Jennifer Neumaier

Institute of Genomics, University of Tartu

Master Thesis

*Benchmarking machine learning performances with compositional* *data*

Table of Contents

[Introduction 2](#_Toc104466246)

[Characteristics of Compositional Data 3](#_Toc104466247)

[The Simplex Space 5](#_Toc104466248)

[Mapping the Simplex Space into Euclidean Space 6](#_Toc104466249)

[Log-Ratio Transformations 8](#_Toc104466250)

[The Problem 10](#_Toc104466251)

[Machine Learning 10](#_Toc104466252)

[CODACORE 11](#_Toc104466253)

[Implementation/Assessment 12](#_Toc104466254)

[Results/Analysis 15](#_Toc104466255)

[INFLUENCE OF TEST SET ON PERFORMANCES 16](#_Toc104466256)

[INFLUENCE OF TRANSFORMATION ON PERFORMANCES 18](#_Toc104466257)

[Binary Classification 18](#_Toc104466258)

[Regression Setting 20](#_Toc104466259)

[INFLUENCE OF DATA LEAKAGE ON PERFORMANCES 22](#_Toc104466260)

[Discussion/Conclusion 26](#_Toc104466261)

[Publication bibliography 29](#_Toc104466262)

[Supplementary 31](#_Toc104466263)

[Methodology 31](#_Toc104466264)

[Data set 31](#_Toc104466265)

[Pre-Processing 31](#_Toc104466266)

[Imputation 32](#_Toc104466267)

[Transformations 32](#_Toc104466268)

[Machine LEarning models 33](#_Toc104466269)

# Introduction

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.

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 1: 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

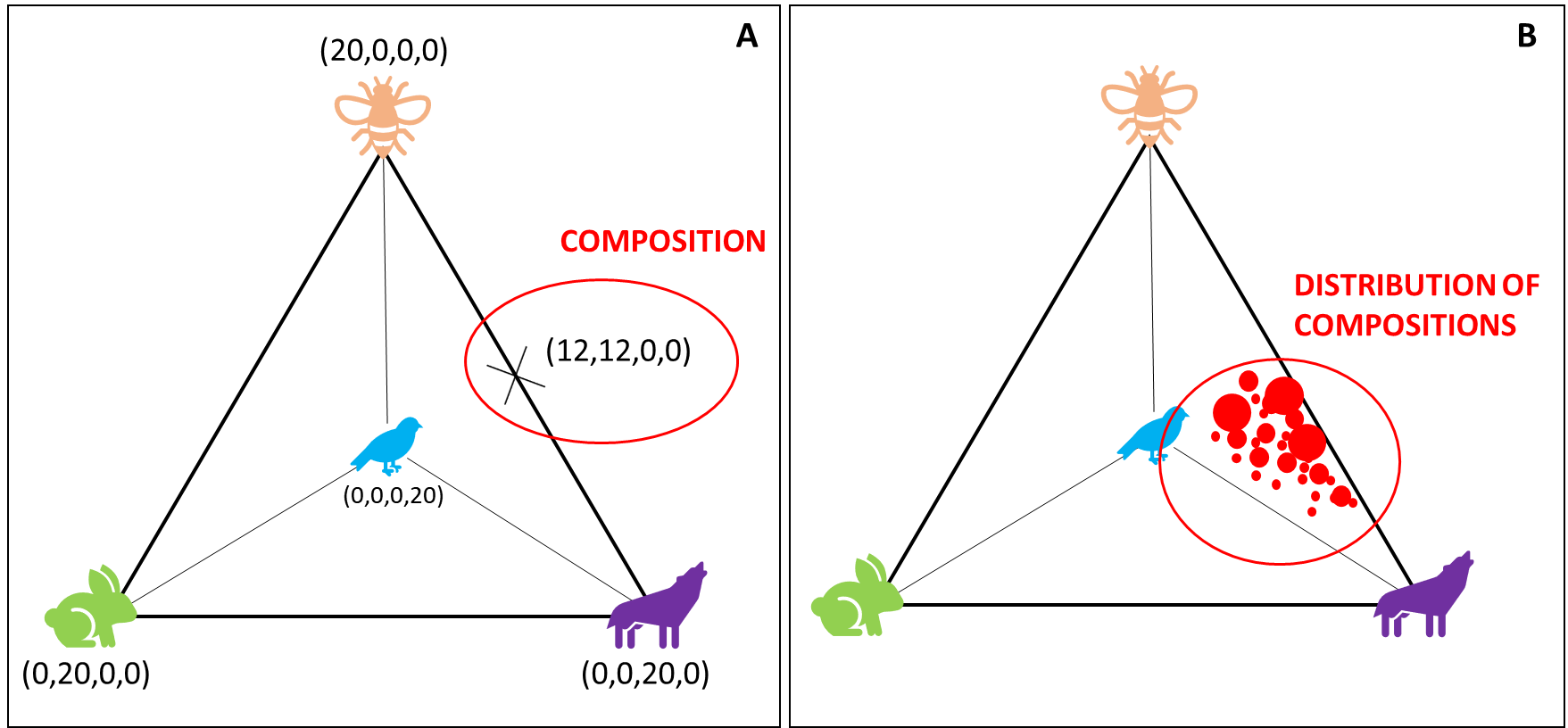
As mentioned, absolute counts in compositional data are irrelevant and only relative abundances are of interest. This puts the data in the so-called “simplex space”, instead of the for us more common Euclidean space. The following Figure 2 shows how the data from field A would look like in the simplex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

Figure 2: 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 10 bees, 10 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 3: 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 : 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).

# The Problem

## 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. However, it is still unclear if any log-ratio transformation improves the performance in a prediction task.

Several studies showed log-ratio transformations in machine learning models with mixed performances (Zhang and Shi 2019; Coenders and Greenacre 2021). 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.~~

## CODACORE

An alternative solution to standard machine learning models as described in the section before, could be *CoDaCoRe*. Published by Gordon-Rodriguez et al in 2021 in the paper “Learning Sparse Log-Ratios for High-Throughput Sequencing Data”, *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.

Thus, *CoDaCoRe* identifies a sequence of balances, in decreasing order of importance, each of which is sparse and interpretable. This makes it 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.

# Implementation/Assessment

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. Several solutions like *CoDaCoRe* and *selbal* have been introduced by the scientific community, where they showed that they work more efficient than machine learning models. The goal of this project focuses on collecting insights on the performances of machine learning models in direct comparison with *CoDaCoRe*, but also comparing the influence of transformations on performance. Additionally, the goal should be also practicality. At the end it would be good to give some practical guidelines on how to employ machine learning models with compositional data.

Thus, four log-ratio transformations will be compared: CLR (centered log-ratio), and three ALR methods (worst, random and optimal). As mentioned in the introduction, those two seem to be the most promising, in terms of applicability and interpretability. Including worst and random ALRs should give insights of potential repercussions of using ALR wrong, or at least not in the most ideal way.

The performances will be directly compared to TSS-transformed data (total sum scaling transformation) and *CoDaCoRe* in the following conceptual Diagram

Description automatically generatedframework:

Figure : 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. It has been decided to use a Generalized Linear Model (glmnet) and a Support Vector Machine (svmRadial) as machine learning models representation. Glmnet represents a common linear machine learning model and svmRadial a non-linear one.

For *CoDaCoRe*, it was decided to train two models per run, one with lambda = 0 and the other with lambda = 1. In *CoDaCoRe* the parameter lambda controls the regularization strength of the model. Lambda = 1 applies the 1-standard-error rule in the discretization step of the log-ratio. This is typically a good choice, leading to models that are both sparse and predictive. Lambda = 0 corresponds to a “0 standard-error rule”, in other words choosing the log-ratio that minimizes cross-validation score (Gordon-Rodriguez et al. 2021). Using both models allows for a direct comparison in how CoDaCoRe handles different types of microbiome data sets, that are typically very sparse. Additionally, CoDaCoRe allows in prediction tasks to include all found log-ratios or just the most descriptive one. It was decided to use both to see how influential one found log-ratio can be for the chosen data sets.

Performance assessment scores are necessary to compare all models directly. For discrete parameters (e.g., two binary classes like “healthy” and “non-healthy”) AUROC will be used to assess model performances. AUROC tells about the model’s ability to discriminate between cases (positive examples) and non-cases (negative examples). An AUROC above 0.8 generally means excellent discriminatory ability for the model, whereas an AUROC of 0.5 corresponds to a coin flip (i.e., a useless model).

For continuous parameters (e.g., BMI, which can have any real number) a root mean squared error (RMSE) is used for performance evaluation. It is the [standard deviation](https://www.statisticshowto.com/probability-and-statistics/standard-deviation/) of the [residuals](https://www.statisticshowto.com/residual/). Residuals are a measure of how far from the regression line data points are and a RMSE consequently measures how spread out these residuals are. A value of 0 would indicate a perfect fit to the data. Contrary to the AUROC, the lower the RMSE, the better the model.

# Results

## INFLUENCE OF TRANSFORMATION ON PERFORMANCES

### Model performances in a Binary Classification Setting

Figure 7 shows the culminated results of the presented pipeline and combined in one figure to show the differences of machine learning and transformations on several data sets directly. CRC and PCOS use 10% abundance filtered sets, whereas EstMB uses a 90% filtered set. For all repeats of the models, the data split has been the same for all models and all transformations in the same data set. Here, all models used a binary classification setting to predict “healthy” or “not-healthy”, the disease differing for every data set.

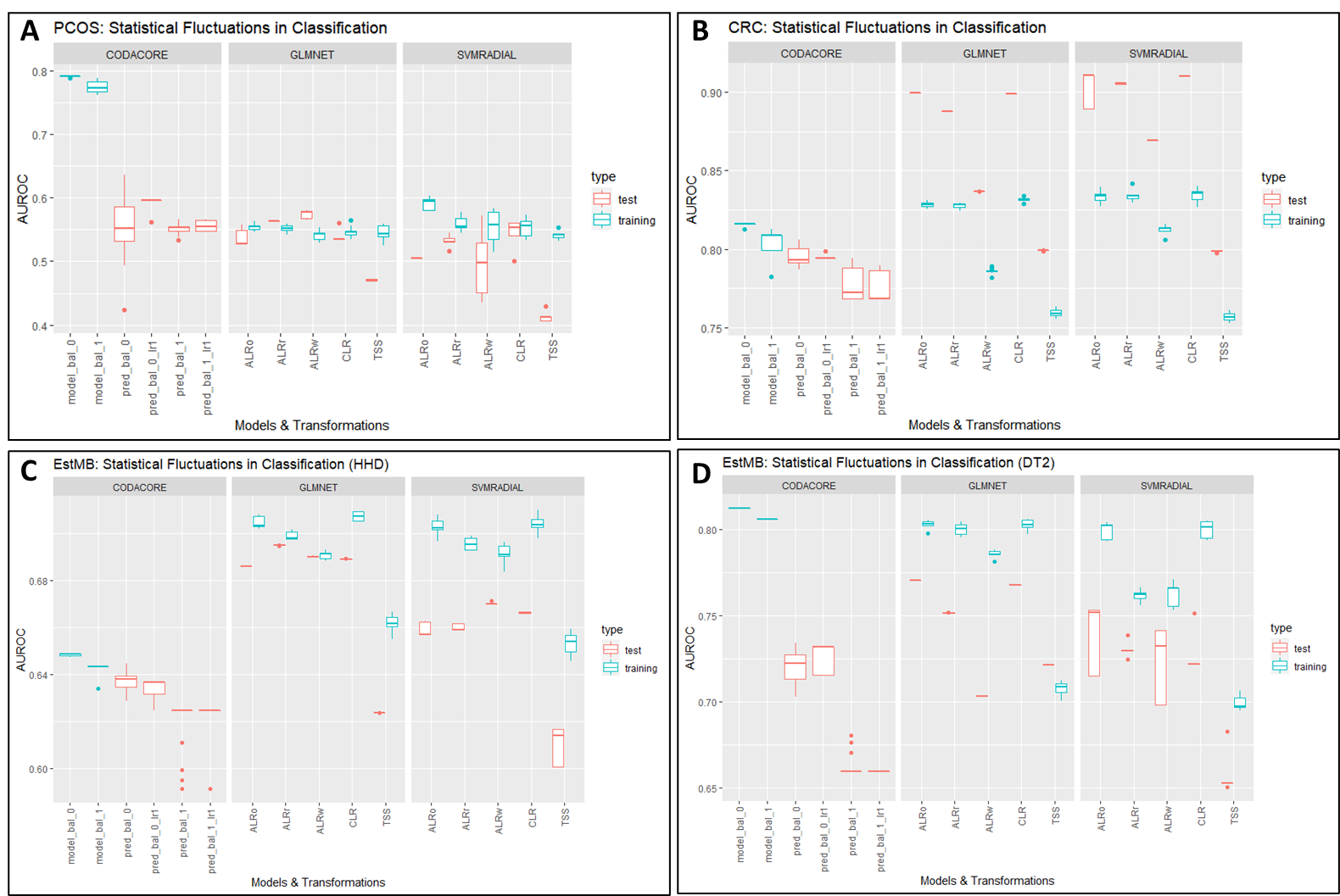


Figure : Statistical Fluctuations in Classification

CRC and PCOS use 10% abundance filtered sets, EstMB uses a 90% filtered set. The y-axis contains AUROC scores. The x-axis contains models and transformations, with the most left always being *CoDaCoRe*, the middle portion displaying performances of glmnet and the right svmRadial. In red test performances are plotted and in blue training performances. model\_bal\_0 and model\_bal\_1 correspond to *CoDaCoRe* models with lambda 0 and lambda 1, respectively. Consequently, pred\_bal\_0 and pred\_bal\_0\_lr1 correspond to predictions with all found log-ratios and only the most descriptive log-ratio. ALRo, ALRw and ALRr correspond to the types of ALR transformation applied in this pipeline (optimal, worst and random). C and D both contain data from EstMB data set, with C predicting for hypertensive heart disease (HHD) and D for diabetes type 2 (DT2).

7A shows the performances for PCOS. The AUROC scores for test and training performances vary from 0.5 to 0.6, with codacore training performances severely overfitting at 0.8. It is not surprising that all models have difficulties predicting PCOS, as it is the data set with a very low correlation between microbiome and the disease PCOS itself. Contrary to that, B shows the performance of models in a data set with a high correlation between the microbiome and colorectal cancer. CoDaCoRe shows test and training performances around 0.8, whereas both glmnet and svmRadial differing clearly. Their training performances vary from 0.75 up to 0.85 and their test performances spring from 0.8 to 0.9 depending on the transformation. Here, glmnet and svmRadial are clearly underfitting, a trend that has also been shown in the Holdout test. C and D both show performances from the EstMB data set, however predicting different diseases. C shows the performances for hypertonic heart disease and D for diabetes type 2. For HHD, codacore shows test and training performances ranging from 0.6 to 0.64, with the glmnet and svmRadial models again differing. Glmnet shows performances around 0.68 AUC for both training and test set. svmRadial shows similar performances, with a higher divergence between training and test set. In both models, TSS shows significantly lower performances, but both glmnet and svmRadial clearly overfit.

For DT2, all three models are overfitting. Test performances for all three models lay around 0.65 to 0.75, with the training performances ranging around 0.8.

Interestingly, in all data sets and all models that use different transformations, TSS seems to be clearly underperforming, with ALRw (ALR worst) behind it, especially visible in the CRC data set. This is also true for the PCOS data set, where an increase in performance is not depending on the transformation, as the underlying correlation has not been given in the first place. This seems to be a clear indication, that log-ratio transformations can indeed affect the performance in machine learning models.

### Regression Setting

A similar conclusion can be drawn for a regression setting. Figure 8 shows the culminated results for all three data sets in a regression setting. Contrary to the classification setting, all three data sets contained BMI as the continuous variable.

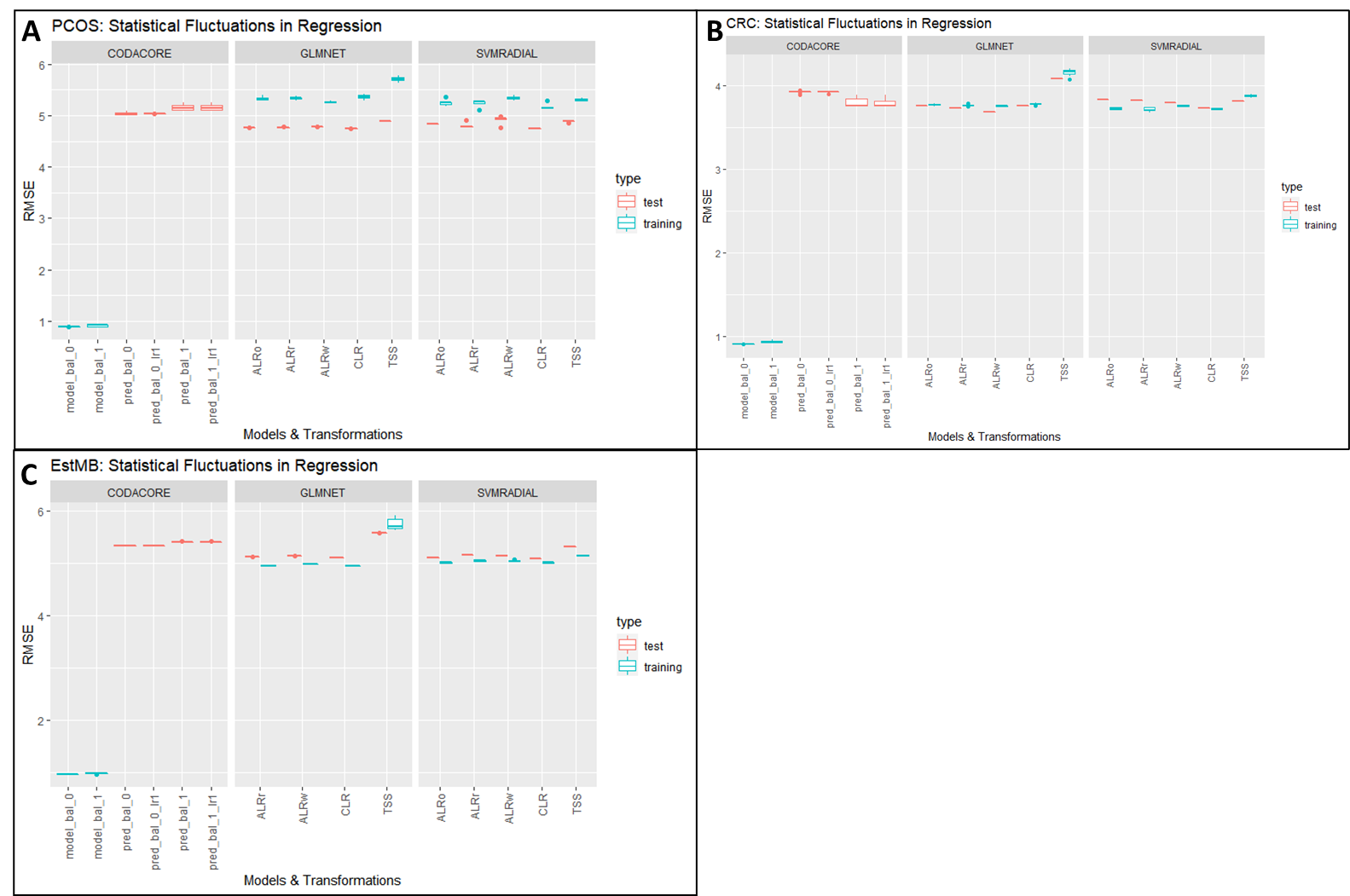


Figure : Statistical Fluctuations in Regression

CRC and PCOS use 10% abundance filtered sets, EstMB uses a 90% filtered set. The y-axis contains RMSE scores. The x-axis contains models and transformations, with the most left always being *CoDaCoRe*, the middle portion displaying performances of glmnet and the right svmRadial. In red test performances are plotted and in blue training performances. model\_bal\_0 and model\_bal\_1 correspond to *CoDaCoRe* models with lambda 0 and lambda 1, respectively. Consequently, pred\_bal\_0 and pred\_bal\_0\_lr1 correspond to predictions with all found log-ratios and only the most descriptive log-ratio. ALRo, ALRw and ALRr correspond to the types of ALR transformation applied in this pipeline (optimal, worst and random). C and D both contain data from EstMB data set, with C predicting for hypertensive heart disease (HHD) and D for diabetes type 2 (DT2).

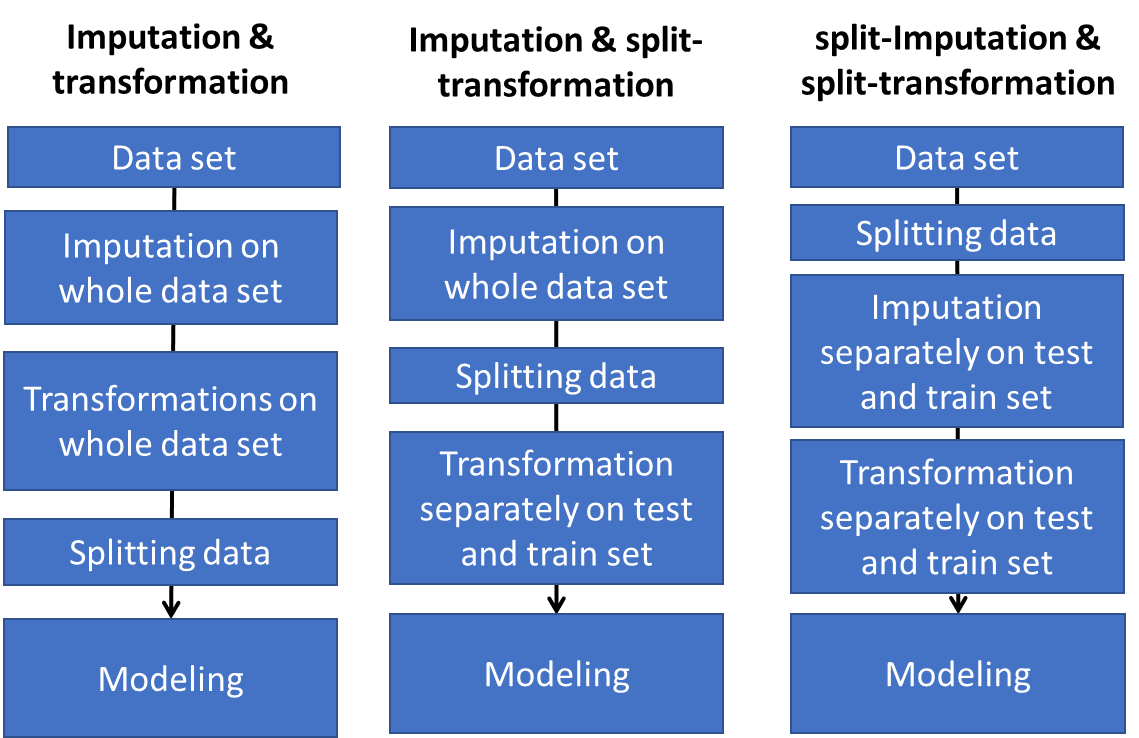
For all sub-graphs in Figure 7 the x-axis contains all models and transformations, with the most left always being codacore, the middle portion displaying performances of glmnet and the right svmRadial. It is very interesting that all three data sets, having a completely different bioinformatics and sample pipeline in the background and containing very different abundances for species in their data set, show a very coherent picture for BMI prediction. In all data sets, PCOS (8A), CRC (8B), and EstMB (8C) show the same behavior over all three models. CoDaCoRe is overfitting, with a training performance around 1.0 and test performances around 4.0 to 5.0. Glmnet and svmRadial have very close test and training performances around 4.0 to 5.0, with the exception of TSS for all three data sets, showing a bit higher test and training performances overall.

## INFLUENCE OF DATA LEAKAGE ON PERFORMANCES

It has to be acknowledged that this conceptual pipeline is considered “leaky” as the imputation procedure use the whole data set for zero replacement. Only afterwards is the data set split into training and test set, which makes the test set not totally unbiased. As not all data sets are big enough to allow a separate imputation for the small test set, it was therefore decided to conduct imputation on the whole data set. However, an additional test to observe the impact of data leakage on transformations and machine learning models is conducted separately.

Additionally, as the study design for the CRC data set had the concept of holdout test sets in mind, this opens the possibility of comparing model performances for holdout sets versus a classic 80/20 split, as the latter is planned for the general pipeline to allow better comparison between model performances.

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. In the case of ALR, the reference was found on the training set and directly applied to the test set. 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.

This test is very interesting, in particular because CLR and ALR\_optimal in theory have different influences on data leakage. CLR is usually conducted row-wise and would therefore not impact data leakage in any form. Contrary to that, ALR is conducted column-wise and could potentially show the same problems as imputation.

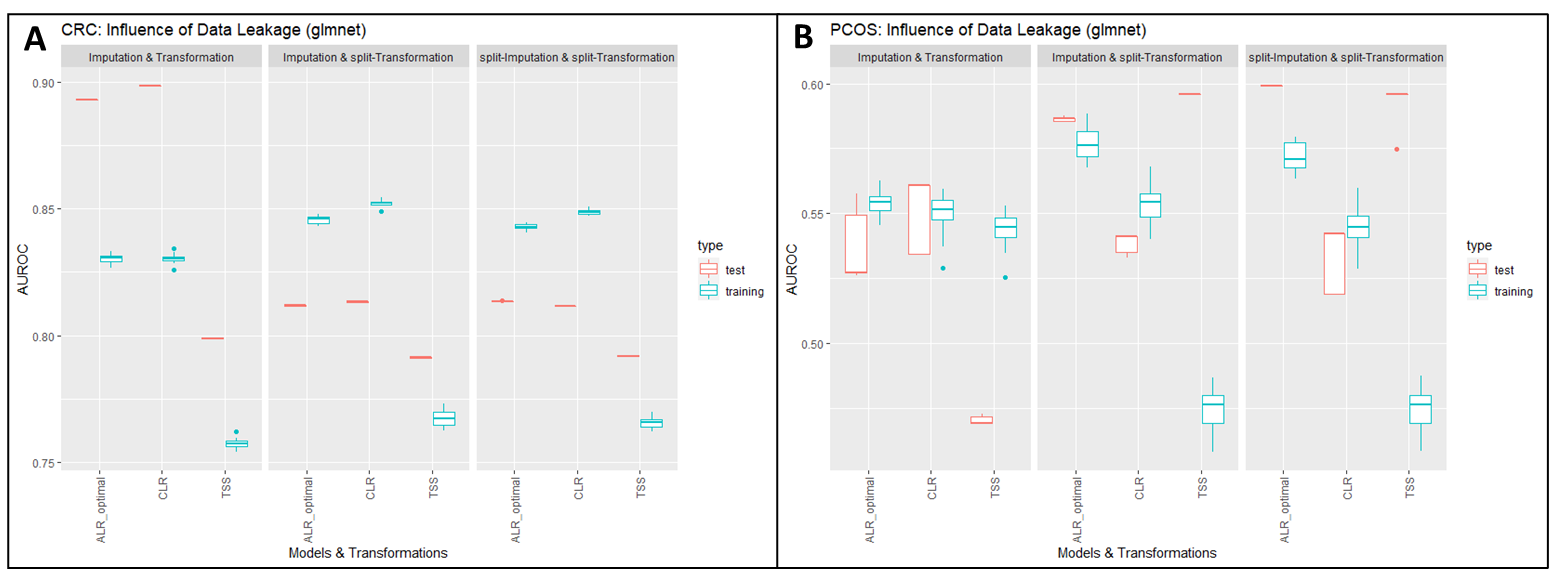


Figure 9: Influence of Imputation and Transformation

For B, split-imputation produced an error in the package zCompositions, therefore all zeros were replaced with a value of 0.5. The machine learning model used was glmnet.

Figure 9A and B show the effect of imputation and transformation procedures on a data set with high correlation (CRC) and with low correlation (PCOS). On the left side, the results after a leaky pipeline is shown, the middle part shows the effect of imputation on the whole data set and the right side shows the results if the data has been split before imputation. It should be noted that the split-imputation for PCOS produced an error, as the test set of this data set was too small to use GBM for imputation. Therefore, all zeros have been replaced by 0.5.

The results in 9A show that the perfect non-leaky pipeline does not affect the performance of machine learning models significantly, compared to imputation on the whole data set. In both cases training performances vary around 0.85, with TSS showing worst performances around 0.75. Test performances revolve around 0.8 for all transformations. However, the difference to imputation and transformation on the data set is very significant. As mentioned in Section () that introduced the results of the pipeline, all models seem to underfit heavily. This issue is resolved by at least splitting the data set before transformation. Subsequently, the test performances of the pipeline are very high, because information from the test set has been used for transformation and Imputation. Interestingly, this seems to be also the case for CLR, which is performed row-wise and should have no impact on the performance on the test set.

Figure 9B shows the effect on a leaky and non-leaky pipeline on small data sets with low correlations. Here, imputing and transforming on the same data set seems to be more beneficial, as test and training performances both show values around 0.55. Contrary to that, splitting the data before transformation increases the discrepancy between all three transformations. ALR performs best with test and training performances around 0.58 and low variances, followed closely behind by CLR with 0.55 for both test and training performances. TSS performs worse with training performance at 0.52 and test performance around 0.6. A similar picture is drawn by split-imputation and split-transformation. CLR is not very impacted by the different pipelines, which is in line with the theory. ALR shows a better performance when the data is split before transformation, with split-imputation creating a higher bias.

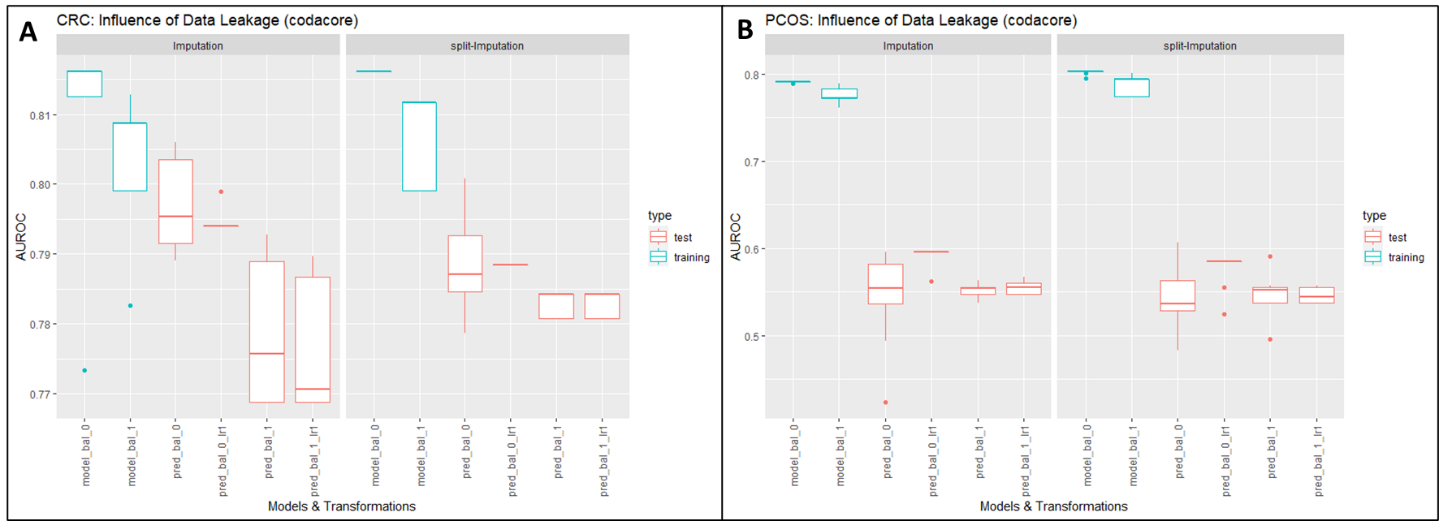


Figure 10 shows the same leakiness concept for codacore. Here, the data was split before imputation (right) and imputation conducted on the whole data set (left), again for both CRC and PCOS. For CRC, imputation on the whole data set seems to lead to higher variances in test performances compared to split-imputation. For PCOS, a low correlation data set, the moment of imputation does not seem to impact the test and training performances at all. In all cases, CoDaCoRe is heavily overfitting the data.

As a summary, in the future, at least split-transformation is something that should be thought about. It does not impact the conclusion, that transformations do show better performances than usually normalized data, even if the data is split only after imputation and transformation. However, the models now show for transformation a clear overfitting, instead of underfitting and the distance between test and training performances is decreasing. It seems that a split imputation impacts the performances only marginally in both high and low correlation data, however, if one wants to use codacore, a split-imputation also seems to be more beneficial.

## INFLUENCE OF TEST SET ON PERFORMANCES

The pipeline used to compare all data sets and their performances includes a standard data set split of 80/20, meaning 80% of data goes into the training set, 20% of data into the test set. However, the CRC data set has been designed with holdout sets in mind and the original paper uses this technique. Applying Holdout to the other data sets would be very difficult, a standard 80/20 split would be more advantageous, as it makes the comparison between all data sets easier. Therefore, one of the first tests was to assess if using Holdout or an 80/20 split makes a big difference for the CRC data set.

The following figure shows the performances of the CRC data set in a glmnet model in a binary classification setting. All models have been repeated 10x to show statistical fluctuations. All five holdout sets have been used separately as test set. In 6A test performances of all different holdout sets are combined, whereas in 6B all performances can be seen in detail.

Graphical user interface, application

Description automatically generated with medium confidence

Figure 6: Comparison of Holdout vs. 80/20

The graph shows the performances of different test sets using the same glmnet model. A shows the performance of Holdout vs 80/20 with all test performances combined. In B all test performances are separated to show the differences in choosing a test set.

Figure 6A shows the difference of test performances as boxplots for different transformations of the same data set. The transformations here are for the moment secondary, but what can be seen clearly is that for all transformations, the holdout sets show a lot more variance in the test set performances and the maximal test performance around 0.85. Compared to the standard 80/20 split, which shows almost no variance at all and also higher test performances around 0.9. The training performances are similar for both procedures, around 0.79 to 0.84. It should however be noted that the distance between test and training performances is quite high in the standard 80/20 split and the model clearly underfitting. This is not the case for the Holdout set, where the test and training performances are a lot closer together and the model tends to overfit the data.

In Figure 6B it becomes clear why the test performances of the Holdout set have such a high variance. For all transformations and holdout sets, the test performance is sturdy. For example, marked in pink dots is the test set performance when USA has been chosen as holdout set. For all ten repeats of the model, the test performance is stable around 0.72. However, choosing CHI as the holdout set produces test set performances of 0.85. This shows that the big boxplots in 6A are not due to statistical fluctuations, but the choice of holdout set. Of course, the standard 80/20 split is heavily underfitting, ideally test and training performances should be close together, like the CHI holdout set. However, in order to compare all data sets, and the standard 80/20 split showing no degradation of performance, it has been decided to continuously use a standard 80/20 split for all data sets, including CRC. This divergence in test set performances is also visible in codacore (see Figure 7).

Chart, waterfall chart

Description automatically generated

The setup is the same as for Figure 6, however the x-axis now contains all different codacore models instead of transformation. It can be seen that Holdout and standard 80/20 show similar training performances around 0.8 AUROC score, with the test performances steadily decreasing until 0.78 for standard 80/20 and down to 0.70 for holdout. Again, the holdout test performance has a high variance also in codacore compared to 80/20. 7B plots again the details and the same results as before are visible. The high variance results in the choice of holdout set, with USA and AUS producing very low test performances and GER and CHI showing very high test performances. Standard 80/20 sits more in the middle. Interestingly, for lambda 0 the test performances fluctuate a bit more compared to lambda 1, which potentially shows higher predictive performances for lambda 1, as it is intended in the algorithm.

# Discussion/Conclusion

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

In the same paper they also mention that no “ratio” is necessary for isometric behavior -> next tests could include such a test to see if it works better/worse than imputation + ALRo

Doing so will often allow the algorithm to identify at least one predictive log-ratio, at the risk of overfitting the training data. Additional care must be taken in validating such log-ratios on held-out data.

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.

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.

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.

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

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

### Data set

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 between 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(-ish) 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 ( paper Kreete). 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

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 were removed in the 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 CRC keeps 650 features and PCOS 1154 features. Respectively, for 50% abundance filters CRC keeps 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. Even with 90% filtering, EstMB keeps 3062 features.

### 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. When using log-ratio transformation however, zeros lead to problems as one: the log(0) is undefined and secondly: when working with ratios, zeros cannot be in the denominator. 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”.

### 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 it is included in *CoDaCoRe*. It has been decided to use TSS (total sum scaling transformation), which is a standard approach to normalize 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 (random ALR), (ii) find the most optimal denominator (optimal ALR) and (iii) worst ALR denominator (worst ALR) 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

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.

Before modeling, the data set is split 80/20 with stratification with the package “tidymodels”. The train set given to the function run\_ml() from “mikropml” by source schlosslab. 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 SVMs (svmRadial). 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 repeated 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. For this algorithm, the filtered and imputed data set is used, without any transformation. The training set that is fed to the function codacore() contains the same samples as the other machine learning models. Two *CoDaCoRe* models are trained, one with lambda = 0 and the other with lambda = 1. 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.