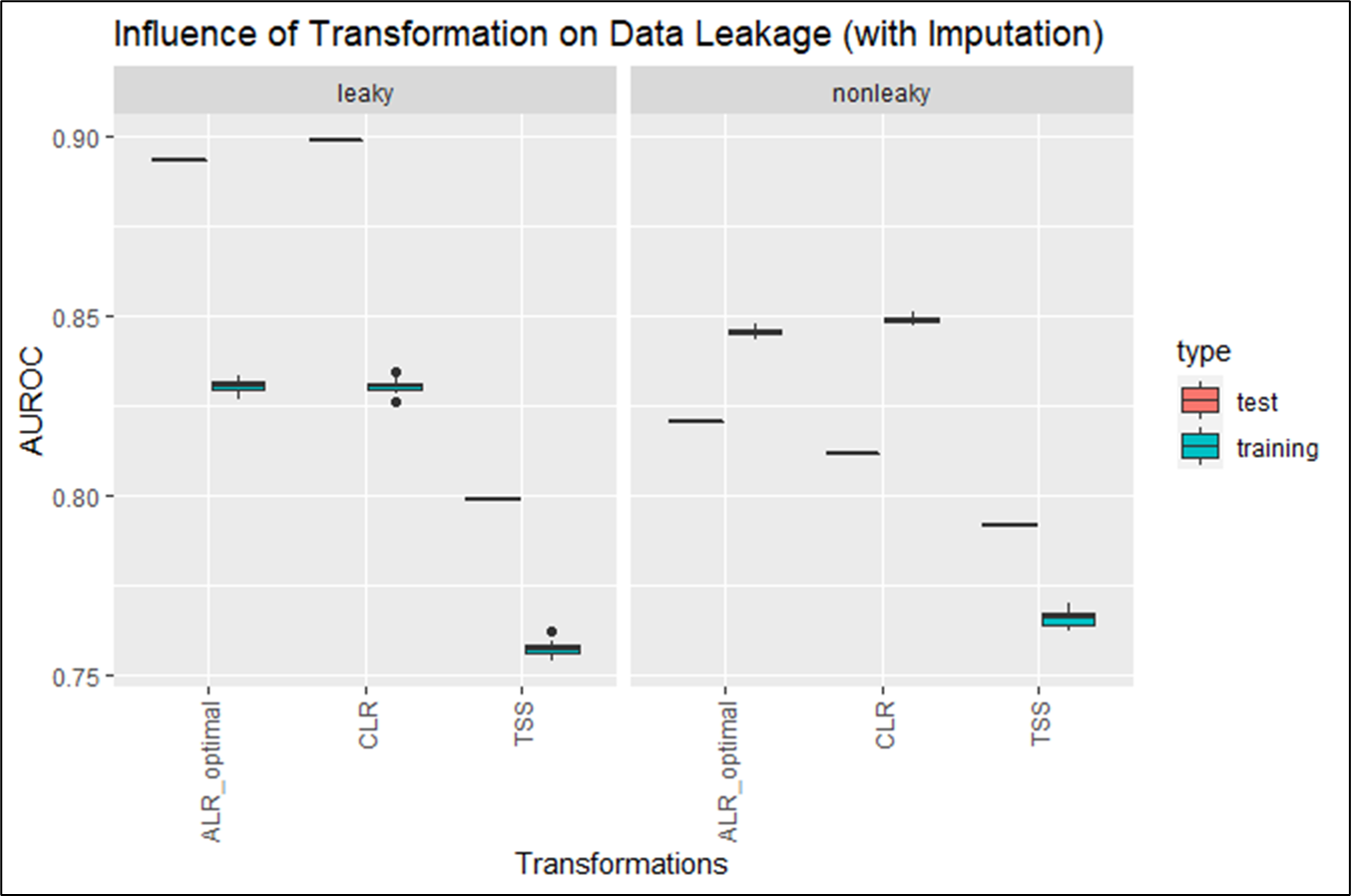
# Results

In the following section all results gathered throughout the project are introduced and described. This section is divided into three parts, each containing the results of one dataset, comparing different transformations directly with codacore and selbal results.

## Holdout vs 80/20

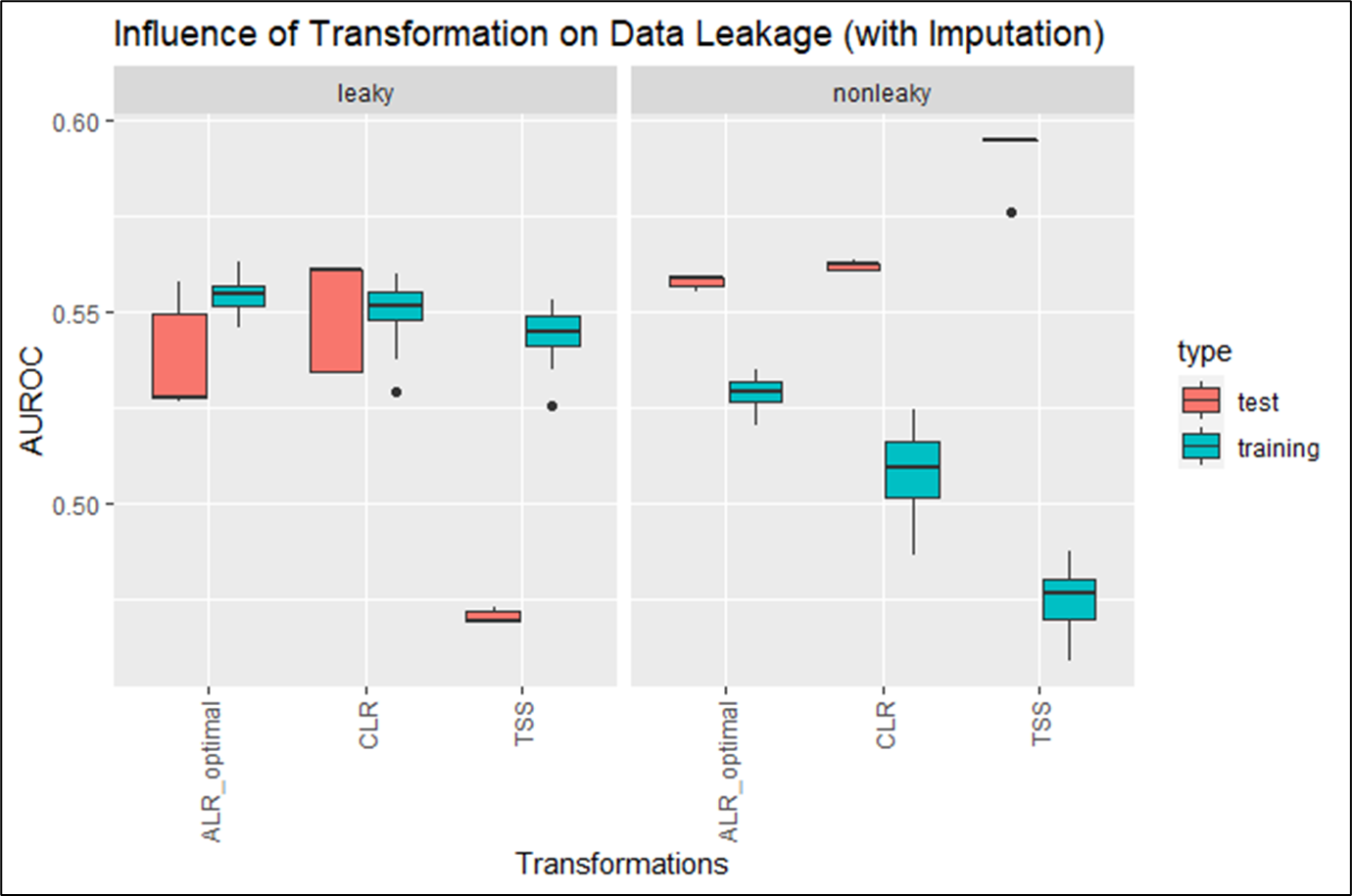
## Leaky Preprocessing

In these results, CRC (10% filtering and imputed) was predicted once on GER and once on FRA as test set and both results combined for this figure.



The figure compares the behavior of AUC in train and test set in a leaky and non-leaky procedure. It can be seen that the achieved AUCs for ALR\_optimal and CLR are both around 0.86, however the leaky procedure shows a lower variance compared to non-leaky. TSS shows in both procedures a AUC between 0.6 and 0.78, with the training set converging on a AUC of around 0.78.

The same procedure was used on the PCOS data set to estimate the influence of data leakage in less clear correlations.



Similar to figure () leaky and non-leaky behavior is compared directly. In PCOS the AUC varies between 0.55 and 0.6, with the training sets converging on 0.55. The test sets however show higher variances.

## Colorectal Cancer Dataset (CRC)

### GLMNET

#### Classification

In this figure, the predictor column was “Group”, i.e. the column that determined if someone had cancer or not. All models were trained on the same training set and the same test set. A holdout set was not used here.

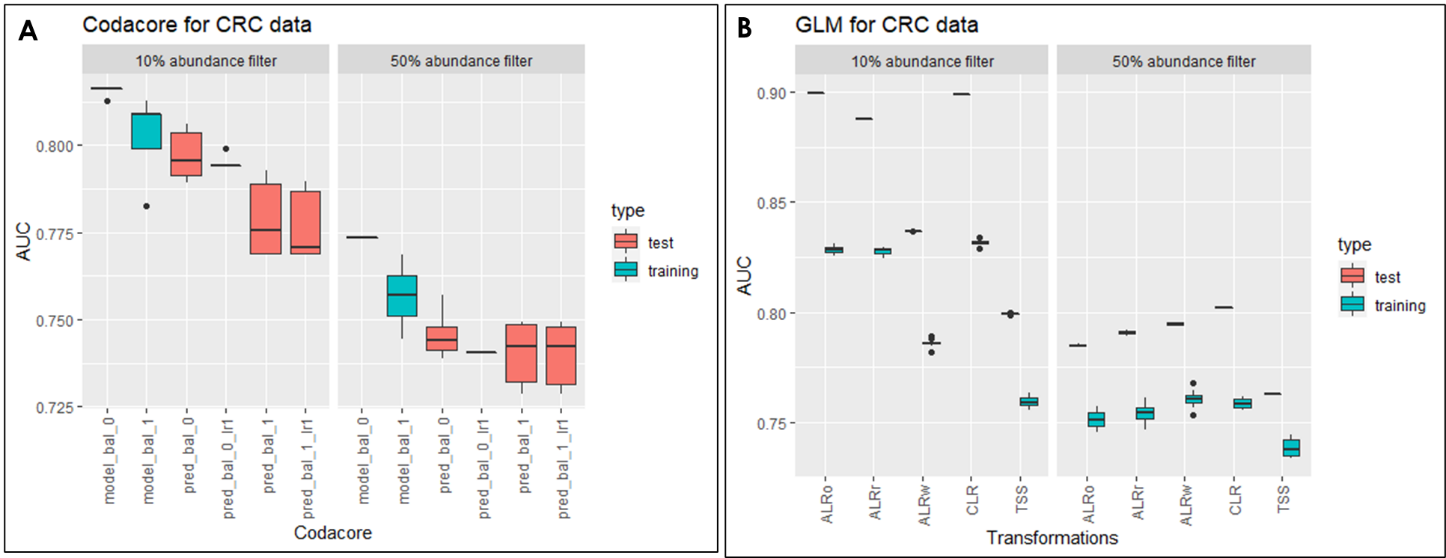


Figure A shows the result for codacore split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the AUC performances and on the x-axis are all codacore models. In both data sets the training set shows higher performances than the test set, with an AUC for 10% abundance filter data set of 0.8 and 0.775 for 50% abundance filter set. The test set ranges between 0.8 for 10% abundance filter and 0.75 for 50% abundance filter for all codacore models.

Figure B shows the result for glmnet split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the AUC performances and on the x-axis are the different transformations. Performances for train and test set vary greatly over all transformations for both data sets, with the train tests showing significantly higher performances between 0.8 and 0.9 for 10% abundance data set and constant 0.8 for 50% filter set. The training set is in general significantly lower than the respective test set with the lowest performances in the 50% abundance filter set with AUC performances around 0.75.

#### Regression

In this figure, the predictor column was “BMI”. All models were trained on the same training set and the same test set. A holdout set was not used here.

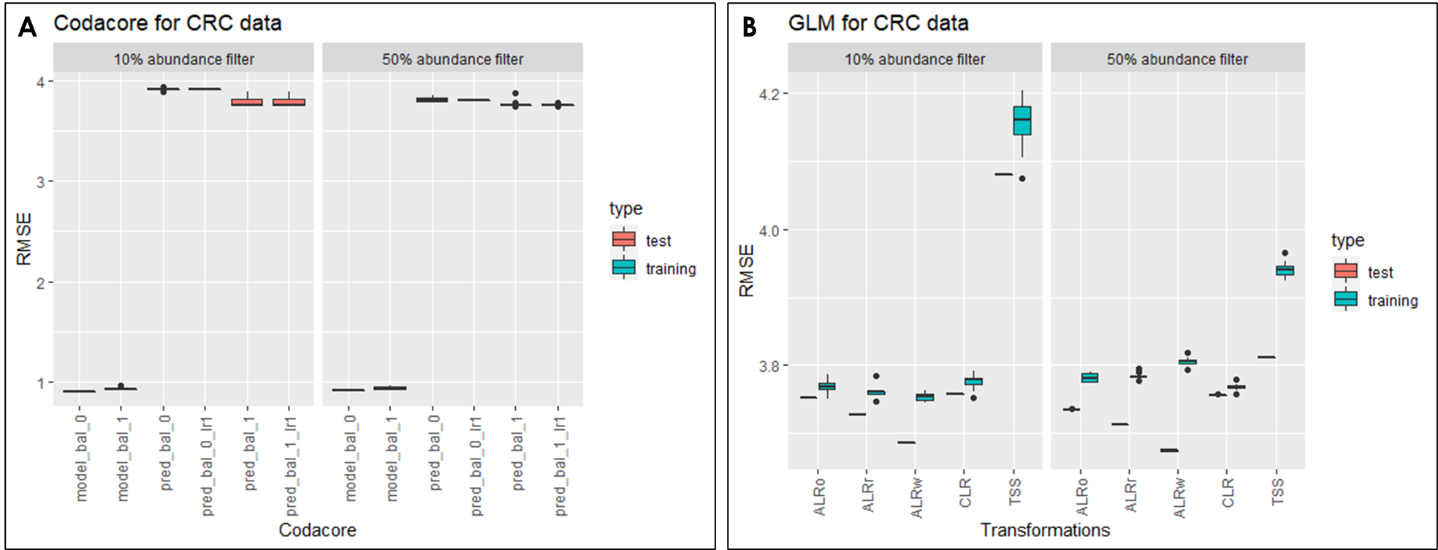


Figure A shows the result for codacore split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the RMSE performances and on the x-axis are all codacore models. In both data sets the test set shows higher performances than the training set, with an RMSE between 1 for all training sets and 4 for all test sets with very low variances.

Figure B shows the result for glmnet split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the RMSE performances and on the x-axis are the different transformations. Performances for train and test sets sit around 3.8 for all transformations besides TSS which shows performances for train and test set around 4.2 for 10% abundance filtering and 3.9 for 50% abundance filters. In general, test sets show lower performances than training sets.

## Polycystic Ovary Syndrome Dataset (PCOS)

The following section describes the result for several machine learning models and their performances on the PCOS data set under various transformations.

### GLMNET

#### Classification

In this figure, the predictor column was “PCOS\_Riikka”, i.e. the column that determined if someone had PCOS or not. All models were trained on the same training set.

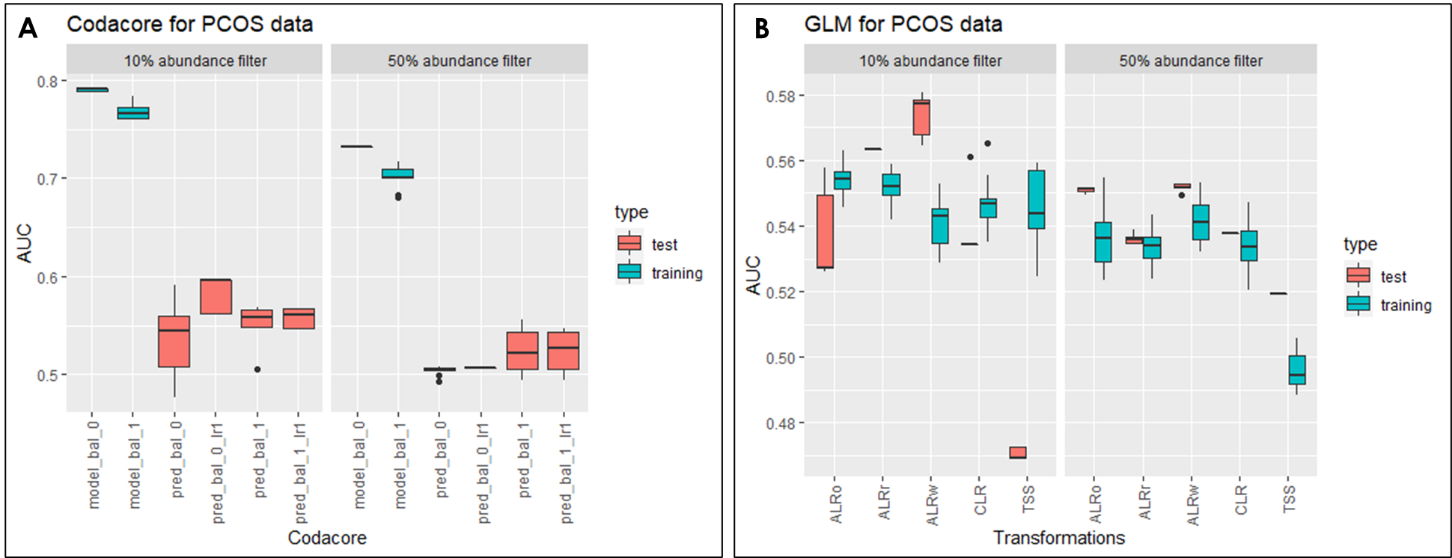


Figure 8

ALRw = worst ALR, ALRo = optimal ALR, ALRr = random ALR.

Figure A shows the result for codacore split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the AUC performances and on the x-axis are all codacore models. In both data sets the training set shows higher performances than the test set, with an AUC between 0.7 and 0.8. The test set ranges between 0.5 and 0.6 for all codacore models.

Figure B shows the result for glmnet split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the AUC performances and on the x-axis are the different transformations. Performances for train and test set vary greatly over all transformations for both data sets, with the train tests showing also higher variances. In general, the AUC trends between 0.48 and 0.58, with the test set for TSS as the lowest and worst ALR as the highest with 0.58.

#### Regression

In this figure, the predictor column was “BMI”. All models were trained on the same training set.

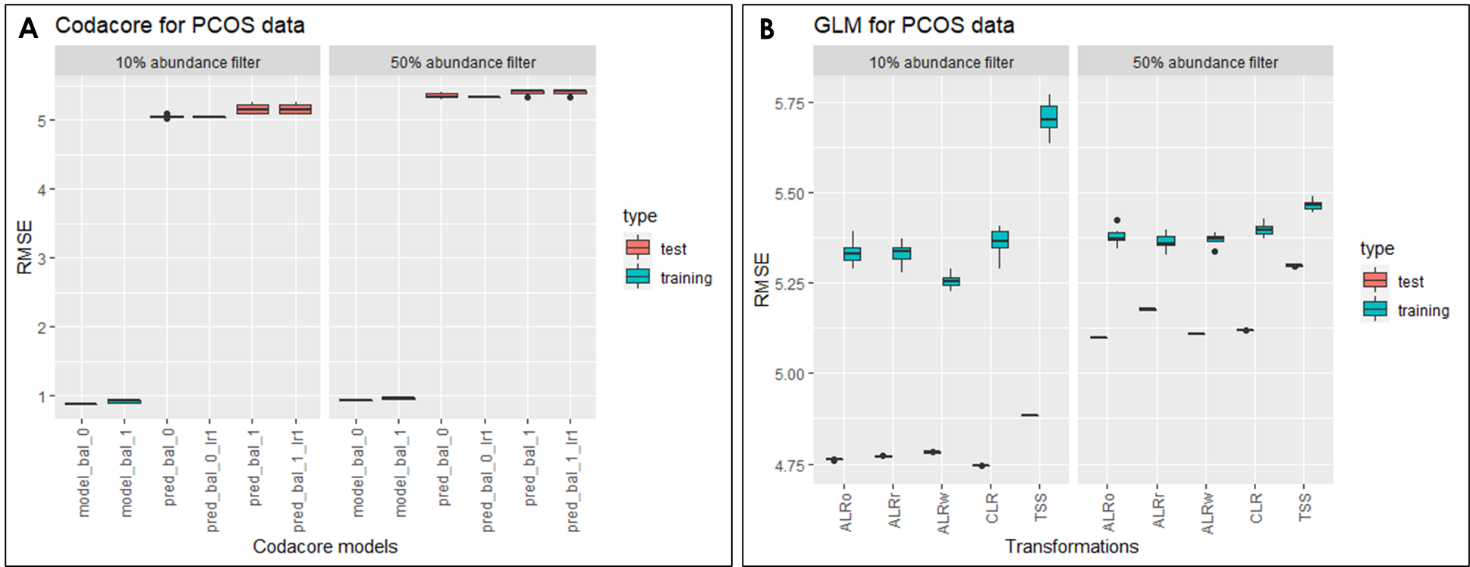


Figure 9

ALRw = worst ALR, ALRo = optimal ALR, ALRr = random ALR.

Figure A shows the result for codacore split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the RMSE performances and on the x-axis are all codacore models. In both data sets the test set shows higher performances than the training set, with an RMSE between 1 for all training sets and 5 for all test sets with very low variances.

Figure B shows the result for glmnet split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the RMSE performances and on the x-axis are the different transformations. Performances for train and test set vary greatly over all transformations for both data sets, with the train tests showing also higher variances. In general, the RMSE for training sets trends between 5.25 and 5.75, and the test set around 4.75 for 10% abundance and 5.1 to 5.25 for 50% filtering.

### xgboost

#### classification

#### regression

## Estonian Biobank Dataset (EstMB)

# Discussion

Mention behaviour of sub-compositional coherence (Greenacre et al. 2022) with 10% and 50% filtered datasets

Outlook:  
data-driven alpha-transformations: <https://hal.archives-ouvertes.fr/hal-03379935v2/document>

Note that the use of

α-transformations also enables one to deal with the presence of 0s in the compositions, unlike the

log-ratio approach which is only suitable for strictly positive compositions.

Performance selbal and codacore:

* takes quite a while for 500x1000 data set
* at least 5-10 minutes for selbal
* codacore faster
* with CV even more
* tensorflow necessary for codacore -> installation problems

comparison codacore and mikropml:

* impact of filtering -> 50% seems to lose too many features, performance generally worse than 10%.
* PCOS: regression overfitting, classification all over the place
* CRC: regression pretty constant (varies only in second decimal place), classification underfitting
* ALRo and CLR seem to have very similar results over both data sets and model types
* Even ALRw is better than TSS for both high and low correlation and regression and classification -> supports former papers that suggest using transformations for compositional data -> also for machine learning concepts
* Do CRC holdout AUCs match paper? Yes
* Transformations plus standard split seems to UNDERFIT data?
* 50% abundance is too few features so not even transformations impact performances

This is because an underfit model has low variance and high bias. Variance refers to how much the model is dependent on the training data. For the case of a 1 degree polynomial, the model depends very little on the training data because it barely pays any attention to the points! Instead, the model has high bias, which means it makes a strong assumption about the data. For this example, the assumption is that the data is linear, which is evidently quite wrong. When the model makes test predictions, the bias leads it to make inaccurate estimates. The model failed to learn the relationship between x and y because of this bias, a clear example of underfitting. (<https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765>) -> xgboost performance in comparison?

**Q: how is codacore working?**

Data leakage:

* in data sets with clear correlation, performance is similarly good, with leaky procedure showing lower variance and therefore preferable.
* Reasoning: using test and train set to perform transformation could potentially not be big enough and therefore lead to higher variances in transformation results. Also, denominators for ALR were different for test and train set -> both denominators were removed for modeling
* Makes interpretability even harder -> use fixed denominator for test set (i.e. denominator from train set)?
* As leaky procedure does not seem problematic it is practicable to conduct transformations on the whole data set
* For data sets that show low correlations and are difficult for models, the nonleaky prodecure seems to work better. However, the AUCs are very similar.

**Q: How is imputation influencing leakiness? -> include imputation in pipeline**

Publication bibliography

Aitchison, J. (1982): The Statistical Analysis of Compositional Data. In *J. Royal Statistical Society* 44, pp. 139–177.

Aitchison, J. (2003): A Concise Guide to Compositional Data Analysis. Girona, Italy.

Cawley, Gavin C.; Talbot, Nicola L. C. (2010): On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. In *Journal of Machine Learning Research* 11, pp. 2079–2107.

Coenders, Germa; Greenacre, Michael (2021): Three approaches to supervised learning for compositional data with pairwise logratios. Available online at http://arxiv.org/pdf/2111.08953v1.

Gloor, Gregory B.; Macklaim, Jean M.; Pawlowsky-Glahn, Vera; Egozcue, Juan J. (2017): Microbiome Datasets Are Compositional: And This Is Not Optional. In *Frontiers in microbiology* 8, p. 2224. DOI: 10.3389/fmicb.2017.02224.

Gordon-Rodriguez, Elliott; Quinn, Thomas P.; Cunningham, John P. (2021): Learning Sparse Log-Ratios for High-Throughput Sequencing Data.

Greenacre, Michael; Grunsky, Eric; Bacon-Shone, John (2021a): A comparison of isometric and amalgamation logratio balances in compositional data analysis. In *Computers & Geosciences* 148, p. 104621. DOI: 10.1016/j.cageo.2020.104621.

Greenacre, Michael; Grunsky, Eric; Bacon-Shone, John; Erb, Ionas; Quinn, Thomas (2022): Aitchison's Compositional Data Analysis 40 Years On: A Reappraisal. Available online at http://arxiv.org/pdf/2201.05197v1.

Greenacre, Michael; Martínez-Álvaro, Marina; Blasco, Agustín (2021b): Compositional data analysis of microbiome and any-omics datasets: a revalidation of the additive logratio transformation.

Griffith, Malachi; Walker, Jason R.; Spies, Nicholas C.; Ainscough, Benjamin J.; Griffith, Obi L. (2015): Informatics for RNA Sequencing: A Web Resource for Analysis on the Cloud. In *PLoS Comput Biol* 11 (8), e1004393. DOI: 10.1371/journal.pcbi.1004393.

Marcos-Zambrano, Laura Judith; Karaduzovic-Hadziabdic, Kanita; Loncar Turukalo, Tatjana; Przymus, Piotr; Trajkovik, Vladimir; Aasmets, Oliver et al. (2021): Applications of Machine Learning in Human Microbiome Studies: A Review on Feature Selection, Biomarker Identification, Disease Prediction and Treatment. In *Frontiers in microbiology* 12, p. 634511. DOI: 10.3389/fmicb.2021.634511.

Palarea-Albaladejo, Javier; Martín-Fernández, Josep Antoni (2015): zCompositions — R package for multivariate imputation of left-censored data under a compositional approach. In *Chemometrics and Intelligent Laboratory Systems* 143, pp. 85–96. DOI: 10.1016/j.chemolab.2015.02.019.

Pawlowsky-Glahn, V.; Egozcue, J. J. (2001): Geometric approach to statistical analysis on the simplex. In *Stochastic Environmental Research and Risk Assessment* 15, pp. 384–398.

Pawlowsky-Glahn, Vera; Egozcue, Juan José (2016): Spatial analysis of compositional data: A historical review. In *Journal of Geochemical Exploration* 164, pp. 28–32. DOI: 10.1016/j.gexplo.2015.12.010.

Quinn, Thomas P.; Erb, Ionas (2020): Interpretable Log Contrasts for the Classification of Health Biomarkers: a New Approach to Balance Selection. In *mSystems* 5 (2). DOI: 10.1128/mSystems.00230-19.

Quinn, Thomas P.; Erb, Ionas; Richardson, Mark F.; Crowley, Tamsyn M. (2018): Understanding sequencing data as compositions: an outlook and review. In *Bioinformatics (Oxford, England)* 34 (16), pp. 2870–2878. DOI: 10.1093/bioinformatics/bty175.

Rivera-Pinto, J.; Egozcue, J. J.; Pawlowsky-Glahn, V.; Paredes, R.; Noguera-Julian, M.; Calle, M. L. (2018): Balances: a New Perspective for Microbiome Analysis. In *mSystems* 3 (4). DOI: 10.1128/mSystems.00053-18.

Tolosana-Delgado, Raimon; Khodadadzadeh, Mahdi; Talebi, Hassan (Eds.) (2019): On machine learning algorithms and compositional data.

Tsamardinos, Ioannis; Rakhshani, Amin; Lagani, Vincenzo (2015): Performance-Estimation Properties of Cross-Validation-Based Protocols with Simultaneous Hyper-Parameter Optimization. In *Int. J. Artif. Intell. Tools* 24 (05), p. 1540023. DOI: 10.1142/S0218213015400230.

Wirbel, Jakob; Pyl, Paul Theodor; Kartal, Ece; Zych, Konrad; Kashani, Alireza; Milanese, Alessio et al. (2019): Meta-analysis of fecal metagenomes reveals global microbial signatures that are specific for colorectal cancer. In *Nature medicine* 25 (4), pp. 679–689. DOI: 10.1038/s41591-019-0406-6.

Zhang, Mo; Shi, Wenjiao (2019): Systematic comparison of five machine-learning methods in classification and interpolation of soil particle size fractions using different transformed data.

# Supplementary

## Influence of Transformation on Holdout vs. 80/20

This test was focused on showing if there are differences in the model performances in various transformations when choosing a specific holdout set vs a standard 80/20 set. As can be read in the CRC paper, it has been decided to use a holdout set and leave-one-out principle to validate the model. As the other data sets do not have the option and it would be of interest to see if transformation impact the choice of test sets, the AUC performances of a standard 80/20 split were compared to performances for every holdout group in the CRC metadata.

Chart

Description automatically generated

The figure shows the result for glmnet split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the AUC performances and on the x-axis are the different transformations. The data set has been trained on the predictor “Group”. In these boxplots, performances for training and test were combined to capture the distances between them. In the supplementary a figure is added that shows every data point. Using USA as holdout set leads to the biggest distance between training and test performance (0.7 to 0.9) for all transformations and data set types. Using GER as holdout set produces the smallest distance between test and training set performances. In general, TSS performs worse compared to all other transformations, with CLR and ALR random and ALR optimal having the highest test set performances for 10% abundance filter.

Separating train and test performances and instead combining all holdout performances support the claim for consistency of 80/20:

Chart, scatter chart, box and whisker chart

Description automatically generated

The figure shows the result for glmnet split into the performances for 10% abundance (left) and 50% abundance (right). The y-axis contains the AUC performances and on the x-axis are the different transformations. The data set has been trained on the predictor “Group”. It can be seen that the performances on the standard test split show higher training and test performances than the combined holdout set performances. Interestingly, the standard split seems to be underfitting the data, compared to the holdout set method. Holdout AUCs match compared to original paper (in range of 0.7 and 0.8). Standard split seems to perform even better.

