www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

UHASSELT KNOWLEDGE IN ACTION

Maastricht University

Faculty of Sciences

School for Information Technology

Master of Statistics and Data Science

Master's thesis

Causal analysis of immune cell composition in systemic lupus erythematosus (SLE) disease.

Amber Huybrechts

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Bioinformatics

SUPERVISOR :
Prof. dr. Olivier THAS

SUPERVISOR :
Koen VAN DEN BERGE
Oliver DUKES

2023
2024

Faculty of Sciences

School for Information Technology

Master of Statistics and Data Science

Master's thesis

Causal analysis of immune cell composition in systemic lupus erythematosus (SLE) disease.

Amber Huybrechts

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Bioinformatics

SUPERVISOR :

Prof. dr. Olivier THAS

SUPERVISOR :

Koen VAN DEN BERGE

Oliver DUKES

Acknowledgement

Much gratitude goes out to professor Olivier Thas and my external promotors Koen Van den Berge and professor Oliver Dukes for all their advice and guidance during this thesis. Their faith in me and their willingness to share their knowledge have been crucial in the research of this topic. I am particularly appreciative of the time and energy they invested in responding to my questions and supporting my academic development, and for the opportunities they have given me.

I am very grateful of my friends Hanne and Eline for their encouragement throughout stressful periods and for their mental health support. Finally, I would like to thank my partner and family for their unconditional support and encouragement as I completed my thesis. I would not have made it this far without them.

Abstract

Analysis of single-cell sequencing data, in particular cell abundance data, involves issues regarding compositionality. Cell composition data contains only relative information due to limited throughput. Therefore an increase in one cell type might also be reflected in other cell types. This makes estimating causal disease effects in cell composition data rather complicated, especially in the presence of confounders. Using a case study, presented by Perez et al. [16], involving cell composition data of lupus patients from European and Asian ancestry, different methodologies are evaluated. Methods include Wilcoxon Rank Sum Test and LinDA, two methods commonly used in microbiome studies. They are compared with a method that is developed for the analysis of cell composition data, called voomCLR. Both LinDA and voomCLR start from an ordinary linear regression model with counts transformed using the Centered Log-Ratio (CLR) transformation. Both methods involve a correction on the effect size to account for compositionality. VoomCLR takes into account additional variability and uses weighted least squares using heteroscedasticity weights. Methods from the causal inference framework are evaluated as well, including inverse probability weighting and standardization. The performance of all methods is assessed and compared using nonparametric and parametric simulation studies. These simulation studies attempt to reflect the compositional nature of cell composition data and include confounding. Both LinDA and voomCLR seem to be the best performing methods for this kind of data, with comparable performance. Both methods do seem to control the FDR. However, voomCLR turns out to be more conservative than LinDA, resulting in a lower sensitivity. The final analysis of the original case study is performed using LinDA. According to this method, there seems to be an effect of lupus disease on the abundance of cell types cM and Prolif.

Contents

1 Introduction

1.1 Background

Single-cell RNA-sequencing (scRNA-seq) data is a technology that is used in many biological studies to evaluate gene expression in hundreds of cells simultaneously. These cells are extracted from tissue or blood samples and exist in various types and states.
The technology behind scRNA-seq has advanced over the years. The first generation used plate-based methods with high sensitivity but limited cell throughput. Second-generation methods employed microfluidics and microparticles, increasing throughput but requiring substantial investment. However, they faced limitations in cell selection due to the microfluidic device's cell-size constraints. The third generation introduced combinatorial barcoding, i.e. multiple rounds of barcoding, which avoids physical cell partitioning and expensive equipment, making scRNA-seq suitable for long-term studies and clinical samples.
Compared to bulk RNA sequencing, which measures the average gene expression across the entire cell population, scRNA-seq is able to examine gene expression at the single-cell level, providing insights into cellular responses to drug treatments and identifying relevant genes. Its clinical relevance lies in understanding disease mechanisms and predicting treatment responses [3].

To assess differences in cell type composition data, cell type labels need to be assigned to each cell. When analyzing these data, it is important to take into account the compositionality of the data. Analyzing cell composition data involves examining a count matrix with N rows representing samples (or patients in the context of the case study from Perez et al. [16] discussed in this thesis) and P columns corresponding to different cell types. A critical challenge in this context is the compositionality of the data.
Compositional data exist within a simplex, where a data point can be represented by a real vector with positive components that sum to a constant [15]:

$$\mathbb{S}^P = \left\{ x = [x_1, \dots, x_P] \in \mathbb{R}^P | x_i \geq 0, i = 1, \dots, P; \sum_{i=1}^{P} x_i = \kappa \right\}$$

with P equal to 11 in this case study and κ an arbitrary constant.
As a result, we observe relative abundance information for cell types rather than absolute abundance. In scRNA-seq, we only observe relative information due to limited throughput, i.e. the constant κ is arbitrary [14].

The compositional nature of the data has implications for interpretation. As the abundance of one cell type increases, it becomes easier to sample from that type, while other cell types become less accessible. This can create a misleading impression of changes in absolute abundance, potentially leading to more false discoveries.
To address this, our methodology needs to consider compositional effects. Specific statistical methods that consider the data's composition can help achieve accurate results.

To address compositionality, one approach is to transform the cell counts. Let Y_{ip} be the random variable representing the observed cell counts for cell type $p \in \{1, \dots, P\}$ in sample $i \in \{1, \dots, n\}$. One transformation proposed by Aitchison is the Centered Log-Ratio (CLR) transformation [8]. These CLR-transformed counts are defined by:

$$Z_{ip} = \log \frac{Y_{ip}}{\left(\prod_{p=1}^{P} Y_{ip} \right)^{1/P}} = \log \frac{Y_{ip}}{\exp(\frac{1}{P} \sum_{p=1}^{P} \log Y_{ip})}. \tag{1}$$

The CLR transformation thus involves the logarithm of the cell counts Y_{ip} divided by the geometric mean in the corresponding sample. To prevent issues with zero counts, a pseudo-count of 0.5 is added to each count before transformation. This is done very often in microbiome studies [12]. This transformation allows us to move the counts out of the compositional simplex space, while maintaining distances; the Aitchison distance between $\mathbf{x}$ and $\mathbf{y}$ equals the Euclidean distance between the CLR-transformed counts with the Euclidean distance between samples $\mathbf{Y_i}$ and $\mathbf{Y_j}$ defined as:

$$d_e(\mathbf{Y_i}, \mathbf{Y_j}) = \sqrt{\sum_{p=1}^{P} (Y_{ip} - Y_{jp})^2}. \tag{2}$$

Nevertheless, the CLR transformation still has a constraint: the sum of the transformed components is 0 by definition (see Appendix B.1 for a mathematical derivation).
However, by transforming the counts to the real space, metrics like the Euclidean distance become meaningful, while for untransformed counts they are misleading [17].

In addition to the issue of compositionality, overdispersion is also a concern in this type of count data. Overdispersion is caused by both biological variation and technical variation [7].

On the other hand, if one is interested in estimating causal effects, one should also take into account confounding [13]. In the context of this case study, we are interested in the effect of lupus disease on the cell composition, correcting for confounding of age and ancestry (see Figure 1).

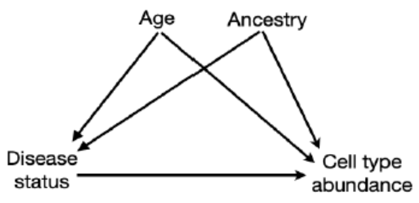


Figure 1: DAG case study. We aim for the estimation of a causal effect of disease status on cell type abundance, accounting for confounding effects from age and ancestry.

Say X_{ip} is the outcome of interest, which corresponds to the (absolute) abundance of cell type p in subject i, and A is a dichotomous exposure variable, which corresponds to lupus disease. In causal inference one uses the terminology of counterfactual outcomes or potential outcomes for $X_{ip}^{a=1}$ and $X_{ip}^{a=0}$. They represent the outcome X_{ip} under exposure $a = 1$ and $a = 0$, respectively (or in this case study for lupus patients and healthy controls, respectively). Only one of these counterfactuals is observed for each individual, namely the one corresponding to the actual exposure experienced by this individual. A causal effect of the exposure on the individual's outcome exists when $X_{ip}^{a=0} \neq X_{ip}^{a=1}$ for the individual. In general, identifying individual causal effects is not possible, so one often looks at aggregated causal effects, i.e. the average causal effect in a population of individuals. An average causal effect of the exposure A on the outcome X_p is present if $E(X_p^{a=1}) \neq E(X_p^{a=0})$ in the population of interest, with X_p the absolute count of cell type p in the population. The null hypothesis of no average disease effect in causal inference is formulated like this:

$$\begin{cases} H_0: & E(X_p^{a=1}) - E(X_p^{a=0}) = 0 \\ H_1: & E(X_p^{a=1}) - E(X_p^{a=0}) \neq 0 \end{cases}. \tag{3}$$

The expression $E(X_p^{a=1}) - E(X_p^{a=0})$ is referred to as an effect measure, more specifically the average treatment effect (ATE), with treatment being the disease. What we actually estimate is $E(X_p|A = 1) - E(X_p|A = 0)$, which is referred to as an association measure. The associational difference is estimated by the difference between the mean outcomes in the observations with $A = 1$ and $A = 0$, respectively.

In randomized experiments, due to the presence of exchangeability, one can infer the expected counterfactual outcome under exposure in the population ($E(X_p^{a=1})$) because it is equal to the expected outcome in the exposed ($E(X_p|A = 1)$). In randomized experiments we can say that association equals causation.

However, not always do we get to analyze a randomized experiment. Very often we need to analyze observational studies, like in this case study. To make causal interpretations in observational studies, there are three identifiability conditions that need to be satisfied:

1. Consistency
2. Exchangeability
3. Positivity

Consistency means that there only exists one type of the exposure, there don't exist multiple versions. We need a well-defined definition of the exposure we want to investigate and this should correspond to the exposure in the observed data. As the exposure in this case is the lupus disease, and this disease is known to exist in multiple forms and states [23], we can not be sure this assumption is not violated. We could assume that the different versions of the lupus disease result in the same potential outcome.

With the identifiability condition of (conditional) exchangeability, we assume that (within levels of confounders), the exposed and unexposed subjects are exchangeable. This means that the distribution of each of the potential outcomes would be the same in both exposure groups, within subgroups of the covariates L (i.e. age and ancestry). This gives the ability to look at the distribution of $Z_p^{a=1}$ in the lupus patients (conditional on covariates).

The identifiability condition of positivity assumes that in each level and combination of the variables (that are used to achieve exchangeability), both exposed and unexposed individuals are present. Each individual should in fact be able to experience every level of exposure, which is in this case the lupus disease. This condition is likely to be satisfied in this setting where the only confounders (assumably) are age and ancestry, and although in the data the diseased patients were typically older, this does not mean that younger individuals can't develop lupus. Violations of the positivity assumption are random in this case, not structural, due to limited sample size [13].

If the distribution of other variables differs between the exposed groups (so between lupus patients and healthy controls), and these variables are confounders, they also need to be adjusted for in the analysis. Two popular methods from the causal inference framework are explored in this thesis to deal with these confounders: inverse probability weighting [22] and standardization [19].

1.2 Research question

The purpose of this thesis is to evaluate the performance of different methodologies for analysis of compositional data, focusing on scRNA-seq data. The aim is to identify whether or not existing methodologies are able to deal with the issue of compositionality and to account for confounding. This in order to infer causal effects of the exposure on the cell type composition, where exposure in this case study corresponds to lupus disease. The goal is to assess the performance of methods borrowed from the analysis of microbiome data, including the Wilcoxon Rank Sum test [11] and LinDA [12], as well as a new method developed specifically for the analysis of cell type composition

data, called voomCLR [1].
We aim to investigate which methods are appropriate for identifying a (causal) disease effect. Causal in this setting means that any difference in (absolute) cell type abundance between healthy and lupus patients is due to the disease status. For this purpose we investigate the presence of and account for confounding, relying on a case study provided by Perez et al. [16]. As already indicated by Perez et al., but also investigated in this thesis during data exploration, age and ancestry are both considered as confounders. The aforementioned methods will also be compared with some well-known methods from the causal inference field, namely inverse probability weighting and standardization (or G-formula) [13].

1.3 Societal relevance and stakeholder awareness

Before the development of scRNA-seq, high-throughput sequencing techniques focused on extracting RNA from a tissue sample consisting of multiple cell types, i.e. bulk sequencing. The sequencing library in this context represents a population of cells. Now however, we are able to sequence individual cells and the sequencing library represents a single cell. This enables studying the transcriptome of different cells within the same tissue type. This technology is particularly useful in studying cancer immunology and the dissection of tumor heterogeneity. Tumors and the stromal component of tumors (i.e. connective tissue, blood vessels, inflammatory cells [9]), are a composition of different cancer cells developed from different genomic events (i.e. clones, tumor heterogeneity) and a mixture of cancer cells and immune cells [25].
Fields like immunology and oncology benefit from scRNA-seq by gaining a deeper understanding of cellular dynamics and interactions in order to develop effective treatments and improve patient outcomes [6].

Given the issue of compositionality and confounding, investigating which statistical methods are appropriate for analyzing scRNA-sequencing data will benefit future practices. To take advantage of this cutting-edge technology it is important to make appropriate choices regarding the analysis of such data.

1.4 Ethical considerations

The simulation studies conducted in this thesis use data from a case study presented by Perez et al.. This case study involves public data from both lupus patients and healthy controls. From all participants, informed consent was obtained [16].

2 Data

The analysis will be conducted on a case study presented by Perez et al. [16] considering healthy and diseased individuals of European or Asian ancestry. The disease being studied is Systemic Lupus Erythematosus (SLE), but will be referred to as 'lupus' in what follows. SLE is the most common type of lupus. It is a chronic autoimmune disease that comes with unpredictable disease flares and remissions. In autoimmune diseases the immune system does not recognize the difference between viruses, bacteria, germs etc. and your own healthy tissues. This leads to the immune system attacking and destroying your healthy tissue. During a flare, there is an increase in disease activity in one or more organ systems, caused by inflammation. The patients experience a return of the symptoms they have experienced before or develop new symptoms. Symptoms vary from fever to painful, swollen joints, an increase in fatigue, rashes, sores or ulcers in the mouth or nose and general swelling in the legs. There are no treatments to cure lupus, but there are treatments to manage the symptoms [23].

The original data contains 355 samples, from which 348 are of European or Asian ancestry. From

each patient peripheral blood mononuclear cells (PBMCs) were isolated. Following the analysis of Perez et al., we will only consider the samples from Asian or European ancestry, removing 7 samples from Hispanic and African American ancestry. Because of replicates in the data, these 348 remaining samples originate from 256 unique individuals. This means there are in total 92 replicates, originating from 68 individuals. 49 of these individuals appear twice, 14 individuals appear 3 times and 5 individuals appear 4 times. Among the samples, 145 samples are from healthy controls and 203 are from lupus patients. Lupus patients occur in 3 different groups: Managed, Flare and Treated. Samples in the Managed group belong to lupus patients whose symptoms are under control, so patients that are not in an active disease flare. The samples belonging to the Flare group belong to lupus patients that are in an active disease flare. For some of these patients there are also samples that belong to the Treated group, which are samples post-flare treatment.
This data set was pre-processed by my external promotor Koen Van den Berge. We use the cell type labels from the original publication, which considered 11 cell types, resulting in a count matrix with for each sample the observed number of cells for each cell type. Besides information on cell type, ancestry and disease status, also information on other variables is available. An extensive list of all the variables and their description is shown in Appendix A. The variables of interest during the analysis are the disease status (also referred to as SLE status), age and ancestry.

Frozen PBMCs were profiled in 23 pools across 4 processing batches. In the first batch, only healthy samples are included (see Appendix C.1). Within this batch there is one individual that has 2 replicates in this same batch. As one of these replicates had a total cell count of only 3, this sample is removed in further analysis.
In batch 2 and batch 4, both Healthy and Managed samples are included, whereas in batch 3 all groups are represented. All Flare and Treated samples consequently are only represented in batch 3.
There are also samples that were age and ancestry matched between batch 2 and batch 4 (26) and between batch 3 and batch 4 (4). These samples come from the same individuals.
By observing the ages of the different individuals in the data (so by not taking into account replicates), it came to the surface that there are two lupus patients in the data (1130_1130 and 1772_1772) that appear multiple times but with different ages in different samples. The age from these individuals is different across different batches, meaning that they have observations in multiple batches and that their age in each batch is different. Other replicates occur either in separate batches (with the same age of the patient) or within the same batch. As from the publication it was not clear how to interpret these replicates (biological or technical replicates), in the simulation study and further analysis was opted to work with only one replicate for each individual. This sample was chosen based on the sample with the most information content, meaning in this context the largest total cell count. I am aware of the loss of information, but this choice was made upon uncertainty about the source of replication. Also, for the purpose of simulation studies it is important to simulate realistic data, which can be achieved also without the replicates in the data.

3 Methodology

3.1 Hypothesis

Before conducting any test, it is important to know what hypothesis we want to test. In the context of cell composition analysis, we want to test the hypothesis of equal cell composition between groups. Since we have more than one cell type, we actually perform more than one hypothesis test. For each cell type in the data, we want to test the following hypothesis:

$$\begin{cases} H_0 : \mu_{p,\text{healthy}} = \mu_{p,\text{lupus}} \\ H_1 : \mu_{p,\text{healthy}} \neq \mu_{p,\text{lupus}} \end{cases} \tag{4}$$

where $\mu_{p,\text{healthy}} = E(X_p|A = 0)$ and $\mu_{p,\text{lupus}} = E(X_p|A = 1)$ indicate the expected absolute count of cell type p in healthy controls and lupus patients, respectively. However, as already mentioned before in the introduction of this section, we only observe relative abundances. Therefore testing the null hypothesis of equal absolute abundance might be too optimistic and not really feasible. Also the presence of confounding makes it difficult to test a marginal hypothesis. These are things to keep in mind when performing tests and interpreting results.

3.1.1 Multiple hypothesis testing

Since multiple hypotheses are tested (one for each cell type, which means in this case study 11 hypotheses), there is need for multiplicity correction to control the false discovery rate (FDR). The FDR is defined as the expected proportion of false positives among the positive findings [2]. The correction that is used in further analysis, is the p-value correction of Benjamini-Hochberg.

3.2 Wilcoxon rank sum test

The Wilcoxon rank sum test (also called Mann-Whitney U test) is a nonparametric test that is often used in microbiome studies to identify differentially abundant taxa [11]. Since microbiome studies also deal with compositional data, it might be a good idea to see how this method performs on scRNA-seq data. In the context of microbiome, this test is performed on normalized counts, for example total sum scaled (TSS) normalized counts. This normalization divides the counts by the total sum of counts in the corresponding sample. For this thesis, both TSS normalized counts and CLR counts will be used for comparison. The Wilcoxon test tests the null hypothesis that the two populations have the same distribution. If this null hypothesis is rejected, there is evidence that the distribution of one population is different. In fact, the null hypothesis can be formulated as:

$$H_0 : P(X_{p,\text{lupus}} < X_{p,\text{healthy}}) = \frac{1}{2} \tag{5}$$

where $X_{p,\text{lupus}}$ and $X_{p,\text{healthy}}$ represent either the TSS or the CLR-transformed counts of cell type p in the lupus and healthy population, respectively. The Wilcoxon rank sum test is used to compare two groups of independent samples. In this case study, this method compares the samples from healthy controls with the samples from lupus patients. Instead of looking at the values of the counts or relative abundances itself, this method uses ranks.
Advantages of this method are that it is less sensitive for outliers since it ranks the values and does not look at individual values. Another advantage is that it makes no distributional assumptions. Disadvantages on the other hand are the fact that this method does not take into account the compositionality and is not designed to account for confounders.

3.3 Linear regression

Ordinary linear regression with the CLR-transformed counts as outcome is one of the methods under evaluation. The model is formulated as follows:

$$Z_{ip} = \beta_0 + \beta_1 L_{1i} + \beta_{2p} L_{2i} + \beta_{3p} \cdot A_i + \varepsilon_{ip} \tag{6}$$

with

- Z_{ip}, the CLR-transformed count of the the observed (relative) abundance count from cell type p in sample i
- L_{1i}, the age from sample i
- L_{2i}, the ancestry from sample $i = \begin{cases} 1, & \text{if sample } i \text{ belongs to patient of European ancestry} \\ 0, & \text{if sample } i \text{ belongs to patient of Asian ancestry} \end{cases}$
- A_i, the SLE status from sample $i = \begin{cases} 1, & \text{if sample i belongs to lupus patient} \\ 0, & \text{otherwise} \end{cases}$
- ε_{ip}, the error term, assumed to be normally distributed with mean zero and constant variance.

Two methods will be evaluated that use an extension of the same linear model.

3.3.1 LinDA

LinDA, or linear models for differential abundance analysis, is a method that is developed for the analysis of microbiome compositional data. Essentially this method requires fitting linear regression models on the CLR-transformed data, applying a bias correction to account for compositional effects [12].

This method can also be applied on scRNA-seq cell type abundance data. After transforming the data using the CLR transformation, linear regression models are fitted using the CLR-transformed abundance data as the response (as in equation (6)). This means that we can use the flexibility of linear models to include confounders as covariates in the model.
The effect of interest is the effect of the disease status, so the parameter β_{3p}. Actually, we want to estimate the effect of the disease status on the absolute count rather than on the CLR-transformed abundance. The estimate $\hat{\beta}_{3p}$ of β_{3p} is biased with respect to the effect sizes one would obtain based on the absolute abundances (see Appendix B.2). That is why linDA uses a bias correction approach that is based on the mode of the effect size across all cell types. The bias correction makes use of the assumption that most cell types are not differentially abundant by substracting the mode of the regression coefficients. That means that we estimate the effect of disease on the absolute abundance of each cell type by

$$\hat{\alpha}_{3p} = \hat{\beta}_{3p} - \tilde{\beta}_3 \tag{7}$$

with $\tilde{\beta}_3$ equal to the estimate of the mode of the $\hat{\beta}_{3p}$ coefficients. We can now test the null hypotheses $H_{0,p} : \alpha_{3p} = 0$ with α_{3p} the effect size of disease on the absolute abundance of cell type p.
Before we can perform this hypothesis test, we need an estimator of the variance of $\hat{\alpha}_{3p}$ to construct a test statistic. The variance of $\hat{\alpha}_{3p}$ can be estimated by:

$$\widehat{\text{Var}(\hat{\alpha}_{3p})} = \widehat{\text{Var}}(\hat{\beta}_{3p}) + \widehat{\text{Var}}(\tilde{\beta}_3) - 2\widehat{\text{Cov}}(\hat{\beta}_{3p}, \tilde{\beta}_3) \approx \widehat{\text{Var}}(\hat{\beta}_{3p}) \tag{8}$$

since Zhou et al. argue that $\widehat{\text{Var}}(\hat{\beta}_{3p})$ dominates $\widehat{\text{Var}}(\tilde{\beta}_3)$ and $\widehat{\text{Cov}}(\hat{\beta}_{3p}, \tilde{\beta}_3)$ as $n, P \to \infty$ under mild conditions. $\widehat{\text{Var}}(\hat{\beta}_{3p})$ is the OLS variance, that we now define as $\hat{\sigma}^2_{3p}$.
LinDA ultimately uses the studentized statistic

$$T_p = \frac{\hat{\alpha}_{3p}}{\hat{\sigma}_{3p}}. \tag{9}$$

This statistic is asymptotically normal, but for small samples, the t-distribution provides a better approximation to the sampling distribution of T_p. The p-value for testing $H_{0,p}$ is defined as

$$p_p = 2F_{n-4}(-|T_p|) \tag{10}$$

where F_{n-4} denotes the cumulative distribution function of a t-distribution with n-d-2 degrees of freedom, with d=2 the number of covariates to adjust for.

3.3.2 voomCLR

Similar to linDA, voomCLR uses CLR transformations for fitting linear models and applies bias correction to the effect sizes. However, this method extends this approach in several ways [1].
Counts typically have a mean-variance relationship, but even after the CLR transformation the variance is a function of the mean, meaning that the cell type counts are still heteroscedastic post-transformation. Compositional transformations are thus not variance-stabilizing. VoomCLR uses heteroscedasticity weights by building on the limma-voom framework from Law et al. [5] to account for counts' mean-variance structure. Where in the limma-voom framework the mean-variance trend is estimated using a loess curve, voomCLR allows to calculate weights analytically using the Delta method [10]

$$\text{Var}(f(X)) = \text{Var}(X) \cdot f'(E(X))^2,$$

assuming either a Poisson distribution, although this might be too restrictive, or a negative binomial distribution. This is useful because we only have a limited number of cell types, which leads to uncertain empirical estimation of the mean-variance trend. Applying these heteroscedasticity weights, linear models are fitted using weighted least squares. This is thus the first extension to the linDA approach; using heteroscedasticity weights to apply weighted least squares when fitting the linear models for each cell type.

Whereas linDA assumes that the uncertainty on the bias term is negligible as compared to the uncertainty of the (uncorrected) effect size, voomCLR also accounts for the sampling variability involved in estimating the bias correction term by adopting a bootstrapping approach. This uncertainty exists because in cell type composition analysis the number of cell types is typically limited, so you can not properly rely on the assumption that is made in equation (8) for statistical inference. Therefore a solution is to adopt a non-parametric bootstrap procedure for each (linear combination of) parameter(s) of interest, say β_{3p}, by resampling $\hat{\beta}_{3p}$ across p with replacement. For each bootstrap sample b the mode $\check{\beta}_{3b}$ is calculated, which is an estimate for the bias in that bootstrap sample. The variance σ^2_{bias} of the bias term $\tilde{\beta}_3$ is approximated by

$$\widehat{\text{Var}}(\tilde{\beta}_3) = \frac{1}{B-1}\sum_{b=1}^{B}(\check{\beta}_{3b} - \bar{\beta}_3)^2$$

with $\bar{\beta}_3 = \frac{1}{B}\sum_{b=1}^{B}\check{\beta}_{3b}$ and B the number of bootstrap samples.
This term is added to the denominator of the moderated t-statistic from limma. Moderated t-test statistics are generated using empirical Bayes for shrinking linear model residual variances towards a common value across cell types. The moderated t-statistic is calculated as follows:

$$T_p = \frac{\hat{\alpha}_{3p}}{\tilde{\sigma}^2_{\text{limma}} + \hat{\sigma}^2_{\text{bias}}} \tag{11}$$

where $\tilde{\sigma}^2_{\text{limma}}$ is the squared standard error obtained with empirical Bayes. The p-value is calculated as follows:

$$p_p = 2 * F_{\text{df.residual+df.prior}}(-|T_p|) \tag{12}$$

where $F_{\text{df.residual+df.prior}}$ denotes the cumulative distribution function of a t-distribution with degrees of freedom the sum of the residual degrees of freedom (n-4) and the prior degrees of freedom obtained using empirical Bayes.

To summarize, both linDA and voomCLR fit linear models on CLR-transformed counts and apply bias correction on the effect sizes. LinDA applies ordinary least squares to fit these linear models, while voomCLR applies weighted least squares, with weights the inverse of observation-level variances that can be estimated analytically. Additionally voomCLR also accounts for uncertainty on the bias correction by applying a bootstrap approach to generate a moderated t-statistic.

3.4 Causal inference

This section includes two methods that are often used in the causal inference framework, aiming at the estimation of average treatment effects. Before introducing these methods, let us first formalize the identifiability conditions mentioned before. Note that in what follows, the outcome for which we are estimating a causal treatment effect is the CLR-transformed count of the observed (relative) abundance count Y_{ip} for cell type p in sample i, denoted as Z_{ip}. More formally, the average treatment effect of interest is

$$\text{ATE} = E(Z_p^{a=1}) - E(Z_p^{a=0}) \tag{13}$$

with $E(Z_p^{a=1})$ and $E(Z_p^{a=0})$ the expected CLR-transformed count of cell type p in the lupus population and healthy population, respectively.

- **Consistency**
  Consistency requires that the potential outcome for exposure is equal to the outcome when exposed, i.e.
  $$E(Z_p^{a=1}) = E(Z_p^{a=1}|A=1) = E(Z_p|A=1). \tag{14}$$
- **Exchangeability**
  Exchangeability means that the counterfactual outcome and the actual exposure are independent, i.e. $Z_p^a \perp\!\!\!\perp A$ for all a. Under exchangeability we have
  $$E(Z_p^a|A=1) = E(Z_p^a|A=0) = E(Z_p^a). \tag{15}$$
  When the exposure is assigned randomly, which particular group received the treatment is irrelevant for the value of $E(Z_p|A=1)$ and $E(Z_p|A=0)$. However, in an observational study like this case study, the exposure is not assigned randomly and often influenced by other covariates or confounders. A more relaxed assumption is conditional exchangeability, where the counterfactual outcome in a level of L, with L the confounders, and the actual exposure are independent, i.e. $Z_p^a \perp\!\!\!\perp A|L$ for all a. Under conditional exchangeability we can write
  $$E(Z_p^a|A=1,L) = E(Z_p^a|A=0,L) = E(Z_p^a|L). \tag{16}$$
- **Positivity**
  The positivity assumption requires that each exposure is observed in each observed stratum l of L, i.e.
  $$P(A=a|L=l) > 0, \text{ for all } l \text{ with } P(L=l) \neq 0 \text{ in population of interest.} \tag{17}$$

That means that under the assumption of consistency and conditional exchangeability, one can write

$$E(Z_p^a|L=l) = E(Z_p|A=a, L=l). \tag{18}$$

An estimate for a causal difference (or ATE) can only be obtained under positivity, additional to the assumptions of consistency and conditional exchangeability, since one needs to estimate both $E(Z_p^{a=1}|L=l)$ and $E(Z_p^{a=0}|L=l)$. If the positivity assumption is violated, these conditional means are not well-defined [13].

3.4.1 Inverse probability weighting [13]

With inverse probability weighting (IPW), a pseudo-population is created in which each individual is represented in both exposure groups (i.e. healthy controls and lupus patients). This eliminates the effect of confounding in the sense that the exposure and the confounders become statistically independent in the pseudo-population (i.e. $L \perp\!\!\!\perp A$). This method relies on the condition that exposed individuals in $L = l$, had they been healthy, would have had the same expected outcome as those in $L = l$ that actually are healthy, i.e. conditional exchangeability $Z_p^a \perp\!\!\!\perp A|L$.
IPW uses inverse probability weights, calculated as $\frac{1}{f(A|L)}$ where $f(A|L)$ represents the probability distribution of belonging to exposure A (in our case lupus disease), given the covariates L (i.e. age and ancestry). Because one of the covariates is continuous, we have to resort to modeling, so these weights are obtained using logistic regression [22]:

$$\text{logit}(P(A_i = 1|L_i)) = \alpha_0 + \alpha_1 L_{1i} + \alpha_2 L_{2i}. \tag{19}$$

For each stratum in L (so for each combination of age and ancestry), one obtains estimates for $\hat{P}(A = 1|L)$. Each individual is weighted using the inverse of the probability that they are exposed, given their covariates.
In the pseudo-population, created by the estimated inverse probability weights, the difference $\hat{E}(Z_p|A = 1) - \hat{E}(Z_p|A = 0)$ is computed for each cell type p. If there indeed is no confounding for the effect of A in the pseudo-population and the model for $P(A = 1|L)$ is correct, association implies causation. In that case an unbiased estimator of the associational difference $E(Z_p|A = 1) - E(Z_p|A = 0)$ in the pseudo-population is also an unbiased estimator of the causal difference $E(Z_p^{a=1}) - E(Z_p^{a=0})$.

To estimate the causal difference, one fits the following marginal structural mean model:

$$E(Z_p^a) = \beta_{0p} + \beta_{1p} a. \tag{20}$$

Under the assumptions made, a consistent estimator for $\beta_{1p} = E(Z_p^{a=1}) - E(Z_p^{a=0})$ can be obtained by a consistent estimator $\hat{\theta}_1$ from the IP-weighted associational model:

$$\frac{E\left[\frac{I(A=a)Z_p}{f(A|L)}\right]}{E\left[\frac{I(A=a)}{f(A|L)}\right]} = \theta_{0p} + \theta_{1p} A. \tag{21}$$

Parameter estimates are obtained using weighted least squares with individuals weighted by their estimated (nonstabilized) inverse probability weights:

$$\widehat{W}_1 = \frac{1}{\widehat{P}(A = 1|L)} \text{ and } \widehat{W}_0 = \frac{1}{1 - \widehat{P}(A = 1|L)}. \tag{22}$$

Nonstabilized weights are opted because we are using a saturated model; we can not make the marginal structural mean model more complex than it is due to the fact that we have a binary exposure and no other covariates. We thus estimate two parameters to estimate two quantities ($E(Z_p^{a=0})$ and $E(Z_p^{a=1}) - E(Z_p^{a=0})$). Statistical superiority (i.e. narrower 95% confidence intervals) of stabilized weights (where $P(A = a)$ is included in the numerator) only occurs when the (IP weighted) model is not saturated. In case of nonstabilized weights, the mean of the weights should be equal to 2, as this approach creates a pseudo-population twice the size of the original population.
The ATE can now be estimated as

$$\hat{\theta}_{1p} = \frac{\hat{E}\left[\frac{I(A=1)Z_p}{P(A=1|L)}\right]}{\hat{E}\left[\frac{I(A=1)}{P(A=1|L)}\right]} - \frac{\hat{E}\left[\frac{I(A=0)Z_p}{P(A=0|L)}\right]}{\hat{E}\left[\frac{I(A=0)}{P(A=0|L)}\right]} \tag{23}$$

under the assumption that in the generated pseudo-population there are no confounders, the model for $f(A|L)$ is correct and under the assumption of positivity.

Under positivity, $E\left[\frac{I(A=a)}{f(A|L)}\right] = 1$ and $\hat{\theta}_{1p}$ is an (asymptotically) unbiased estimator of $E\left[\frac{I(A=1)Z_p}{P(A=1|L)}\right] - E\left[\frac{I(A=0)Z_p}{P(A=0|L)}\right]$. Under the assumption of conditional exchangeability and consistency, $\hat{\theta}_{1p}$ is therefore an unbiased estimate of the ATE $E(Z_p^{a=1}) - E(Z_p^{a=0})$, i.e. β_{1p}.
The variance of $\hat{\theta}_{1p}$ is estimated using a robust variance estimator [22] (or alternatively using nonparametric bootstrap).

3.4.2 Standardization [13]

An alternative for inverse probability weighting is standardization. The standardized mean for exposure a is calculated as

$$\sum_l E(Z_p|A=a, L=l)P(L=l). \tag{24}$$

When L is continuous, this sum is replaced by an integral and $P(L=l)$ is replaced by the probability density function $f_L(l)$.
Under the assumption of conditional exchangeability and consistency we have

$$E(Z_p|A=a, L=l) = E(Z_p^a|L=l). \tag{25}$$

$E(Z_p|A=a, L=l)$ is only well-defined when $P(A=a|L=l) > 0$ for each l with $P(L=l) \neq 0$, i.e. under the assumption of positivitiy.
Under these assumptions, the standardized mean is a consistent estimator of the expected outcome if everyone had been diseased ($E(Z_p^{a=1})$). Analogously in healthy controls, the standardized mean outcome in the healthy controls is a consistent estimator of the expected outcome if everyone had been healthy ($E(Z_p^{a=0})$).

To compute the standardized mean outcome in the lupus patients (or in the healthy controls), we require two things: the conditional means in each stratum l of the confounders L $E(Z_p|A=a, L=l)$ and weights as the prevalence of each value l in the study population $P(L=l)$. We have to resort to modeling since we have a continuous covariate and therefore more strata than observations in our study.

To obtain parametric estimates for the conditional mean, a linear regression model is fitted for the mean outcome with disease A and all confounders (age and ancestry) in L included as covariates. Essentially the same model as in equation (6) is fitted. Then we obtain an estimate $\hat{E}(Z_p|A=a, L=l)$ for each combination of values A and L and therefore for each of the individuals in the study population.
Estimating $P(L=l)$ nonparametrically from the data by dividing the number of individuals in the strata defined by $L=l$ by the total number of individuals in the population is not feasible due to the high number of strata. However, $P(L=l)$ does not need to be estimated explicitly. We only need to estimate $E(Z_p|A=a, L=l)$ for the l value of each individual i in the study and then compute the average

$$\hat{\theta}_{ap} = \frac{1}{n}\sum_{i=1}^{n} \hat{E}(Z_{ip}|A_i=a, L_i) \tag{26}$$

since the weighted mean $\sum_l E(Z_p|A=a, L=l)P(L=l)$ can also be written as the double expectation $E(E(Z_p|A=a, L))$.

The ATE is estimated by

$$\hat{\theta}_p = \hat{\theta}_{1p} - \hat{\theta}_{0p} = \frac{1}{n}\sum_{i=1}^{n} \hat{E}(Z_{ip}|A_i=1, L_i) - \frac{1}{n}\sum_{i=1}^{n} \hat{E}(Z_{ip}|A_i=0, L_i). \tag{27}$$

The standard error of $\hat{\theta}_p$ is calculated analytically using a robust standard error [19].
P-values can be obtained using this test statistic:

$$T_p = \frac{\hat{\theta}_p}{\sqrt{\mathrm{Var}(\hat{\theta}_p)}} \sim t_{n-2} \tag{28}$$

to test the null hypothesis $\theta_p = 0$ of no average treatment effect for cell type p.

3.5 Simulation

To assess the performance of the different methods, a nonparametric and a parametric simulation study was set up reflecting the data of the case study. The advantage of using a simulation study is that one knows the truth; i.e. one knows in which cell types there is a difference in the abundance caused by the disease state.

3.5.1 Nonparametric simulation

The goal of the nonparametric simulation is to sample observations from available data, without making any assumptions about the distribution. The simulated data should reflect a difference in the distribution of ancestries and in the distribution of age. The two groups under comparison should be comparable, had both groups been healthy, conditional on the age and ancestry. For this purpose, the simulation uses only healthy observations, as these are assumed to be comparable conditional on the confounders. After introducing a disease effect on the count of some of the cell types in the second group, we can identify the average disease effect for each cell type. Introducing a disease effect is referred to as introducing a signal.

Two groups (n=45 each) are sampled to represent the healthy controls (group 1) and the lupus patients (group 2). The effect size of interest is the disease effect. During simulation, it is important to know for which of the cell types there is a disease effect. At the same time, we have to take into account confounding. The first step is to create two groups that only differ in the distribution of age and ancestry. After these groups are created, a signal in the second group is introduced. In randomly sampled cell types, the cell counts are replaced with the cell count of another cell type from the same observation.

The first step consists of first creating two groups with equal age distribution and the same ratio of Europeans and Asians, followed by exchanging observations between the two groups to create the imbalance. More formally, the data is divided in different strata defined by the age and ancestry. These strata are determined by quantiles of age within each ancestry (see Table 1).

Table 1: Age categories per ancestry based on quantiles in healthy population.

Ancestry	Quantiles (0% - 20% - 40% - 60% - 80% - 100%)
Asian ancestry	21.0 - 26.6 - 31.2 - 48.8 - 59.8 - 74.0
European ancestry	23.0 - 26.0 - 29.0 - 33.0 - 42.2 - 75.0

From each stratum (i.e. quantile), the observations are randomly split into the two groups. In this way, the age distribution should be approximately the same and the ratio of Asians and Europeans is exactly the same.

To incorporate the confounding nature of the original data in the simulated data, some manipulations are done on the obtained split. A certain number of samples are exchanged between the two groups. Essentially some of the older observations in the first group are exchanged with some of the younger observations in the second group, from which 2/3 of the older observations belong to Asian samples and 1/3 to European samples. This to create an imbalance between the age distributions and the ratio of the ancestries, without consistently having only older Asians in the second group. One disadvantage is that also the Europeans end up imbalanced. This is not the case in the original data.

After these manipulations, the only difference in cell composition between the groups are caused by the confounders. A disease effect is simulated by introducing a signal in some of the cell types. This signal is introduced in the second group, which is meant to represent the group of lupus patients. A signal is obtained by replacing the cell count of one cell type with the cell count of another cell type within the same sample [1]. For this purpose a sampling distribution is generated to sample pairs (p, q) of cell types, in which p should be the cell type in which a signal should be introduced, and q the cell type from which the count will be used to replace the original count. In other words, after signal introduction in cell type p, Y'_{ip} is equal to Y_{iq} with Y'_{ip} the count of cell type p in sample i after introduction of disease effect and Y_{iq} the count of cell type q in sample i that is used as replacement. This sampling distribution is based on the Euclidean distances between the cell type count vectors from CLR-transformed counts. We want to make sure that replacing a count will indeed introduce signal, but at the same time that this signal is realistic (i.e. the count of a rare cell type should not be replaced by the highest count and vice versa). That is why the probabilities are calculated inversely proportional to the Euclidean distance:

$$P(p,q) = \frac{1/\text{Euclidean distance(p,q)}}{\sum_k^K 1/\text{Euclidean distance}(p_k, q_k)} \tag{29}$$

with K the total number of pairs (p, q), which is equal to 55. Say we want to simulate a disease effect in three cell types, then three pairs of cell types are sampled in each iteration.

As replacing one count will lead to a change in the total sum of counts in each sample, a compositional correction is applied to maintain the total counts. This compositional correction is based on the relative proportion of each cell type using weights. To avoid this problem, we could also swap the cell counts of cell types p and q, leaving the total count constant and introducing signal to two cell types at once. However, this does not simulate the reality of compositionality, where other cell types need to compensate for changes in one cell type.

This compositional correction is applied as follows. After replacing a cell count, the difference d_i in the total sum count is calculated for each sample i. d_i is equal to the difference between the original count of cell type p and the cell count of the replacement cell type q in sample i:

$$d_i = Y'_{ip} - Y_{ip} = Y_{iq} - Y_{ip}. \tag{30}$$

For each other cell type, weights are calculated based on their proportion in the sample (based on the original total count). Say N_i is the total sum of the cell counts from sample i (before replacing the cell count). The proportion R_{ik} of cell type k in sample i is then defined as

$$R_{ik} = \frac{Y_{ik}}{N_i}. \tag{31}$$

The weights W_{ik} for each cell type k (with $k \neq p$) are calculated as

$$W_{ik} = \frac{R_{ik}}{\sum\limits_{j \neq p}^{11} R_{ij}}. \tag{32}$$

To apply the compositional correction, one subtracts $W_{ik} * d_i$ from the original count. This ensures that the total sum remains the same and that the other cell types compensate for the change in abundance proportional to their relative abundance.

The choice for applying a compositional correction is made because the purpose is to simulate data as realistic as possible. As already mentioned in the context of compositional data, when one cell type increases (or decreases) in abundance, other cell types compensate for this since we only obtain relative information. By keeping the sum constant, we simulate that from each observation the same number of information is sampled. By applying the compositional correction, we simulate the change that other cell types undergo when another cell type increases or decreases.

3.5.2 Parametric simulation

In parametric simulation, there are more options to make the simulation more flexible. For instance, we can vary the number of samples and the number of cell types studied. The scenarios used in the simulations are shown in Table 2 (although more options are possible).

Table 2: Different simulation scenarios used in parametric simulation. All scenarios are compared with and without accounting for confounding. Scenario A corresponds to the settings in the nonparametric simulation.

Scenario	Number of observations (n)	Number of cell types (P)	Number of differential cell types (k)
A	90	11	3
B	90	11	6
C	90	30	6
D	20	11	3

In each iteration, the first half of the n observations corresponds to the group of healthy controls and the second half corresponds to the group of lupus patients. Cell counts Y_{ip} are sampled for each observation i and cell type p using a multinomial distribution. The outline of this procedure is shown in Figure 2.

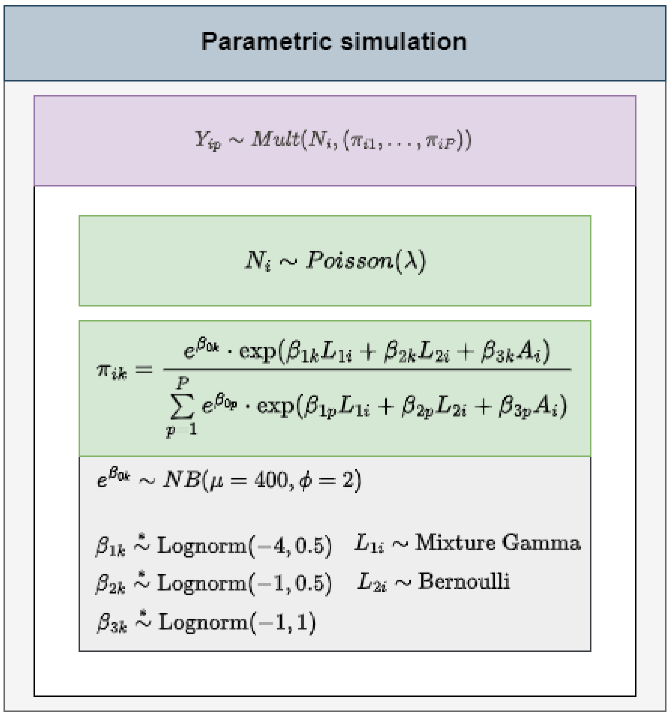

Parametric simulation
$Y_{ip} \sim Mult(N_i, (\pi_{i1}, \dots, \pi_{iP}))$
$N_i \sim Poisson(\lambda)$
$\pi_{ik} = \frac{e^{\beta_{0k}} \cdot \exp(\beta_{1k}L_{1i} + \beta_{2k}L_{2i} + \beta_{3k}A_i)}{\sum_{p=1}^{P} e^{\beta_{0p}} \cdot \exp(\beta_{1p}L_{1i} + \beta_{2p}L_{2i} + \beta_{3p}A_i)}$
$e^{\beta_{0k}} \sim NB(\mu = 400, \phi = 2)$
$\beta_{1k} \overset{*}{\sim} \text{Lognorm}(-4, 0.5)$ $L_{1i} \sim$ Mixture Gamma
$\beta_{2k} \overset{*}{\sim} \text{Lognorm}(-1, 0.5)$ $L_{2i} \sim$ Bernoulli
$\beta_{3k} \overset{*}{\sim} \text{Lognorm}(-1, 1)$

Figure 2: Parametric simulation framework. ** Some values for $\beta_{.k}$ are set to zero or multiplied by -1 at random to ensure that not all cell types have an effect size different from zero and allow both increasing and decreasing effect sizes. For β_{3k} a fixed number of values is set equal to zero to control the number of cell types with a disease effect.*

The age (L_{1i}) and ancestry (L_{2i}) distributions between both groups are simulated as close as possible to the original data. For the age distribution of both groups, a mixture of gamma distributions is used with different values of the parameters in each group (for a comparison of the original data with a simulated data set see Appendix B.4). The ancestries are sampled with different sampling probabilities of European and Asian ancestries between the two groups to simulate an imbalance of the ancestries between groups. For more details on the choice of parameters, see Appendix B.3.

Cell counts are sampled from a multinomial distribution with probabilities depending on the age (L_{1i}), ancestry (L_{2i}) and group (A_i) of the corresponding individual. For each individual the library size N_i is sampled from a Poisson distribution with parameter λ equal to the mean library size of the original data. The mean of the confounding effect of age is smaller because this value needs to be multiplied by the age (L_{1i}). To make sure the effect of age does not explode, this value is chosen to be much smaller.
Note that the signal cell types are now defined as those that have β_{3p} different from zero.

3.5.3 Assessment of methodologies in simulation study

In both the nonparametric and the parametric simulation study, the aforementioned methodologies are evaluated on the simulated data. To assess their performance, 250 iterations are used to simulate the data as mentioned before. In each simulation, all methods are performed using functions created in `R` (see Appendix E), both accounting for confounders and not accounting for them. For each method both the raw p-value and the adjusted p-values are stored.
Confidence intervals are computed on confidence levels of 90%, 95% and 99% (except for Wilcoxon). For the parametric simulation also the coverage of these confidence intervals is estimated, to evaluate how often the confidence interval includes the true parameter.

The performance of each method is assessed using different criteria. For each iteration (i.e. simulated data set) and for different values of the significance level α, the true positive proportion (TPP) and false discovery proportion (FDP) are calculated, defined as

$$\text{TPP}_i = \frac{\#\text{ True positives}}{\#\text{ Truly differential cell types}} \tag{33}$$

and

$$\text{FDP}_i = \frac{\#\text{ False positives}}{\#\text{ Positives}} \tag{34}$$

respectively. Positives are defined as cell types whose null hypothesis is rejected. True positives are therefore the rejections from cell types that are truly differential between lupus patients and healthy controls. False discoveries on the other hand are hypotheses that are rejected for cell types that are in fact not differential between lupus patients and healthy controls.
After 250 iterations, the sensitivity and false discovery rate (FDR) are estimated by taking the average of the TPP and FDP, respectively:

$$\text{Sensitivity} = \frac{1}{250}\sum_{i=1}^{250} TPP_i \tag{35}$$

and

$$\text{FDR} = \frac{1}{250}\sum_{i=1}^{250} FDP_i. \tag{36}$$

ROC curves are generated using the raw p-values and the functionality of `iCOBRA` [20].

Another interesting feature to evaluate the performance is looking at the top k cell types, sorted by significance, and see if this matches the truth (i.e. the cell types for which there is truly a disease effect).

4 Software

In Table 3 a list of the most important functions in `R` [18] (and the used version) is shown for each method discussed in this section.

Table 3: Most important `R` functions used. The version of `R` used is `R 4.3.1`.

Method	Function	Package	Version
Wilcoxon	`wilcox.test`	`stats`	4.3.1
Linear regression	`lm`	`stats`	4.3.1
voomCLR	`voomCLR`	`voomCLR`	0.99.24
linDA	`linda`	`MicrobiomeStat`	1.2
IPW	`ipwpoint`	`ipw`	1.2.1
	`svyglm`	`ipw`	1.2.1
Standardization	`glm`	`stats`	4.3.1
	`stdGlm`	`stdReg`	3.4.1

5 Results

5.1 Data exploration

Before setting up a simulation study and conducting any analysis, it is useful to start with some data exploration. There are 145 samples from healthy individuals and 203 from lupus patients. They originate from 98 and 158 unique individuals respectively. As already mentioned in the data description (chapter 2) only one sample for each individual is included for further analysis. That is why for the data exploration, only those samples will be used in the exploration.

Age is considered to be an important confounder in this case study. On the one hand, age has been shown to influence the blood cell type composition [21]. On the other hand, the lupus patients in the data are generally older. The observed ages in the data range from 20 to 83 years. As shown in Figure 3, the ages are not uniformly distributed. The average age in this case study is 41 years. However, here it becomes already clear that the lupus patients are on average older (44 years) than the healthy controls (37 years).

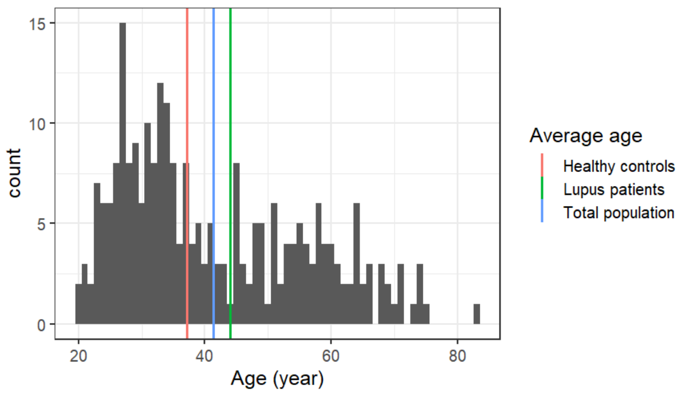
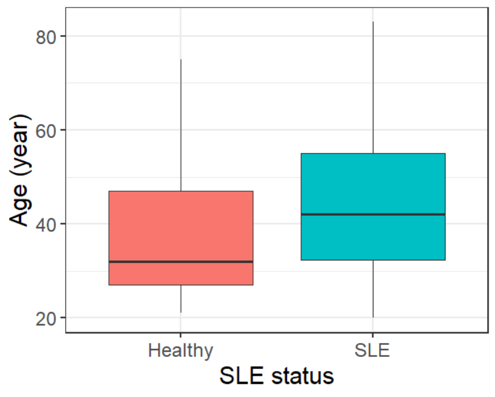
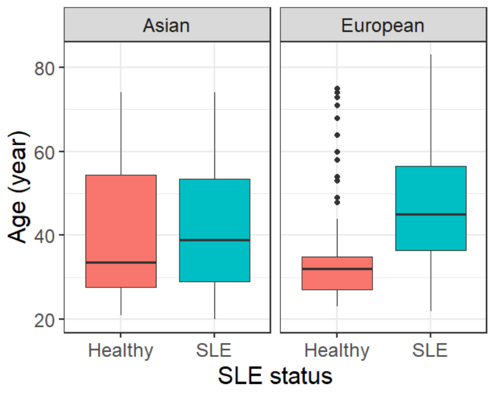


Figure 3: The distribution of the age of the individuals. The overall mean age is 41 years (indicated in red). The mean age from the healthy controls is 37 years (indicated in green). The mean age from the lupus patients is 44 years (indicated in blue).

To further investigate the association between age and disease, the boxplots in Figure 4 show the distribution of age for each disease status. Since for the analysis only one observation per individual is used, the age is represented for each individual rather than for each sample.



(a)



(b)

Figure 4: Age per disease status in all individuals *(left)* and per ancestry *(right)*. Lupus patients are typically older.

It is striking that the lupus patients in the data are typically older (Figure 4a). If you make a distinction between the ancestries (Figure 4b), this difference seems more obvious in the European patients. However, we do have to note that from the 107 Asian individuals only 24 are healthy. On the other hand, the European individuals are balanced as there are 74 healthy individuals and 75 lupus patients. An overview of these numbers are presented in Table 4. By this observation, one can consider ancestry as a confounder as well; if you observe a healthy individual it is more likely from European ancestry. The original paper compares frequencies of cell types for each ancestry separately.

Table 4: Distribution of disease status per ancestry.

	Asian	European	Total
SLE	83 (77.57%)	75 (50.34%)	158
Healthy	24 (22.43%)	74 (49.66%)	98
Total	107 (100%)	149 (100%)	256

Investigating the role of age and ancestry on the cell composition is more complicated, because of the compositional characteristic of the data. If age or ancestry has influence on the absolute abundance of one cell type, this can also be reflected in the other cell types.

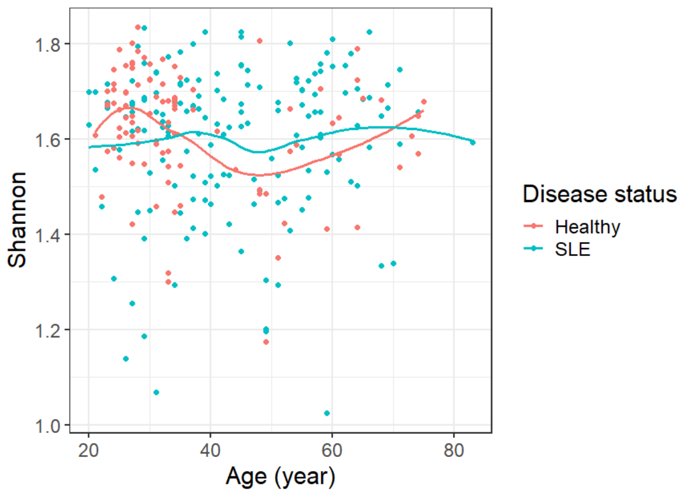


Figure 5: Shannon index for each individual in function of age.

Figure 5 shows the Shannon index for each individual's sample in function of age. The Shannon index is a way to measure the diversity of cell types in a sample. The higher the value of the Shannon index, the higher the diversity of cell types in a particular sample. The lower this index, the lower the diversity [4]. There does not seem to be a trend in this diversity over age. However, as from literature it is known that age does influence the immune cell composition, this will be considered as a confounder [21].

Now it is time to investigate what really is of interest; the cell type composition across different disease statuses.

5.1.1 Relative abundance

As we only observe relative abundances with scRNA-seq data, it is a logical choice to look at relative abundances of cell types. The relative abundance is defined as the observed count from a cell type divided by the total count observed in the corresponding sample.

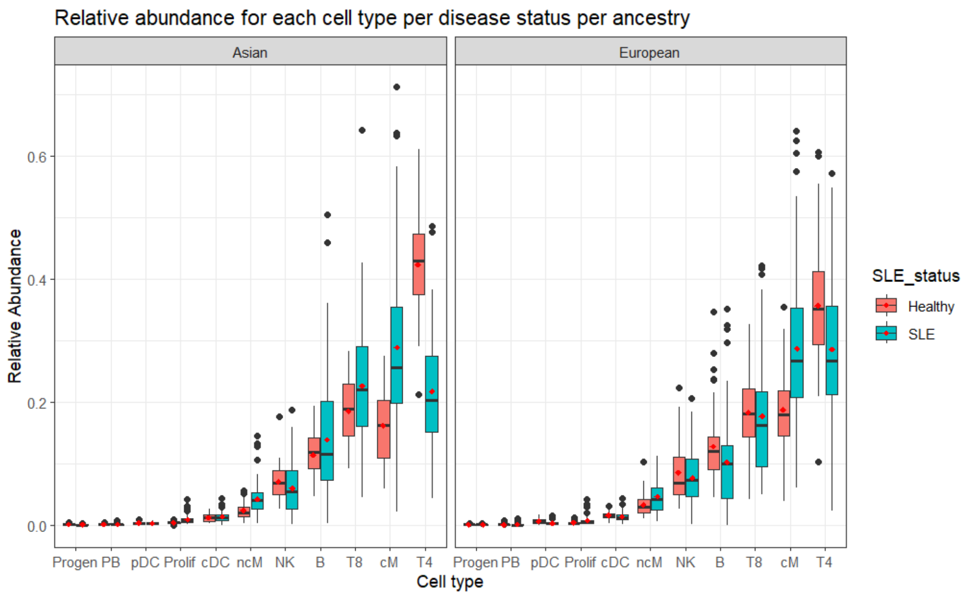


Figure 6: Boxplots for each cell type for different disease status per ancestry. The red dots indicate the mean for each cell type.

In Figure 6 the relative abundance for each cell type is shown, comparing the SLE observations with the healthy controls for each ancestry separately. The means for each cell type are indicated by a red dot. There seems to be a difference in relative abundance for cM and T4 in both ancestries. The difference in relative abundance of T4 between SLE patients and healthy controls seems to be larger in observations from Asian ancestry than from European ancestry. Also in the ncM cells there seems to be a difference in relative abundance. The B cell type appears to have a wider range in Asian SLE patients than in healthy Asian observations. The NK cells don't seem to differ in relative abundance between SLE observations and healthy controls.

The relative abundances of Progen, PB, pDC, Prolif and cDC are very small, regardless of the disease status. However, as this figure shows the relative abundance, it is not clear from this scale whether or not there is a difference in this relative abundance for these rare cell types. Figure 20 in Appendix C.2 shows only these cell types with the relative abundance on a smaller scale.
In the Asian ancestry, there seems to be only a difference in relative abundance in Prolif cells. In European ancestry however, there do seems to be a difference as well in pDC and cDC cells.
This gives an indication that there might also be differences in cell composition between ancestries. Figure 7 can be used for an exploratory comparison between immune cell compositions of different ancestries. The average relative abundance for each cell type are shown for each ancestry-SLE status combination.

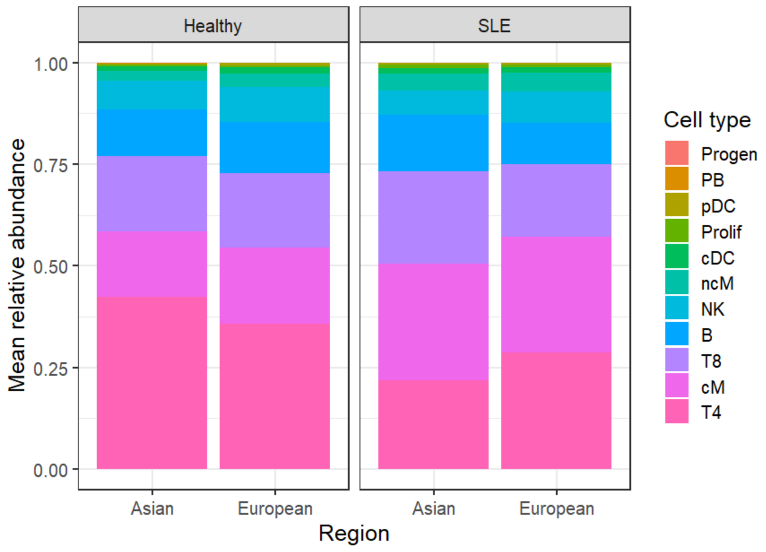


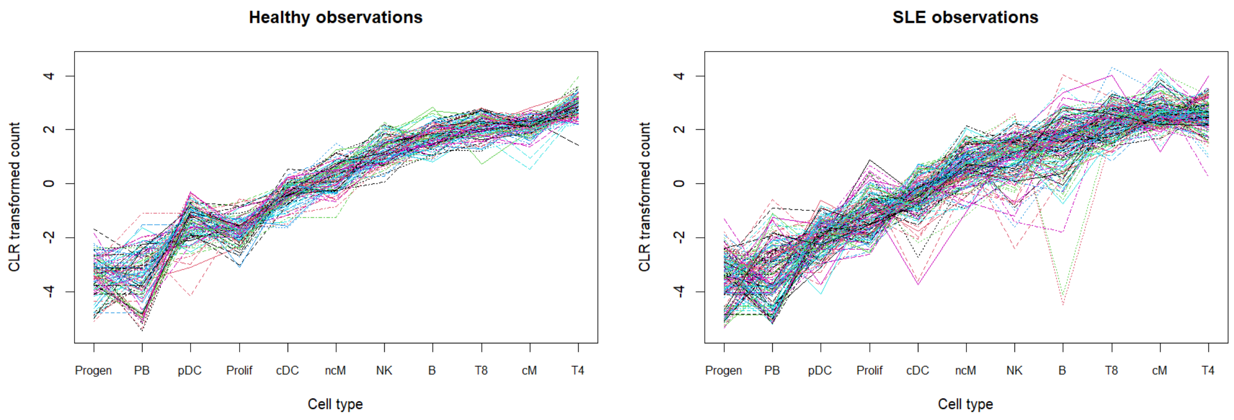
Figure 7: Mean relative abundance per ancestry and disease status.

If we compare the healthy Asians with the healthy Europeans, the largest difference seems to be in the relative abundance of T4 and cM. However, also in the B, NK, ncM and cDC cells there seems to be a minor difference between the ancestries. Although it is not visible because of the small relative abundance, the average relative abundance from the PB cells is twice that of the European ancestry. For the pDC cell type it is the other way around. The Prolif cell type seems to have different average relative abundance as well between lupus patients and healthy controls for both ancestries.
In the SLE patients, the difference in the relative abundance of B is somewhat larger between the ancestries. The largest difference seems again to be for T4, but also for T8 there is a larger difference in SLE patients than for healthy patients. The average relative abundance of cM seems not to be different between the two ancestries in SLE patients.

In summary, we observe differences in relative abundance of the cM celltype between healthy and diseased patients for both ancestries. For the B cell type we observe differences as well between healthy and diseased patients, although it might be subtle. We also observe a difference between the diseased patients of different ancestry. In the T8 cell type we observe a difference in relative abundance between lupus patients from different ancestry and between healthy controls and lupus patients from Asian ancestry. The most remarkable cell type however seems to be T4. Both between the ancestries as between lupus patients and healthy controls there seems to be a difference in relative abundance.

5.1.2 CLR-transformed counts

Because the relative abundance can be misleading in terms of interpretation due to compositionality, it might be better to investigate the CLR-transformed counts. Figure 8 shows the CLR-transformed count per cell type for each individual. In Figure 8a the healthy control samples are shown and in Figure 8b the lupus samples are shown.



(a) Healthy observations (b) SLE observations

Figure 8: Individuals' CLR-transformed counts for each cell type for healthy controls *(left)* and for SLE patients *(right)*.

It seems that among the lupus samples there is more variability in the CLR count for each cell type than for healthy observations. In the Prolif cell type for instance we notice that the CLR counts in the healthy observations are never above zero, while in the SLE observations there are samples that have a CLR count above zero. This means that for the healthy controls, the Prolif cell type is observed less than the geometric mean in each sample, while in SLE observations this is not always the case.

Figure 9 shows another representation of the CLR counts per cell type, but now separate for each ancestry. The red dots indicate the mean CLR count for each disease status in the corresponding cell type.

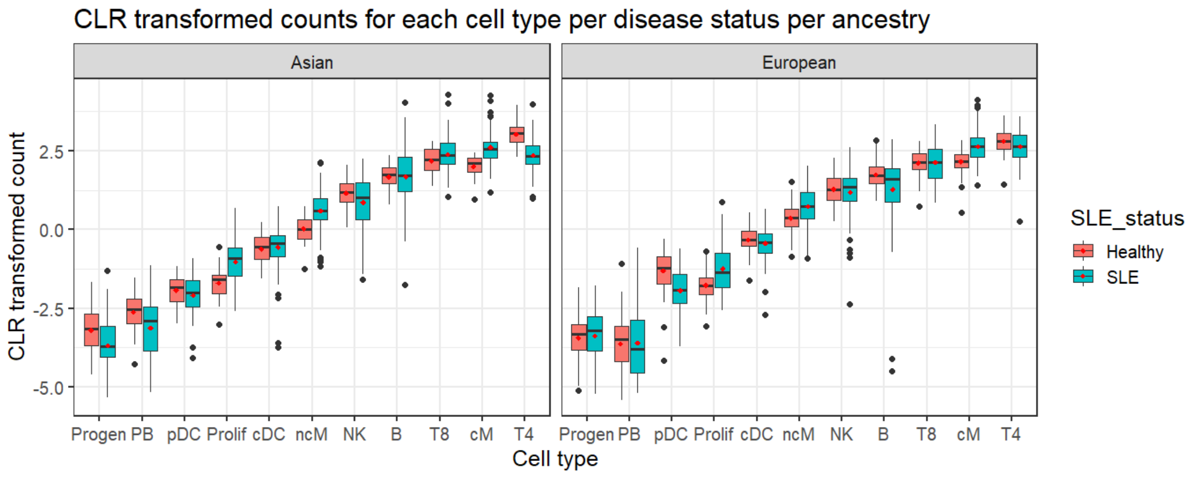


Figure 9: Boxplots for CLR-transformed counts from each cell type for different disease status per ancestry. The red dots indicate the mean for each cell type.

According to this figure it seems that the CLR-transformed counts from the Progen cell type are different between lupus patients and healthy controls, at least for Asian ancestry. Also the T4 cells seem to only show differences in the Asian samples. In the pDC cell type on the other hand, the difference seems to be more obvious in the European ancestry. In both ancestries there seems to be a difference in the CLR-transformed counts of Prolif, ncM and cM cells. The B cell type seems to show differences in the range of the CLR-transformed counts. Although the mean in the Asian ancestry seems to be the same, there is a difference in the mean CLR-transformed count in European samples.

5.1.3 PCA on compositional data

PCA is a very common way to visualize data. However, in the context of compositional data the CLR-transformed counts should be used in order to make Euclidean distances meaningful.
Figure 10 shows a scree plot, indicating the explained variance by each principal component. The first two principal components explain 42% of the variance. Ideally, we would show the first four, maybe even five principal components, but unfortunately this is not possible.

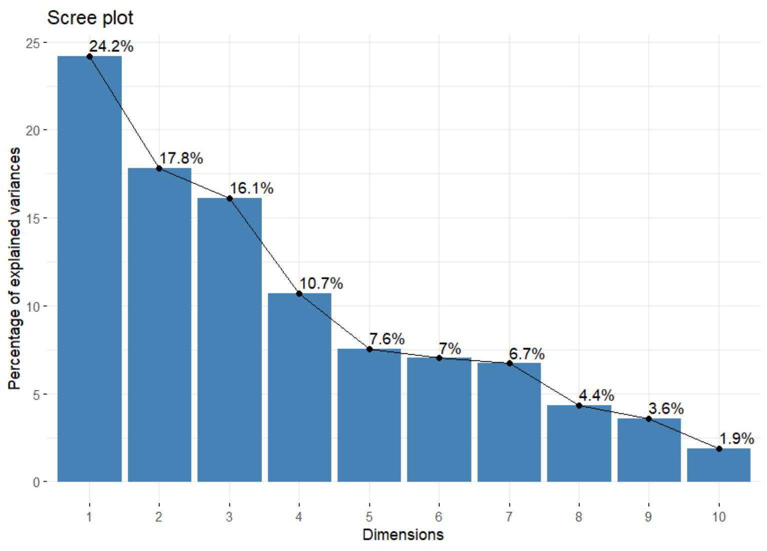


Figure 10: Scree plot for PCoA. The first two principal components only explain 42% of the variance.

Figure 11 shows the scores from the first two principal components. There seems to be more or less a separation in the second dimension between the healthy controls and the lupus patients. The samples from different ancestries are more scattered through the plot.

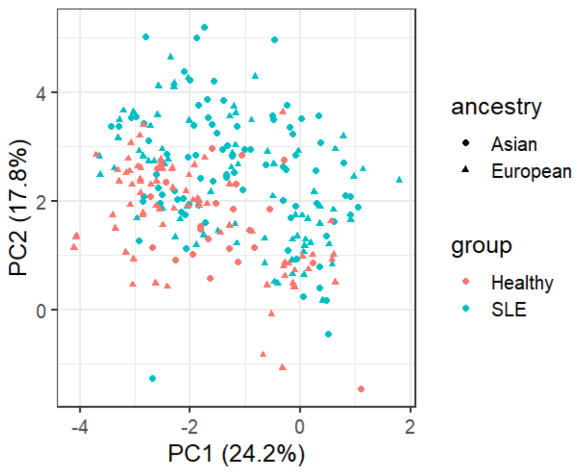
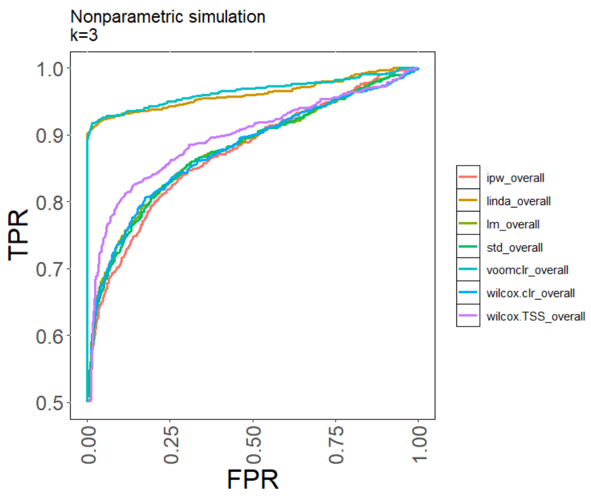


Figure 11: Scores from first two principal components.

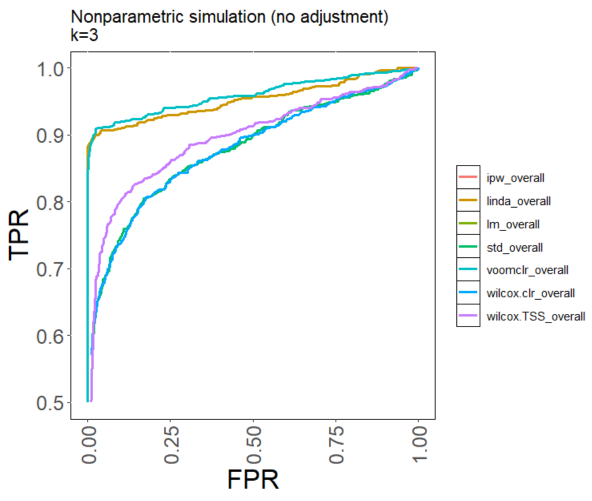
5.2 Simulation study

5.2.1 Nonparametric simulation

Figure 12 shows the overall performance of the different methodologies in terms of false positive rate (FPR) (or 1-specificity) and true positive rate (TPR) (or sensitivity) [24]. Figure 12a shows the performance of the procedures when they account for confounding of age and ancestry. In Figure 12b the methods did not take into account confounding. Although it might not be clear, the curves from linear regression (lm), inverse probability weighting (ipw) and standardization (std) overlap in the latter.



(a) Accounting for confounding



(b) Not accounting for confounding

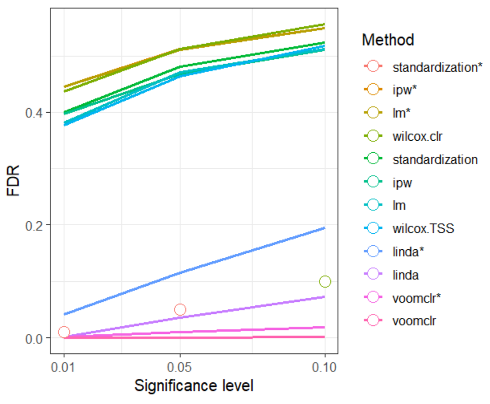
Figure 12: ROC curves after 250 iterations of nonparametric simulation with 3 differential cell types. The small difference between linear regression (lm), inverse probability weighting (ipw) and standardization (std) is not there anymore when no confounders are included; their curves overlap.

Table 5 displays the proportion of iterations where the three cell types with the smallest p-values were the actual signal cell types. In each iteration the methods were evaluated both accounting for confounding (a) and not accounting for confounding (b).

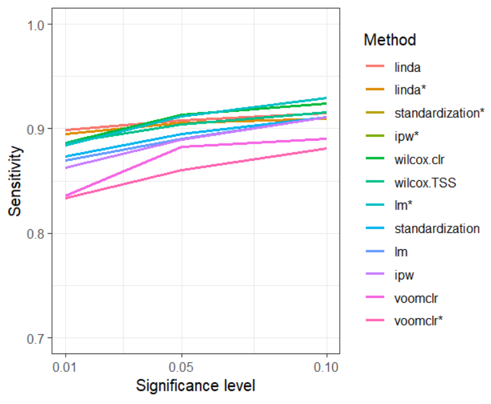
Table 5: The proportion of simulations where the top 3 cell types according to the method matched the truth. Row a shows the performance when the methods account for confounding and row b shows the performance when the methods do not account for confounding.

	Wilcoxon (TSS)	Wilcoxon (CLR)	voomCLR	linDA	Linear regression	IPW	Standardization
a	0.516	0.524	0.788	0.812	0.548	0.516	0.540
b	0.516	0.524	0.744	0.752	0.540	0.540	0.540

To check whether the methods control the FDR and have good sensitivity, Figure 13a and Figure 13b show the estimated FDR and sensitivity over 250 iterations, respectively.



(a) FDR



(b) Sensitivity

Figure 13: Estimated FDR and sensitivity over 250 iterations for each method. The circles indicate the threshold for controlling the FDR on significance level $\alpha = 0.01, 0.05$ and 0.1. ** Indicate the setting without accounting for confounding.*

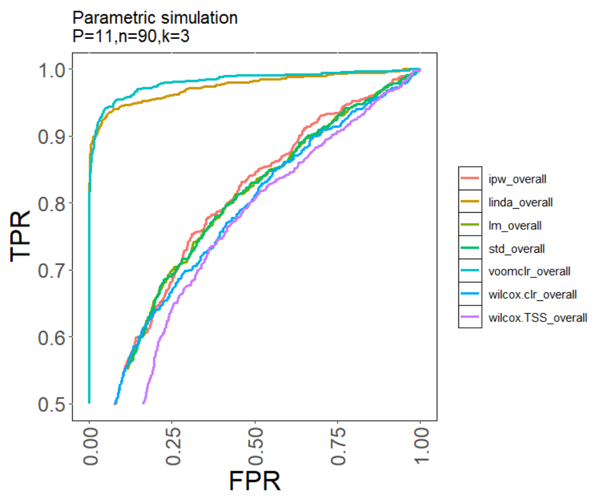
Both LinDA and voomCLR estimate the bias of the effect sizes in the model for the CLR-transformed counts. The mean of the bias over the 250 iterations is shown in Table 6 for both methods. For the distribution of the bias values, see Appendix C.3.

Table 6: Mean (sd) of the bias of the effect sizes based on the CLR transformed counts according to linDA and voomCLR.

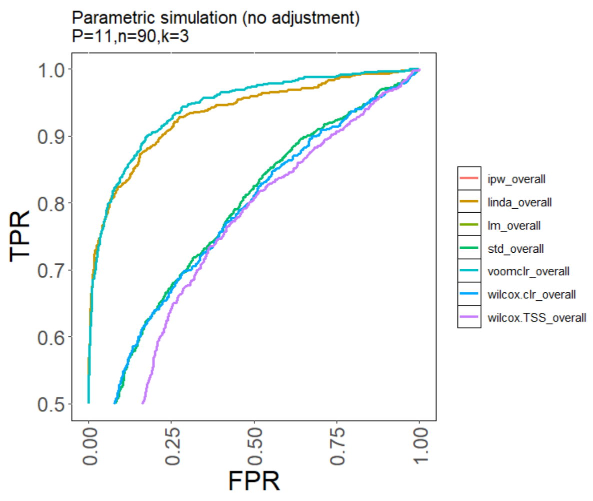
	linDA	voomCLR
a	-0.00854 (0.544)	-0.00421 (0.379)
b	-0.0121 (0.548)	-0.00636 (0.379)

5.2.2 Parametric simulation

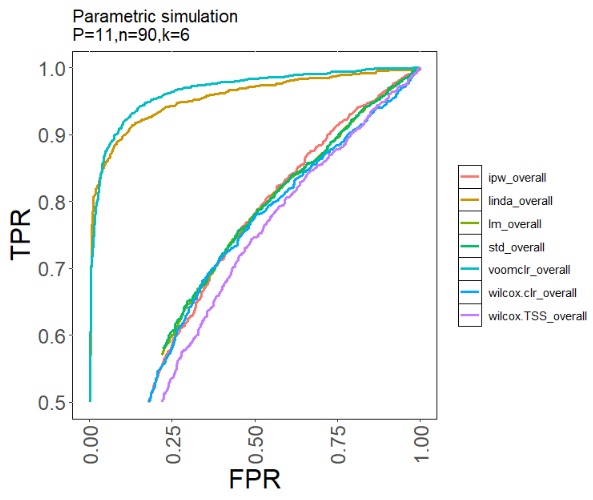
Figure 14 and 15 show the overall performance of the different methodologies in terms of false positive rate (FPR) and true positive rate (TPR) in different settings for the parametric simulation. These settings are described in Table 2. The left panels show the results when the procedures account for confounding of age and ancestry. In the right panels, the procedures did not account for confounding. Again, in the latter the curves for linear regression, inverse probability weighting and standardization overlap.



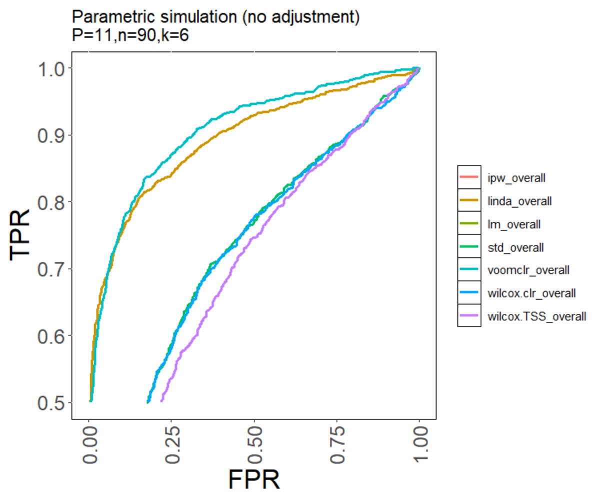
(a) A: Accounting for confounding



(b) A: Not accounting for confounding

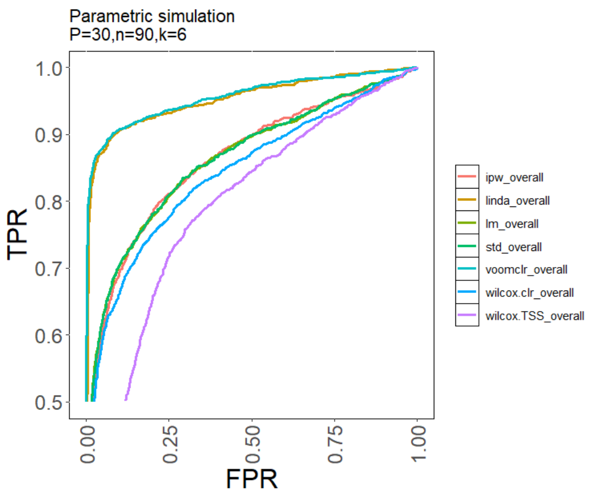


(c) B: Accounting for confounding

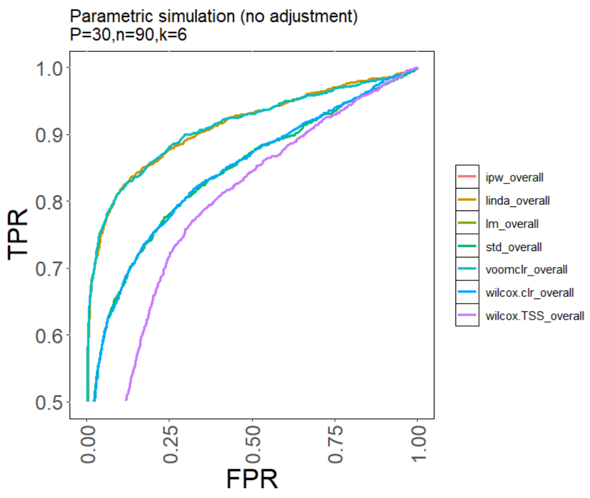


(d) B: Not accounting for confounding

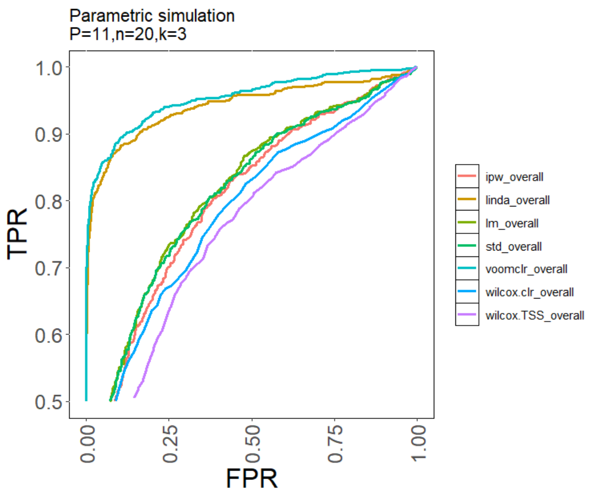
Figure 14: ROC curves after 250 simulations in parametric setting A *(top)* and B *(bottom)*. Without accounting for confounders, the curves for linear regression (lm), inverse probability weighting (ipw) and standardization (std) overlap.



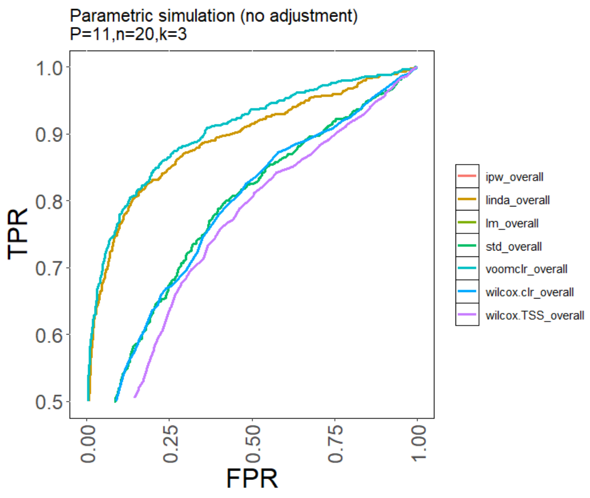
(a) C: Accounting for confounding



(b) C: Not accounting for confounding



(c) D: Accounting for confounding



(d) D: Not accounting for confounding

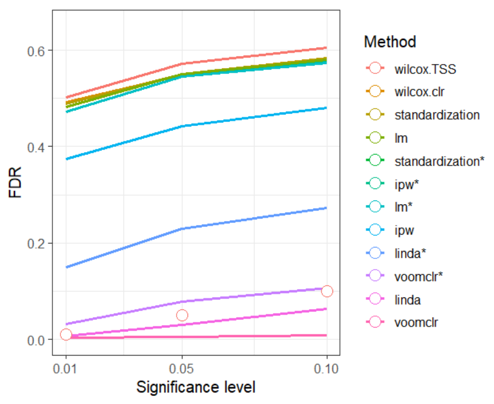
Figure 15: ROC curves after 250 simulations in parametric setting C *(top)* and D *(bottom)*. Without accounting for confounders, the curves for linear regression (lm), inverse probability weighting (ipw) and standardization (std) overlap.

In each situation, voomCLR and linDA outperform the other methods.
Table 7 shows the proportion of iterations where the top k according to the smallest p-value were the actual signal cell types. In each iteration and for each setting, the methods were evaluated both accounting for confounding and not accounting for confounding (indicated by '*').

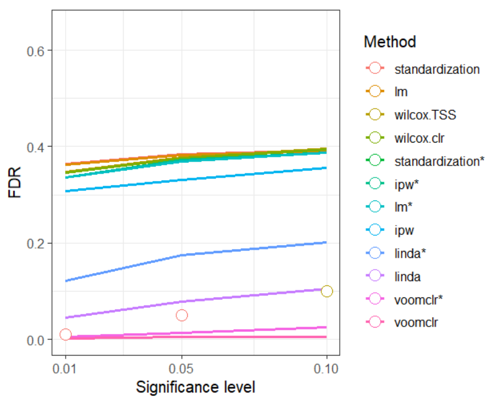
Table 7: The proportion of simulations where the top k cell types according to the method matched the truth.

	Wilcoxon (TSS)	Wilcoxon (CLR)	voomCLR	linDA	Linear regression	IPW	Standardization
A	0.212	0.288	0.772	0.816	0.380	0.340	0.376
A*	0.212	0.288	0.540	0.576	0.280	0.280	0.280
B	0.096	0.136	0.512	0.620	0.212	0.188	0.212
B*	0.096	0.136	0.328	0.360	0.156	0.156	0.156
C	0.088	0.124	0.424	0.404	0.196	0.148	0.192
C*	0.088	0.124	0.188	0.196	0.132	0.132	0.132
D	0.168	0.232	0.636	0.628	0.368	0.288	0.368
D*	0.168	0.232	0.416	0.432	0.256	0.260	0.260

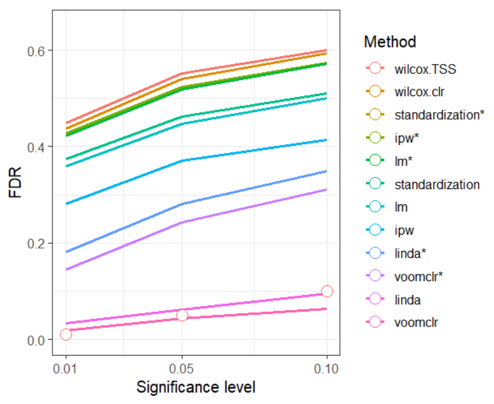
To investigate which methods control the FDR, Figure 16 shows the estimated FDR for each method in each setting over 250 iterations.



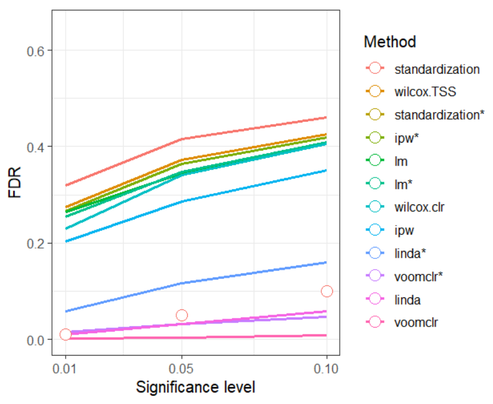
(a) Setting A



(b) Setting B



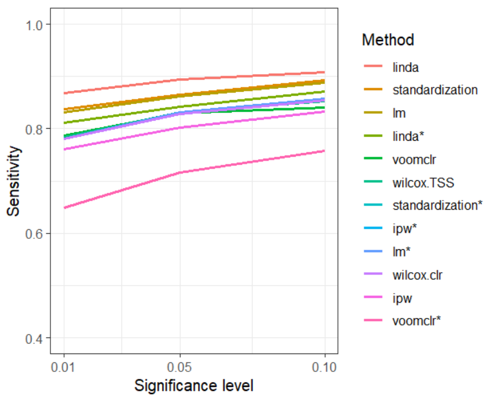
(c) Setting C



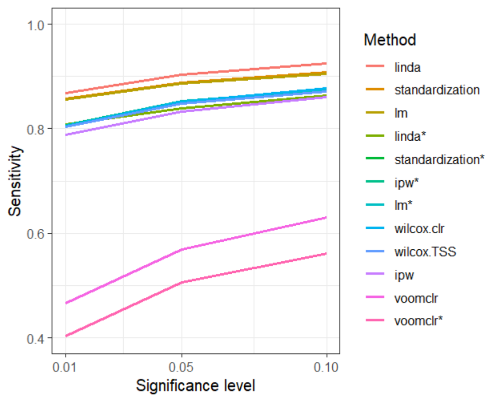
(d) Setting D

Figure 16: Estimated FDR over 250 iterations for each method. The circles indicate the threshold for controlling the FDR on significance level $\alpha = 0.01, 0.05$ and 0.1. ** Indicate the setting without accounting for confounding.*

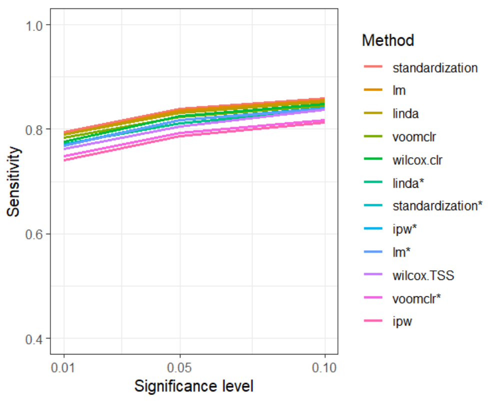
Figure 17 shows the estimated sensitivity over 250 iterations for each method for different values of the significance level.



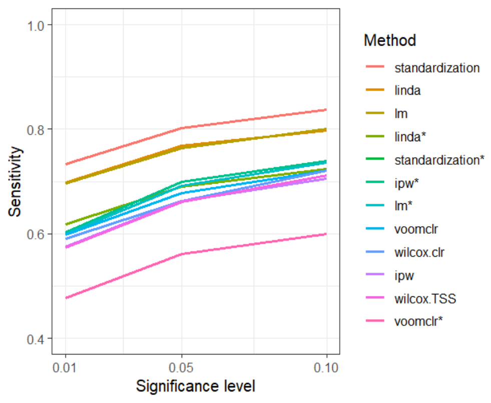
(a) Setting A



(b) Setting B



(c) Setting C



(d) Setting D

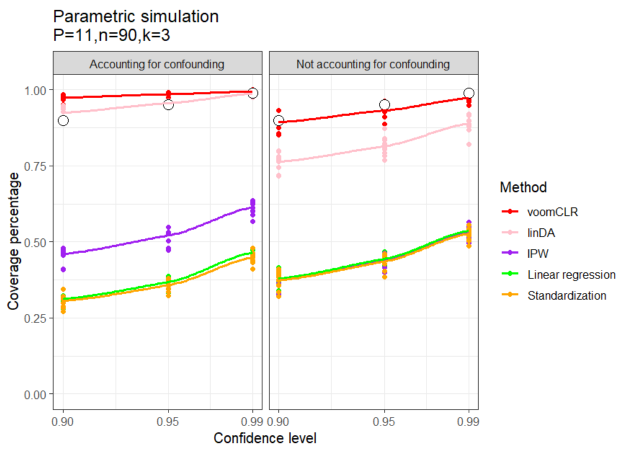
Figure 17: Estimated sensitivity over 250 iterations for each method on significance levels $\alpha = 0.01, 0.05$ and 0.1. ** Indicate the setting without accounting for confounding.*

For each setting, the mean (and standard deviation) of the bias estimates from linDA and voomCLR are shown in Table 8. For the distribution of these estimates, see Appendix C.4.

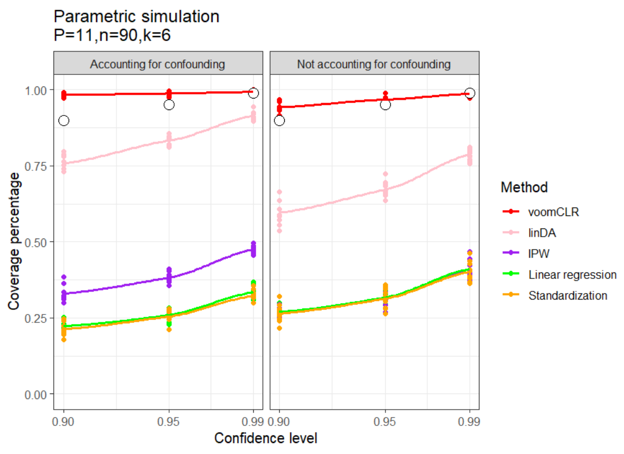
Table 8: Mean (sd) of the bias of the effect sizes based on the CLR transformed counts according to linDA and voomCLR. ** Indicate the setting without accounting for confounding.*

	linDA	voomCLR
A	-0.00579 (0.215)	-0.00427 (0.149)
A*	-0.00683 (0.219)	-0.00502 (0.152)
B	-0.0137 (0.332)	-0.00982 (0.232)
B*	-0.0163 (0.338)	-0.0118 (0.234)
C	-0.0179 (0.164)	-0.0119 (0.112)
C*	-0.0204 (0.168)	-0.0144 (0.117)
D	-0.0258 (0.269)	-0.0187 (0.185)
D*	-0.0239 (0.274)	-0.0185 (0.189)

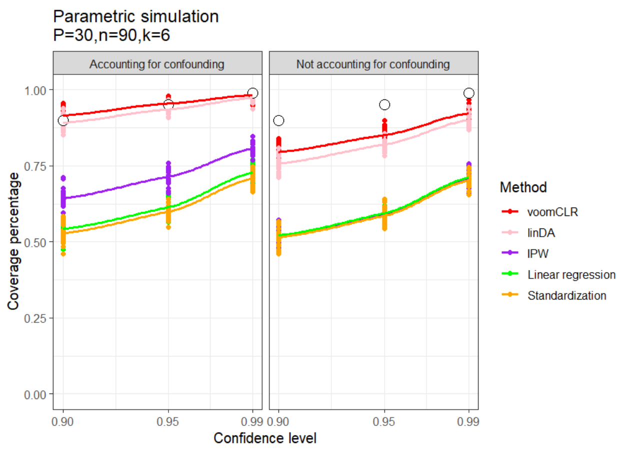
In the parametric simulation, the true value of the disease effect is known. Figure 18 shows the proportion of simulations for which the effect size was included in the corresponding confidence interval for each cell type. The circles indicate the proportion one expects to see for the given confidence level.



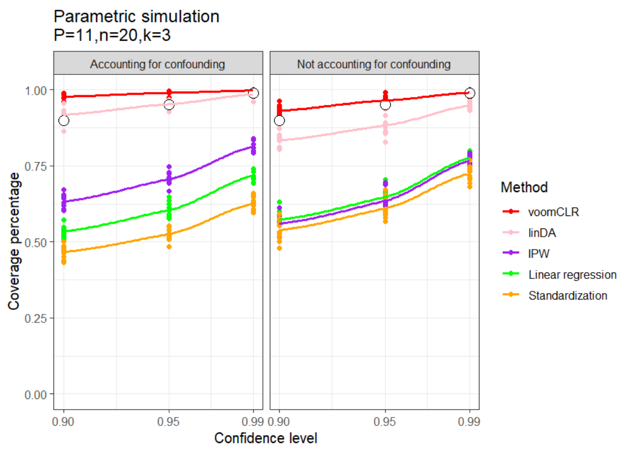
(a) Setting A



(b) Setting B



(c) Setting C



(d) Setting D

Figure 18: Coverage percentage of confidence intervals with confidence levels 90%, 95% and 99%. The points represent the percentage of coverage for a certain cell type over iterations for a given method. The circles indicate the expected level of coverage for the given confidence level.

Note that linDA calculates fold change estimates on the $\log_2$ scale, while the true effect is generated on the log fold change scale. The confidence intervals for the coefficients from linDA were therefore transformed to the log scale.

5.3 Implementation case study

It seems that linDA performs slightly better than voomCLR in terms of sensitivity, while still controlling the FDR. The case study is therefore analyzed using linDA. The results are shown in Table 9. However, because the performance of these methods is comparable, the results according to voomCLR can be found in Appendix C.5.

Table 9: Results analysis case study data with linDA. The estimates are shown in both $\log_2$ scale, which is given by linDA automatically, and the log scale. The cell types in bold are the ones that are found to be significant on the significance level α of 5%.

	log2FoldChange	SE	logFoldChange	pvalue	padj
cM	1.072	0.090	0.743	2.861e-26	**3.147e-25**
Prolif	1.203	0.125	0.834	5.347e-19	**2.941e-18**
ncM	0.874	0.113	0.606	2.724e-13	**9.989e-13**
T8	0.510	0.098	0.353	4.211e-7	**1.158e-6**
pDC	-0.262	0.121	-0.181	3.100e-2	6.800e-2
T4	-0.185	0.097	-0.128	5.700e-2	1.040e-1
cDC	0.194	0.117	0.134	9.900e-2	1.560e-1
B	-0.112	0.176	-0.077	5.270e-1	6.440e-1
Progen	0.112	0.161	0.077	4.880e-1	6.440e-1
PB	0.085	0.192	0.059	6.560e-1	7.220e-1
NK	-0.031	0.135	-0.022	8.180e-1	8.180e-1

6 Discussion

6.1 Results

The results show that voomCLR and linDA outperform the other methods in terms of true positive rate while controlling the false positive rate. This is visible from the ROC curves from both the non-parametric and parametric simulations. Both methods seem to perform equally well, at least in terms of the FDR. They both seem to control the FDR where the other methods did not control the FDR at all. However, in the setting where the number of differential cell types is about half of the total number of cell types (setting B), only voomCLR seems to control the FDR, even without accounting for confounders. In setting C, where the number of cell types is larger, both methods have a higher FDR compared to other settings. In this setting however, all methods have comparable sensitivity.

In the nonparametric simulations, the difference in performance between when the methods account for confounding and when they don't, seems not to be that large. In the parametric simulations on the other hand there seems to be a larger difference in performance. Wilcoxon does not take into account confounding, so this methods' performance remains the same. The other methods however perform better when they take the confounders into account. In the parametric simulation, Wilcoxon with TSS normalized counts seemed to be the least performing method, while this is not the case in the nonparametric simulation. A possible explanation could be that the confounding in the original data, that is used for the nonparametric simulations, does not have that much effect, compared to the signal of the disease that is introduced. In the parametric simulation on the other hand, we can control the level of confounding, and this effect seems to be stronger than in the nonparametric setting. Since the TSS transformed counts suffer from the issue of compositionality and Wilcoxon does not take into account other covariates, this method might fail to distinguish these effects from each other. Using the CLR counts shows slight improvements, but this method still performs poorly in comparison with linDA and voomCLR.

While accounting for confounding, there is a small difference in the performance of linear regression, IPW and standardization, which is not there anymore when the methods do not take into account the confounders. In fact, the only difference between these methods when no confounders are taken into account, is the standard error. Both IPW and standardization use a robust standard error, while linear regression does not. Their parameter estimates are exactly the same. Therefore, the difference in performance is minimal. These methods do seem to perform better than Wilcoxon, although the difference in performance vanishes when these methods don't take into account confounders.

If we would compare the methods in terms of the ability to identify the top k differential cell types, in every setting voomCLR and linDA perform much better than all other methods, even without accounting for confounding. In all settings without accounting for confounding, the linear regression, ipw and standardization seem to perform equally poor as the Wilcoxon using CLR transformed counts.

In terms of coverage of confidence intervals, voomCLR performs the best over all simulation settings, even without accounting for confounding. When voomCLR accounts for confounders, the coverage of the confidence intervals corresponds to the expected coverage for the given confidence levels. It even exceeds the desired coverage percentage. This might be an indication that this method is somewhat too conservative with too wide confidence intervals. For linDA, the estimated coverage percentage matches the desired confidence levels, except for setting B, where the number of cell types that are truly differential is about half of the total cell types. This can be explained because of the assumption that is made in both linDA and voomCLR. They both use the assumption that the majority of the cell types is not differential to estimate a bias correction on the effect size. That assumption is violated in this case.

In summary, it seems that both linDA and voomCLR are suitable to identify cell types with a significant disease effect. In all simulation settings, voomCLR controls the FDR on various levels of significance. In most settings linDA also shows control of the FDR, although this seems more of an issue when the number of truly differential cell types gets larger. In terms of sensitivity, linDA shows better performance. If interest lies in identifying as much cell types as possible, linDA is preferred. If it is more important that not too many cell types are identified, voomCLR might be preferred. The choice of the best method therefore depends on the research question.

The data from the case study is analyzed with linDA. The results show that these 4 cell types are significantly differential between lupus patients and healthy controls on the 5% level of significance; cM, Prolif, ncM and T8. The first two cell types were also identified as differential in the original paper [16]. For comparison, the results according to voomCLR are shown in Appendix C.5. According to voomCLR, cM and prolif are the only significant cell types.

6.2 Possible drawbacks

It is possible that the signal that is introduced in the nonparametric simulation to emulate a disease effect is not realistic and too strong compared to the confounding effect. This might lead to unrealistic signals where the minimum count in one group exceeds the maximum in the other (or vice versa). One could think of more realistic ways to introduce a signal (e.g. add or subtract a constant proportional to the relative abundance of a cell type).

During parametric simulation, the parameter settings were chosen arbitrarily. Perhaps other values could simulate a more realistic setting. However, an attempt was done to make sure that during the simulation there was not one cell type that was dominating all the others in terms of the multinomial probabilities. However, it is possible that the effect sizes might still not be realistic, so reviewing the literature on realistic effect sizes or asking advice from experts in the field might be useful.

It is important to think about the consequences if one of the identifiability conditions is violated before making causal interpretations. As mentioned before, in the original data it is possible that the consistency assumption is violated, since the disease exists in multiple forms. There is no guarantee that the different states and types result in the same potential outcome, i.e. cell composition. This assumption is important in both inverse probability weighting and standardization to be able to interpret an associational effect as a causal effect. The same holds for (conditional) exchangeability. This assumption is satisfied in the nonparametric simulation study, since we only worked with the healthy population. These observations should be exchangeable. Another condition that is important is the positivity, which ensures unbiased estimates for the causal effect. This assumption was not violated either in the simulation studies and in the original data, except for random violations due to limited sample size.

Another issue is that voomCLR might be too conservative. It has been shown to be a good method in terms of FDR, but the confidence intervals can be wide because of the additional uncertainty that is taken into account. This method however looks promising.

6.3 Further research

There are still topics that need further investigation. For instance, now we only investigate main effects. It might also be interesting to explore interaction effects and assess heterogeneity in the disease effect. From the data exploration it already seemed there are differences in disease effects between ancestries.

Additionally, we could leverage more information based on the replicates that were now left out of the analysis, if we could figure out how they are obtained. Another option is to investigate the different disease groups and investigate differences among lupus patients.

The causal inference methods seemed not to perform that well using the CLR-transformed counts as outcome. To combine the causal inference field with the compositional data analysis, it might be insightful to come up with solutions to combine both frameworks in order to come up with causal effect estimators that can handle compositionality. Perhaps we could introduce the bias correction in some way in the inverse probability weighting procedure. For future research purposes, it is advised to investigate further the effects of violations of these conditions by implementing these violations in the simulation framework.

Regarding the CLR-transformed counts, it might also be useful to investigate other approaches to deal with zero counts, as these do occur in cell composition data. Including a pseudo-count is rather arbitrary and perhaps a more robust approach can do a better job.

7 Conclusion

It has been shown that it is important to take into account compositionality. Methods like voomCLR and linDA that make adjustments to deal with the issues related to compositionality clearly showed better performance in terms of identifying significant results. Although the overall performance of both methods is very similar, linDA seems to be less conservative than voomCLR. However, it still remains a challenge to interpret the obtained coefficients as causal effects, as these methods include confounders in their model formulation. Interpretation is therefore not marginal but conditional. A suggestion for further research is therefore to look into the causal inference framework in combination with compositional data analysis.

References

[1] Takele Assefa Alemu, Verbist Bie, and Van den Berge Koen. *voomCLR*. Tech. rep. Statistics and Decision Sciences, Johnson and Johnson Innovative Medicine, 2024.

[2] Dhammika Amaratunga and Javier Cabrera. *Exploration and Analysis of DNA Microarray and Protein Array Data*. Wiley & Sons, Inc, 2003. DOI: 10.1002/9780470317129.

[3] Laura Tabellini Pierre (Parse Biosciences). *The Why and How of scRNA-Seq: A Guide for Beginners*. 2024. URL: https://www.parsebiosciences.com/blog/the-why-and-how-of-scrna-seq-a-guide-for-beginners/#:~:text=ScRNA-Seq%20technology%20allows%20researchers,to%20identify%20rare%20cell%20populations.

[4] Zach Bobbitt. *Shannon Diversity Index: Definition & Example*. Mar. 2021. URL: https://www.statology.org/shannon-diversity-index/.

[5] Law C.W, Chen Y., and Shi W. "voom: precision weights unlock linear model analysis tools for RNA-seq read counts". In: *Genome Biol* 15.R29 (2014). URL: https://doi.org/10.1186/gb-2014-15-2-r29.

[6] Changde Cheng et al. "A Review of Single-Cell RNA-Seq Annotation, Integration, and Cell–Cell Communication". In: *Cells* 12.15 (2023). ISSN: 2073-4409. DOI: 10.3390/cells12151970. URL: https://www.mdpi.com/2073-4409/12/15/1970.

[7] S. Choudhary and R. Satija. "Comparison and evaluation of statistical error models for scRNA-seq." In: *Genome Biology* 23.27 (2022). URL: https://doi.org/10.1186/s13059-021-02584-9.

[8] Lucia Clarotto, Denis Allard, and Alessandra Menafoglio. "A new class of -transformations for the spatial analysis of Compositional Data". In: *Spatial Statistics* 47 (2022), p. 100570. ISSN: 2211-6753. DOI: https://doi.org/10.1016/j.spasta.2021.100570. URL: https://www.sciencedirect.com/science/article/pii/S2211675321000725.

[9] JL Connolly, SJ Schnitt, HH Wang, et al. "Tumor Structure and Tumor Stroma Generation". In: *Holland-Frei Cancer Medicine*. Ed. by DW Kufe, RE Pollock, RR Weichselbaum, et al. 6th. Hamilton, ON: BC Decker, 2003. URL: https://www.ncbi.nlm.nih.gov/books/NBK13447/.

[10] W. Hosmer David, Lemeshow Stanley, and May Susanne. "Applied Survival Analysis: Regression Modeling of Time-to-Event Data, Second Edition". In: *John Wiley & Sons* (2008).

[11] Matthew L Davis, Yuan Huang, and Kai Wang. "Rank normalization empowers a t-test for microbiome differential abundance analysis while controlling for false discoveries". In: *Briefings in Bioinformatics* 22.5 (Apr. 2021), bbab059. ISSN: 1477-4054. DOI: 10.1093/bib/bbab059. URL: https://doi.org/10.1093/bib/bbab059.

[12] Zhou H. et al. "LinDA: linear models for differential abundance analysis of microbiome compositional data". In: *Genome Biology* 23.95 (2022). URL: https://doi.org/10.1186/s13059-022-02655-5.

[13] Miguel A. Hernán and James M. Robins. *Causal Inference: What If*. Chapman & Hall/CRC, 2020.

[14] Ostner Johannes, Büttner Maren, and Schubert Benjamin. *Compositional data analysis in scRNA-seq*. Accessed on 31/05/2024. URL: https://sccoda.readthedocs.io/en/latest/compositional_data.html#compositional-data-analysis-in-scrna-seq.

[15] Aitchison John. "Principles of Compositional Data Analysis". In: *Lecture Notes-Monograph Series* 24 (1994), pp. 73–81. ISSN: 07492170. URL: http://www.jstor.org/stable/4355794.

[16] Richard K. Perez et al. "Single-cell RNA-seq reveals cell type–specific molecular and genetic associations to lupus". In: *Science* 376.6589 (2022), eabf1970. DOI: 10.1126/science.abf1970. URL: https://www.science.org/doi/abs/10.1126/science.abf1970.

[17] Thomas P Quinn et al. “Understanding sequencing data as compositions: an outlook and review”. In: *Bioinformatics* 34.16 (Mar. 2018), pp. 2870–2878. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bty175. URL: https://doi.org/10.1093/bioinformatics/bty175.

[18] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing. Vienna, Austria, 2023. URL: https://www.R-project.org/.

[19] A. Sjölander. “Regression standardization with the R package stdReg”. In: *Eur J Epidemiol* 31 (2016), pp. 563–574. URL: https://doi.org/10.1007/s10654-016-0157-3.

[20] Charlotte Soneson and Mark D Robinson. “iCOBRA: open, reproducible, standardized and live method benchmarking”. In: *Nature Methods* 13.4 (2016), p. 283. URL: http://www.nature.com/nmeth/journal/v13/n4/full/nmeth.3805.html.

[21] Qihua Tan et al. “Handling blood cell composition in epigenetic studies on ageing”. In: *International Journal of Epidemiology* 46.5 (June 2017), pp. 1717–1718. ISSN: 0300-5771. DOI: 10.1093/ije/dyx083. URL: https://doi.org/10.1093/ije/dyx083.

[22] Willem M. van der Wal and Ronald B. Geskus. “ipw: An R Package for Inverse Probability Weighting”. In: *Journal of Statistical Software* 43.13 (2011), pp. 1–23. DOI: 10.18637/jss.v043.i13. URL: https://www.jstatsoft.org/index.php/jss/article/view/v043i13.

[23] *What is a lupus flare?* Accessed on June 4, 2024. URL: https://www.lupus.org/resources/what-is-a-flare.

[24] Wikipedia contributors. *Receiver operating characteristic — Wikipedia, The Free Encyclopedia.* [Online; accessed 16-June-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Receiver_operating_characteristic&oldid=1229142688.

[25] X. Yu et al. “Statistical and Bioinformatics Analysis of Data from Bulk and Single-Cell RNA Sequencing Experiments”. In: *Methods in Molecular Biology (Clifton, N.J.)* 2194 (2021), pp. 143–175. DOI: 10.1007/978-1-0716-0849-4_9. URL: https://doi.org/10.1007/978-1-0716-0849-4_9.

A Data description

Table 10: Description of variables and their values. The range of values is considered without one replicate sample with ID IGTB1906_IGTB1906:dmx_count_AHCM2CDMXX_YE_0831 that is left out of the analysis due to small total cell count.

Variable name	Description	Values
patient	Unique ID for each sample (n=347)	1004_1004:dmx_YE_7-13, 1014_1014:dmx_YE_7-13,...
group	The group the patient belongs to	Flare, Healthy, Managed, Treated
batch_cov	Pool ID from sample (n=23)	dmx_YE_7-13, dmx_YS-JY-22_pool5, dmx_YS-JY-20_pool4,...
ind_cov	Unique ID for each individual (n=256)	1004_1004, 1014_1014, ..., FLARE001, FLARE004, ...
Processing_Cohort	Batch ID from sample (n=4)	1,2,3,4
L3 [1]	Binary indicator if sample belongs to the cohort of cases and controls that are age matched and equal in number of cases of Asian and European ancestry in processing batch 4 and their replicates in other batches. Processing batch 4 refers to the L3 cohort samples within processing batch 4 only.	0,1
Age	Age of the patient (in years)	[20,83]
Sex	Sex of the patient	Female, Male
pop_cov	Ancestry of the patient	European, Asian
SLE_status	Disease status of the patient	Healthy, SLE
B	Observed absolute count B cells	[0, 1801]
NK	Observed absolute count NK cells	[0, 1474]
Progen	Observed absolute count Progen cells	[0, 13]
Prolif	Observed absolute count Prolif cells	[0, 176]
T4	Observed absolute count T4 cells	[87, 5303]
T8	Observed absolute count T8 cells	[86, 2243]
cDC	Observed absolute count cDC cells	[0, 211]
cM	Observed absolute count cM cells	[1, 2835]
ncM	Observed absolute count ncM cells	[0, 752]
pDC	Observed absolute count pDC cells	[0, 72]
PB	Observed absolute count PB cells	[0, 38]

B Methodology

B.1 Derivation sum CLR counts

$$\begin{aligned}
\sum_{p=1}^{P} clr(Y_{ip}) &\overset{(1)}{=} \sum_{p=1}^{P} \log \frac{Y_{ip}}{exp(\frac{1}{P}\sum_{k=1}^{P}\log(Y_{ik}))} \\
&= \sum_{p=1}^{P}\left(\log(Y_{ip}) - \log\left(exp(\frac{1}{P}\sum_{k=1}^{P}\log(Y_{ik}))\right)\right) \\
&= \sum_{p=1}^{P}\left(\log(Y_{ip}) - \left(\frac{1}{P}\sum_{k=1}^{P}\log(Y_{ik})\right)\right) \\
&= \sum_{p=1}^{P}\log(Y_{ip}) - P\frac{1}{P}\sum_{k=1}^{P}\log(Y_{ik}) \\
&= 0
\end{aligned}$$

B.2 Bias effect size based on CLR (linDA)

Say the (unobserved) absolute counts from sample i are annotated as X_{ip} for cell type p and the observed 'relative' abundances are annotated as Y_{ip} for cell type p.
Assume a multinomial distribution for the cell composition from sample i: $Y_{ip} \sim Mult(N_i, \frac{X_{ip}}{\sum_{p=1}^{P} X_{ip}})$

[1]Supplementary materials Perez et al. [16].

with $N_i = \sum_{p=1}^{P} Y_{ip}$. This implies that $E(Y_{ip}) = N_i \cdot \frac{X_{ip}}{\sum_{p=1}^{P} X_{ip}}$ or $E\left(\frac{Y_{ip}}{\sum_{p=1}^{P} Y_{ip}}\right) = \frac{X_{ip}}{\sum_{p=1}^{P} X_{ip}}$.
Under this assumption, we can write

$$\log\left(\frac{Y_{ip}}{\sum_{p=1}^{P} Y_{ip}}\right) = \log\left(\frac{X_{ip}}{\sum_{p=1}^{P} X_{ip}}\right) + e_{ip} \tag{37}$$

The interest is the effect of disease on the absolute abundance of cell types. In case we would observe absolute abundances, we would fit the following log-linear model:

$$\log(X_{ip}) = \alpha_p u_i + \beta_{0p} + \beta_{1p} X_{1i} + \beta_{2p} X_{2i} + \varepsilon_{ip} \tag{38}$$

with

- $u_i = \text{SLE status} = \begin{cases} 1, & \text{if sample } i \text{ belongs to lupus patient} \\ 0, & \text{else} \end{cases}$
- X_{1i} = Age from sample i
- $X_{2i} = \text{Ancestry} = \begin{cases} 1, & \text{if sample } i \text{ belongs to patient of European ancestry} \\ 0, & \text{if sample } i \text{ belongs to patient of Asian ancestry} \end{cases}$
- ε_{ip} the error term, assumed to be normally distributed with constant variance.

We are only interested in the parameter α_p, the effect of the disease on the absolute abundance of cell type p. In fact, we are testing the null hypotheses $H_{0,p} : \alpha_p = 0$ versus the alternative $H_{1,p} : \alpha_p \neq 0$, which corresponds to the hypotheses in (4).
The linear model for the CLR-transformed counts satisfies the following linear model:

$$\begin{aligned} clr(Y_{ip}) := \log\left(\frac{Y_{ip}}{(\prod_{p=1}^{P} Y_{ip})^{1/P}}\right) &= \log\left(\frac{Y_{ip}}{\sum_{k=1}^{P} Y_{ik}}\right) - \frac{1}{P}\sum_{j=1}^{P} \log\left(\frac{Y_{ij}}{\sum_{k=1}^{P} Y_{ik}}\right) \\ &= \log(X_{ip}) + e_{ip} - \frac{1}{P}\sum_{p=1}^{P} log(X_{ip}) - \frac{1}{P}\sum_{p=1}^{P} e_{ip} \\ &= u_i(\alpha_p - \bar{\alpha}) + (\beta_{0p} - \bar{\beta}_0) + (\beta_{1p} - \bar{\beta}_1)X_{1i} + (\beta_{2p} - \bar{\beta}_2)X_{2i} + \tilde{\varepsilon}_{ip} - \bar{\varepsilon}_p \end{aligned}$$

where $\bar{\alpha} = \frac{1}{P}\sum_{p=1}^{P} \alpha_p$, $\bar{\beta}_j = \frac{1}{P}\sum_{p=1}^{P} \beta_{jp}$ (j=0,1,2) and $\bar{\varepsilon}_p = \frac{1}{P}\sum_{p=1}^{P} \tilde{\varepsilon}_{ip}$ and $\tilde{\varepsilon}_{ip} = e_i + \varepsilon_{ip}$.

The estimator for α_p based on the CLR-transformed data is biased with the bias term being $\bar{\alpha}$.
In many applications, it is reasonable to assume that there is only a small portion of differential cell types (most $\alpha_p = 0$). Denote $\tilde{\alpha}_p$ as an unbiased estimate for $\alpha_p - \bar{\alpha}$. The mode of $\tilde{\alpha}_p$ is expected to be close to $-\bar{\alpha}$.
One estimates α_p by the bias-corrected estimator $\hat{\alpha}_p = \tilde{\alpha}_p + \tilde{\alpha}$, with $-\tilde{\alpha}$ the estimate for the mode of $\tilde{\alpha}_p$. In fact, we make sure that we shift the obtained estimate such that the mode becomes zero [12].

B.3 Parameters parametric simulation

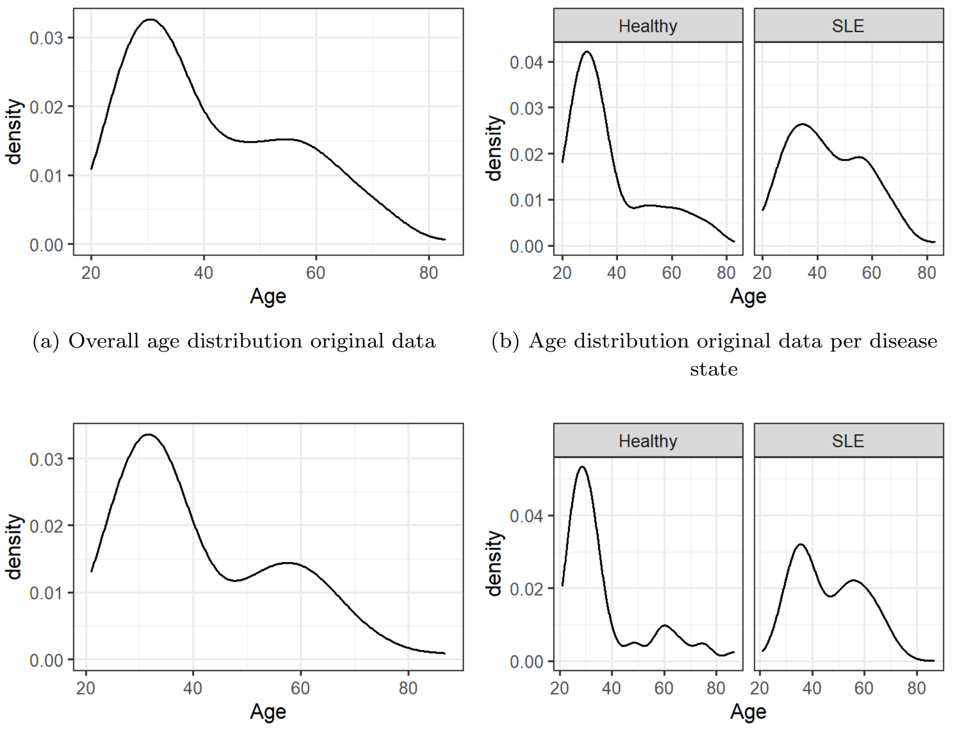
Table 11 shows the parameters used to simulate a dataset with an age and ancestry for each individual.

Table 11: Parameter settings used to simulate age and ancestry distribution per group. Age is sampled using a mixture of gamma distributions. Ancestry is sampled using a binomial distribution.

Age	Group 1	Group 2
Probability weights	0.7, 0.3	0.55, 0.45
α	30, 65	35, 55
β	1,1	1,1
Ancestry	Group 1	Group 2
Asian	0.25	0.55
European	0.75	0.45

B.4 Parametric simulation age distribution

In Figure 19 you can see that the simulated distribution looks similar to the original data.[2]



(a) Overall age distribution original data

(b) Age distribution original data per disease state

(c) Overall age distribution simulated data

(d) Age distribution simulated data per disease state

Figure 19: Age distribution in original data *(top)* versus in simulated data *(bottom)*.

[2]`set.seed(1234)` was used to generate this figure

C Results

C.1 Data exploration batches

Table 12: Distribution of SLE patients and healthy controls in each batch. Batch 1 only contains healthy individuals. The distribution between healthy and SLE is more or less evenly distributed in batches 3 and 4.

Batch	Healthy	SLE
1	48	0
2	36	124
3	17	27
4	44	52

C.2 Data exploration: rare cell types

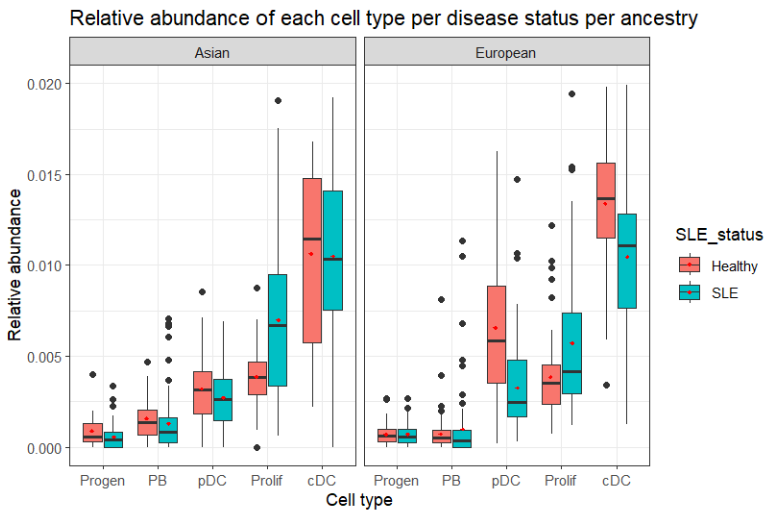
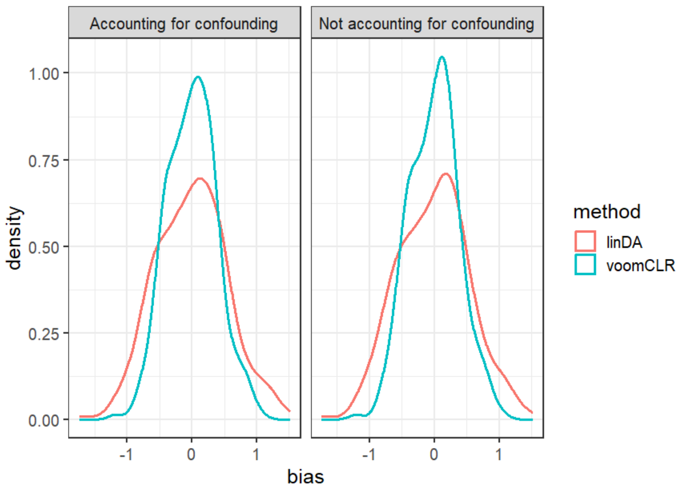


Figure 20: Relative abundance for rare cell types compared between lupus patients and healthy controls for different ancestries.

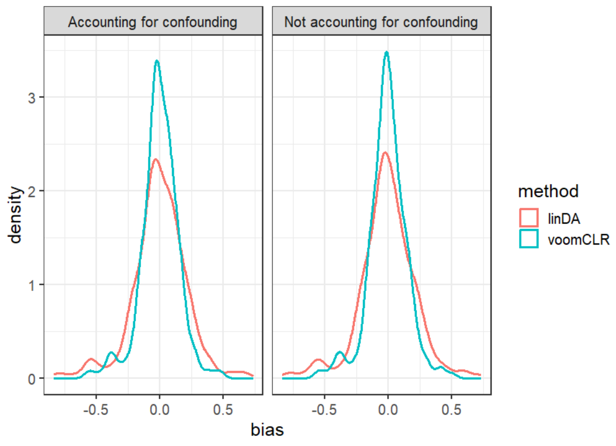
C.3 Nonparametric simulation bias



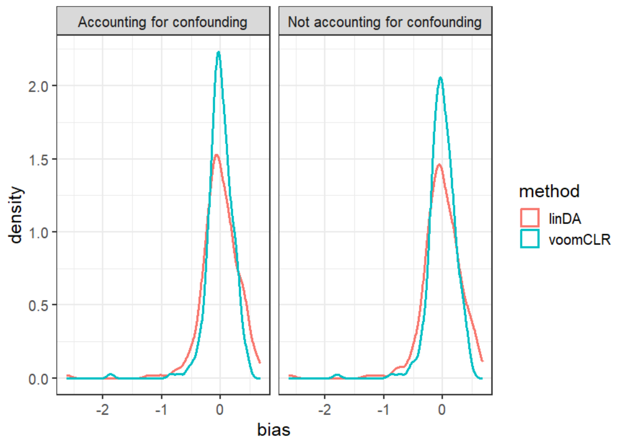
(a)

Figure 21: Distribution of bias terms from voomCLR and linDA in the simulations when accounting for confounders and without accounting for them.

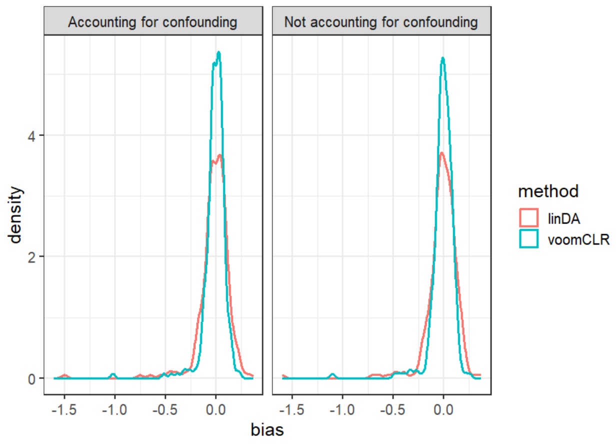
C.4 Parametric simulation bias



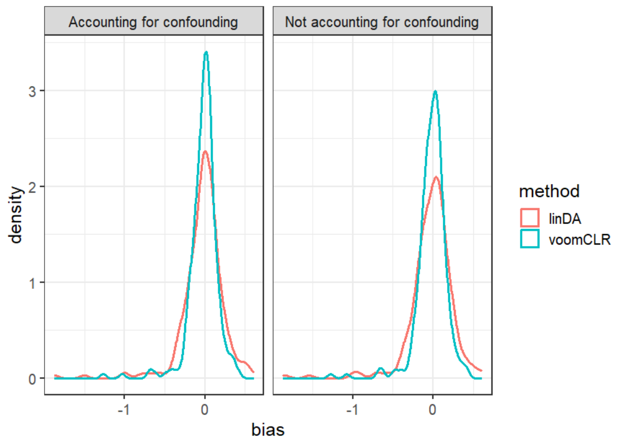
(a) Setting A: n=90, P=11, k=3



(b) Setting B: n=90, P=11, k=6



(c) Setting C: n=90, P=30, k=6



(d) Setting D: n=20, P=11, k=3

Figure 22: Distribution of bias terms from voomCLR and linDA in the simulations when accounting for confounders and without accounting for them.

C.5 Implementation case study (voomCLR)

Table 13: Results voomCLR implementation on case study. The cell types in bold are the cell types that are significant on the 5% significance level.

	logFC	t	P.Value	adj.P.Val
Prolif	0.851	3.343	9.384e-4	**1.032e-2**
cM	0.753	3.059	2.435e-3	**1.340e-2**
ncM	0.606	2.416	1.630e-2	5.978e-2
T8	0.362	1.459	1.457e-1	4.007e-1
pDC	-0.193	-0.765	4.447e-1	9.293e-1
cDC	0.145	0.579	5.630e-1	9.293e-1
T4	-0.128	-0.520	6.038e-1	9.293e-1
B	-0.068	-0.253	8.002e-1	9.293e-1
Progen	0.067	0.253	8.006e-1	9.293e-1
PB	0.054	0.196	8.449e-1	9.293e-1
NK	-0.019	-0.074	9.414e-1	9.414e-1

D R code data exploration

```r
############################## Data ##########################
data <- readRDS("230705_popCountsWide_individualBatchID.rds")
data$Age <- as.numeric(paste(data$Age))
celltypes <- colnames(data[,c(11:21)])

# Eurazia
eurazia <- data %>% filter(pop_cov %in% c("Asian","European"))
eurazia$pop_cov <- eurazia$pop_cov[drop=T]

# Replicates Eurazians
eurazia$totalcounts <- rowSums(eurazia[,c(11:21)])
patients <- eurazia %>% group_by(ind_cov) %>% filter(totalcounts==max(totalcounts))
patients <- patients$patient

eurazians.duplicates.rm <- as.data.frame(eurazia %>% filter(patient %in% patients))
healthy.eurazians.duplicates.rm <- as.data.frame(eurazians.duplicates.rm %>% filter(SLE_status=="Healthy"))
healthy.eurazians.duplicates.rm$SLE_status <- healthy.eurazians.duplicates.rm$SLE_status[drop=T]

eurazians.duplicates.rm.clr <- eurazians.duplicates.rm
eurazians.duplicates.rm.ra <- eurazians.duplicates.rm
# CLR
geomMean <- exp(rowMeans(log(eurazians.duplicates.rm.clr[,c(11:21)]+0.5)))
CLR <- log((eurazians.duplicates.rm.clr[,c(11:21)]+0.5)/geomMean)
eurazians.duplicates.rm.clr[,c(11:21)] <- CLR
# RA
eurazians.duplicates.rm.ra[,c(11:21)] <- t(microbiome::transform(t(eurazians.duplicates.rm[,c(11:21)]),
                                        "compositional"))

############# Exploration ##############
n.samples <- nrow(eurazia)
n.patients <- length(unique(eurazia$ind_cov))

table(eurazia$Processing_Cohort, eurazia$SLE_status)

# Be aware that there are replicates of some patients!
n.SLE <- sum(eurazia$SLE_status=="SLE")
n.healthy <- sum(eurazia$SLE_status!="SLE")

SLE <- eurazia %>% filter(SLE_status=="SLE")
n.SLE.ind <- length(unique(SLE$ind_cov))
healthy <- eurazia %>% filter(SLE_status!="SLE")
n.healthy.ind <- length(unique(healthy$ind_cov))

replicates.eurazia <- table(eurazia$ind_cov)[table(eurazia$ind_cov)>1]

sum(replicates.eurazia==2)
sum(replicates.eurazia==3)
sum(replicates.eurazia==4)
replicated.individuals <- names(replicates.eurazia)

n.diff.batches <- rep(0, length(replicated.individuals))
for (i in 1:length(replicated.individuals)){
  batches <- eurazia$Processing_Cohort[eurazia$ind_cov == replicated.individuals[i]]
  n.diff.batches[i] <- length(unique(batches))
}
```

```r

#################### Exploration of age ###################################
eurazia$Age <- as.numeric(paste(eurazia$Age))
range(eurazia$Age)

df <- eurazia %>% group_by(ind_cov) %>% summarise(Age = unique(Age), SLE_status = unique(SLE_status),
Sex=unique(Sex), Ancestry=unique(pop_cov))
age.table <- table(df$ind_cov,df$Age)
which(rowSums(age.table)>1)
df$Age[df$ind_cov %in% names(which(rowSums(age.table)>1))]

# 1130_1130
eurazia$Processing_Cohort[eurazia$ind_cov%in% names(which(rowSums(age.table)>1))[1]]
eurazia$Age[eurazia$ind_cov%in% names(which(rowSums(age.table)>1))[1]]
# measured in batches 2 (age=27),3 (age=29),3 (age=29)

# 1772_1772
eurazia$Processing_Cohort[eurazia$ind_cov%in% names(which(rowSums(age.table)>1))[2]]
eurazia$Age[eurazia$ind_cov%in% names(which(rowSums(age.table)>1))[2]]
# Measured in batches 3 (age=21) and 4 (age=20)


mean_age <- mean(eurazians.duplicates.rm$Age)
mean_sle <- mean(eurazians.duplicates.rm$Age[eurazians.duplicates.rm$SLE_status=="SLE"])
mean_healthy <- mean(eurazians.duplicates.rm$Age[eurazians.duplicates.rm$SLE_status!="SLE"])
central_values <- data.frame(Mean_Age = c("Total population", "Lupus patients", "Healthy controls"),
                             value=c(mean_age, mean_sle, mean_healthy))

# Histogram age
ggplot(eurazians.duplicates.rm.clr, aes(Age)) + geom_histogram(binwidth = 1) +
  geom_vline(data = central_values, aes(xintercept = value, color = Mean_Age), linewidth = 1) +
  theme_bw(base_size=15) + labs(col="Average age", x="Age (year)")

# Boxplots age
ggplot(eurazians.duplicates.rm.clr, aes(x=SLE_status,y= Age)) +
geom_boxplot(aes(fill=SLE_status), show.legend=F) +
facet_wrap(~pop_cov) + theme_bw(base_size=15) +
theme(text=element_text(size=18))+labs(x="SLE status",y="Age (year)")

ggplot(eurazians.duplicates.rm.clr, aes(x=SLE_status, y= Age)) +
geom_boxplot(aes(fill=SLE_status), show.legend=F) +
theme_bw(base_size=15) + theme(text=element_text(size=18))+
labs(x="SLE status",y="Age (year)")


# Sort cell types according to median ra
median <- colMedians(as.matrix(eurazians.duplicates.rm.ra[,c(11:21)]))
sorted.celltypes <- names(sort(median))
sorted.data <- eurazians.duplicates.rm[,-c(11:22)]
sorted.data <- cbind(sorted.data, eurazians.duplicates.rm.ra[,sorted.celltypes])
sorted.data.clr <- eurazians.duplicates.rm.clr[,-c(11:22)]
sorted.data.clr <- cbind(sorted.data.clr, eurazians.duplicates.rm.clr[,sorted.celltypes])

# Plot clr count per cell type
# Only SLE
matplot(t(sorted.data.clr[sorted.data.clr$SLE_status=="SLE",c(11:21)]), type="l", xaxt='n',
    ylab="CLR transformed count", xlab="Cell type", ylim=c(-5.5,4.5))
axis(side=1,at=1:11,labels=sorted.celltypes, cex.axis=0.8)
```

```r
title("SLE observations")

# Only for healthy patients
matplot(t(sorted.data.clr[sorted.data.clr$SLE_status!="SLE",c(11:21)]), type="l", xaxt='n',
    ylab="CLR transformed count", xlab="Cell type", ylim=c(-5.5,4.5))
axis(side=1,at=1:11,labels=sorted.celltypes, cex.axis=0.8)
title("Healthy observations")


# Boxplots clr counts
asian.european <- sorted.data.clr %>% pivot_longer(cols=all_of(sorted.celltypes))
ggplot(asian.european, aes(x = factor(name, levels = sorted.celltypes), y=value, fill=SLE_status)) +
geom_boxplot() +
labs(x="Cell type", y="CLR transformed count",
    title="CLR transformed counts for each cell type per disease status per ancestry") +
 theme(text=element_text(size=10)) +
 facet_wrap(~pop_cov) +
 stat_summary(fun = mean, geom = "point", position = position_dodge(width = 0.8),
    size = 1.5, color = "red",shape = 18) +
 theme_bw(base_size=15)

# Boxplots RA
asian.european <- sorted.data %>% pivot_longer(cols=all_of(sorted.celltypes))
ggplot(asian.european, aes(x = factor(name, levels = sorted.celltypes), y=value, fill=SLE_status)) +
geom_boxplot() +
labs(x="Cell type", y="Relative Abundance",
    title="Relative abundance for each cell type per disease status per ancestry") +
theme(text=element_text(size=10)) +
facet_wrap(~pop_cov) +
stat_summary(fun = mean, geom = "point", position = position_dodge(width = 0.8),
    size = 1.5, color = "red",shape = 18) +
theme_bw(base_size=15)

# Rare celltypes
rare_celltypes <- c("Progen", "PB", "pDC", "Prolif", "cDC")
rare <- asian.european %>% filter(name %in% rare_celltypes)
ggplot(rare, aes(x=factor(name, levels=rare_celltypes), y=value, fill=SLE_status)) +
geom_boxplot() +
labs(x="Cell type", y="Relative abundance",
    title="Relative abundance of each cell type per disease status per ancestry") +
theme(text=element_text(size=10)) +
facet_wrap(~pop_cov) +
stat_summary(fun = mean, geom = "point", position = position_dodge(width = 0.8),
    size = 1, color = "red",shape = 18) + ylim(c(0,0.02)) + theme_bw(base_size=15)


# Shannon index over age
div <- microbiome::diversity(as.matrix(t(eurazians.duplicates.rm[,c(11:21)])))
div.data <- data.frame(inv.simpson = div$inverse_simpson,
                       simpson = 1/div$inverse_simpson,
                       shannon = div$shannon,
                       gini = div$gini_simpson,
                       age=eurazians.duplicates.rm$Age,
                       sle = eurazians.duplicates.rm$SLE_status)
ggplot(div.data, aes(x=age, y=shannon, col=sle)) +
geom_point() + geom_smooth(method="loess",se=F)+
theme_bw(base_size=15) +
theme(text=element_text(size=18))+
labs(y="Shannon", x="Age (year)", col="Disease status")

# Average relative abundance
summary_with_sle <- asian.european %>% group_by(pop_cov, name, SLE_status) %>% summarize(mean=mean(value))
df_SLE <- data.frame(Celltype=unique(summary_with_sle$name),
                     Asian_healthy = summary_with_sle %>% filter(pop_cov=="Asian" & SLE_status == "Healthy") %>%
                     ungroup() %>% select(mean),
                     European_healthy = summary_with_sle %>% filter(pop_cov=="European" & SLE_status == "Healthy") %>%
                     ungroup()  %>% select(mean),
                     Asian_sle = summary_with_sle %>% filter(pop_cov=="Asian" & SLE_status != "Healthy") %>%
                     ungroup()  %>% select(mean),
                     European_sle = summary_with_sle %>% filter(pop_cov=="European" & SLE_status != "Healthy") %>%
                     ungroup()  %>% select(mean))
colnames(df_SLE) <- c("Celltype", "Asian healthy",  "European healthy", "Asian sle", "European sle" )

# Barplot
summary_with_sle$celltype <- factor(summary_with_sle$name, levels=unique(asian.european$name))
ggplot(summary_with_sle, aes(x=pop_cov, y=mean, fill=celltype)) +
geom_bar(stat="identity") + facet_wrap(~SLE_status) +
labs(x="Region", y="Mean relative abundance", fill="Cell type") + theme_bw(base_size=15) +
theme(text=element_text(size=15))

ggplot(summary_with_sle, aes(x=SLE_status, y=mean, fill=celltype)) +
geom_bar(stat="identity") +
facet_wrap(~pop_cov)+
labs(x="SLE status", y="Mean relative abundance", fill="Cell type") +
theme_bw(base_size=15) +
theme(text=element_text(size=15))

# PCoA
library(compositions)
x <- acomp(eurazians.duplicates.rm[,c(11:21)])
pcx <- princomp(x)
fviz_eig(pcx, col.var="blue",addlabels=T)
scores <- data.frame(pcx$scores)
scores$group <- eurazians.duplicates.rm$SLE_status
scores$ancestry <- eurazians.duplicates.rm$pop_cov
ggplot(scores, aes(x=Comp.1, y=Comp.2, col=group, shape=ancestry)) +
geom_point() +
theme_bw(base_size=15) +
labs(x="PC1 (24.2%)",y="PC2 (17.8%)")
```

E R code simulation

```r
library(readr)
library(ggplot2)
library(tidyverse)
library(MicroBioMap)
library(microbiome)
library(phyloseq)
library(ggpubr)
library(factoextra)
library(FactoMineR)
library(limma)
library(voomCLR)
library(compositions)
library(iCOBRA)
library(DiscriMiner)
library(survey)
library(boot)
library(stdReg2)
library(ipw)
library(stdReg)
library(bmixture)

####################### Data ##########################
data <- readRDS("230705_popCountsWide_individualBatchID.rds")
data$Age <- as.numeric(paste(data$Age))
celltypes <- colnames(data[,c(11:21)])

############################ Eurazia ####################################
eurazia <- data %>% filter(pop_cov %in% c("Asian","European"))
eurazia$pop_cov <- eurazia$pop_cov[drop=T]

# Replicates Eurazians
eurazia$totalcounts <- rowSums(eurazia[,c(11:21)])
patients <- eurazia %>% group_by(ind_cov) %>% filter(totalcounts==max(totalcounts))
patients <- patients$patient
eurazians.duplicates.rm <- as.data.frame(eurazia %>% filter(patient %in% patients))
healthy.eurazians.duplicates.rm <- as.data.frame(eurazians.duplicates.rm %>% filter(SLE_status=="Healthy"))
healthy.eurazians.duplicates.rm$SLE_status <- healthy.eurazians.duplicates.rm$SLE_status[drop=T]

##################### Sampling distribution for cell type pairs  ################
# CLR transformed vector for each cell type
geomMean <- exp(rowMeans(log(healthy.europeans.duplicates.rm[,c(11:21)]+0.5)))
counts.clr <- log((healthy.eurazians.duplicates.rm[,c(11:21)]+0.5)/geomMean)

euclidean_dist <- function(x, y) sqrt(sum((x - y)^2))
pairs <- c(1:(factorial(11)/factorial(9)))
names <- rep("NA", length(pairs))
index <- 1
for (i in 1:11){
  p <- celltypes[i]
  q.candidates <- c(1:11)[-i]
  for (j in q.candidates){
    q <- celltypes[j]
    pairs[index] <- euclidean_dist(counts.clr[,i], counts.clr[,j])
    names[index] <- paste0(p, ",",q)
    index <- index + 1
  }
}

inverse <- 1/pairs
sampling.probabilities.pq <- data.frame(Pair=names, Probability=(inverse/sum(inverse)),
                                        Euclid.dist=pairs)

######################## Functions #####################
sampling.pairs <- function(k, candidates = sampling.probabilities.pq$Pair,
                           probabilities = sampling.probabilities.pq$Probability){
    # Sample pairs (p,q) using a sampling distribution

    pairs <- signal.celltype <- replacement.celltype <- rep("NA",k)

    for (i in 1:k){
      pairs[i] <- sample(candidates, 1, prob=probabilities)
      signal.celltype[i] <- as.vector(unlist(data.frame(strsplit(pairs[i],","))[1,]))
      replacement.celltype[i] <- as.vector(unlist(data.frame(strsplit(pairs[i],","))[2,]))
      # Once a signal was introduced,make sure this cell type can no longer be sampled
      # as replacement cell type (or signal cell type)
      remaining.candidates <- !grepl(paste0("\\b",signal.celltype[i],"\\b"), candidates)
      candidates <- candidates[remaining.candidates]
      probabilities <- probabilities[remaining.candidates]
    }
    return(data.frame(signal=signal.celltype, replacement=replacement.celltype))
}

comp.corr <- function(data.new, data.old, k, nr.of.celltypes=11){
    # Introduce signal in group 2 (data.old) - dimensions are rows of cell types, columns of samples

  pairs <- sampling.pairs(k)
  signal.celltype <- pairs$signal
  replacement.celltype <- pairs$replacement

  for (i in 1:k){
    data.new[rownames(data.new)==signal.celltype[i]] <-
        data.old[rownames(data.old)==replacement.celltype[i]]

    # To remain the total counts in group 2 I will now calculate the spillover counts
    d <- data.new[rownames(data.new)==signal.celltype[i]] -
        data.old[rownames(data.old)==signal.celltype[i]]

    # For each cell type and each sample we need to estimate the weight
    proportions <- matrix(0, nrow=ncol(data.old), ncol=nr.of.celltypes)
    for (j in 1:nr.of.celltypes){
      proportions[,j] <- data.old[j,]/colSums(data.old)
    }
    weights <- matrix(0, nrow=ncol(data.old), ncol=(nr.of.celltypes-1))
    # Only calculate the weights for the cell types that need compositional correction
    celltype.nr <- c(1:nr.of.celltypes)[rownames(data.new)!=signal.celltype[i]]
    for (iter in 1:(nr.of.celltypes-1)){
      weights[,iter] <- proportions[,celltype.nr[iter]]/rowSums(proportions[,celltype.nr])
    }
    # Apply compositional correction to all cell counts (except where we introduced the signal)
    c <- -weights * d
    data.new[rownames(data.new)!=signal.celltype[i]] <-
        data.new[rownames(data.new)!=signal.celltype[i]] +  t(c)
    data.old <- data.new
  }
  return (list(data.old, signal.celltype, replacement.celltype))}

################# Simulation of confounded data set ################
sample.age <- function(data=healthy.europeans, swap.pct=0.3){
  # sample groups of equal size with the same age distribution
  n <- nrow(data)
  n1 <- n2 <- floor(n/2)

  group1 <- group2 <- meta1 <- meta2 <- data.frame()
  group1age <- group2age <- c()

  quantiles <- quantile(data$Age,probs=seq(0,1,0.2))
  data$age.cat <- cut(data$Age, breaks = quantiles, include.lowest = TRUE)

  # From each category, we sample the same amount of observations in both groups
  for (cat in levels(data$age.cat)){
    healthy.cat <- data %>% filter(age.cat == cat)
    n.cat <- nrow(healthy.cat)
    n1.cat <- n2.cat <- floor(n.cat/2)
    index.cat <- sample(1:n.cat, n1.cat+n2.cat)

    group1 <- rbind(group1, healthy.cat[index.cat[1:n1.cat], c(11:21)])
    group2 <- rbind(group2, healthy.cat[index.cat[(n1.cat+1):(n1.cat+n2.cat)], c(11:21)])

    meta1 <- rbind(meta1, healthy.cat[index.cat[1:n1.cat], -c(11:21)])
    meta2 <- rbind(meta2, healthy.cat[index.cat[(n1.cat+1):(n1.cat+n2.cat)], -c(11:21)])

    group1age <- c(group1age, meta1$Age)
    group2age <- c(group2age, meta2$Age)
  }
  group1 <- t(group1)
  group2 <- t(group2)
  return (list(group1, group2, data.frame(mean.1 = mean(group1age), mean.2=mean(group2age)), meta1, meta2))
}

sample.groups <- function(data=healthy.eurazians.duplicates.rm, swap.pct=0.3){
  # This functions split the data in 2 groups of equal size
  n <- nrow(data)
  n1 <- n2 <- floor(n/2)

  group1 <- group2 <- meta1 <- meta2 <- data.frame()
  group1ancestry <- group2ancestry <- group1age <- group2age <- c()

  # For each ancestry sample groups with similar age distribution
  for (ancestry in levels(data$pop_cov)){
    subset <- data %>% filter(pop_cov == ancestry)
    groups <- sample.age(subset, swap.pct)
    group1 <- rbind(group1, t(groups[[1]]))
    group2 <- rbind(group2, t(groups[[2]]))
    stats <- groups[[3]]
    meta1 <- rbind(meta1,groups[[4]])
    meta2 <- rbind(meta2,groups[[5]])
  }

  group1 <- as.data.frame(cbind(group1, meta1))
  group2 <- as.data.frame(cbind(group2, meta2))

  # Create an imbalance in both the age and the ancestry by replacing swap.pct of the oldest in group1
  # with the youngest from group2 but keep in mind that we want to have an imbalance as well in the
  # asian distribution between groups so 2/3 of the oldest from group 1
  # will come from the europeans and the rest from the asians, to not create a systematic difference
  # in the interaction between age and ancestry
  nr.of.swap <- ceiling(swap.pct*nrow(group1))
  nr.of.swap2 <- floor(nr.of.swap/3)
  nr.of.swap1 <- 2*nr.of.swap2
  youngest.asian <- (group2 %>% filter(pop_cov=="Asian") %>% arrange(Age))[1:nr.of.swap2,]
  oldest.asian <-  (group1 %>% filter(pop_cov=="Asian") %>% arrange(-Age))[1:nr.of.swap1,]
  youngest.european <- (group2 %>% filter(pop_cov=="European") %>% arrange(Age))[1:nr.of.swap1,]
  oldest.european <-  (group1 %>% filter(pop_cov=="European") %>% arrange(-Age))[1:nr.of.swap2,]

```

```r
  group1[group1$patient %in% oldest.asian$patient,] <- youngest.european
  group1[group1$patient %in% oldest.european$patient,] <- youngest.asian
  group2[group2$patient %in% youngest.asian$patient,] <- oldest.european
  group2[group2$patient %in% youngest.european$patient,] <- oldest.asian

  meta1 <- group1[,-c(1:11)]
  meta2 <- group2[,-c(1:11)]

  group1age <- meta1$Age
  group2age <- meta2$Age
  group1ancestry <- meta1$pop_cov
  group2ancestry <- meta2$pop_cov

  return (list(t(group1[,c(1:11)]), t(group2[,c(1:11)]),
               data.frame(European1=round(mean(group1ancestry=="European")*100,2),
                          Asian1=round(mean(group1ancestry=="Asian")*100,2),
                          European2=round(mean(group2ancestry=="European")*100,2),
                          Asian2=round(mean(group2ancestry=="Asian")*100,2)),
               meta1, meta2))
  }


.getMode <- function(beta, n){
  suppressMessages(mode <- modeest::mlv(sqrt(n) * beta,
                                        method = "meanshift", kernel = "gaussian")/sqrt(n))
  return(mode)
}

voom <- function(data, variables=c("group","age","ancestry"), adjustment="BH",
                meta, meanvar = "analytical", distr="NB"){
 # Function to run voomCLR with or without accounting for variables
  group <- meta$group
  age <- meta$age
  ancestry <- meta$ancestry
  n <- ncol(data)

  formula_str <- paste(variables, collapse = " + ")
  formula <- as.formula(paste("~", formula_str))
  design <- model.matrix(formula)
  v <- voomCLR(counts = data,
               design = design,
               plot = F,
               varCalc = meanvar,
               varDistribution = distr,
               span = 0.8)
  fit <- lmFit(v, design)
  fit <- eBayes(fit)

  # The bias is estimated by the mode of the coefficients (or the shift to get the coefficients to 0)
  bias <- apply(fit$coefficients, 2, function(x){
    .getMode(x, n=n)
  })
  # We are interested in the coefficients for the group/disease status (+1 for the intercept)
  coef.group <- which(variables=="group")+1
  tt <- topTableBC(fit, coef=coef.group, n=Inf, adjust.method = adjustment, sort.by="none",
                   bootstrap="nonparametric")
  return(list(tt, bias,fit$df.residual, fit$df.prior))
}
```

```r

linDA <- function(data, variables=c("group","age","ancestry"), meta, adjustment="BH"){
    # Function to run linDA with and without accounting for variables
  meta <- meta %>% select(all_of(variables))
  formula_str <- paste(variables, collapse = " + ")
  lindaRes <- MicrobiomeStat::linda(feature.dat = data, # rows features, cols samples
                                    meta.dat = meta,
                                    formula = paste("~",formula_str),
                                    feature.dat.type = 'count',
                                    adaptive=TRUE,
                                    zero.handling = 'pseudo-count',
                                    p.adj.method=adjustment)
  return(list(lindaRes$output$group, lindaRes$bias))
}

linear.regression <- function(data, variables=c("group","age","ancestry"), adjustment="BH",
 celltypes=colnames(eurazia[,c(11:21)])){
  # Requires CLR counts; run ordinary least squares with or without correcting for additional variables
  formula_str <- paste(variables, collapse = " + ")

  coef <- se <- t <- p.val <- c()
  fits <- list()

  for (celltype in celltypes){
    formula <- as.formula(paste("data[[celltype]]~", formula_str))
    fit <- lm(formula, data=data)
    fits[[length(fits)+1]] <- fit
    s <- summary(fit)
    group.coef <- s$coefficients[grepl("group", rownames(s$coefficients))]
    coef <- c(coef, group.coef[1])
    se <- c(se, group.coef[2])
    t <- c(t, group.coef[3])
    p.val <- c(p.val, group.coef[4])
  }
  pval <- p.adjust(p.val, method=adjustment)
  names(pval) <- celltypes
  return(list(fits, data.frame(coef, se, t, pval, raw.p=p.val)))
}

inverse.probability.weighting <- function(data, variables=c("group","age","ancestry"), adjustment="BH",
                                          celltypes=colnames(eurazia[,c(11:21)]), stabilized=F){
  # Requires CLR counts and columns of the variables of interest in the data

  # Calculate the inverse probability weights (depending on the variables we want to correct for)
  if (stabilized){
    if ("age" %in% variables){
      if ("ancestry" %in% variables){
        inverse.p <- ipwpoint(exposure=group,family="binomial",link="logit",numerator=~1,
                              denominator=~age+ancestry, data=data)
      }
      else{inverse.p <- ipwpoint(exposure=group,family="binomial",link="logit",numerator=~1,
                                 denominator=~age, data=data)}
    } else if("ancestry" %in% variables){
        inverse.p <- ipwpoint(exposure=group,family="binomial",link="logit",numerator=~1,
                              denominator=~ancestry, data=data)
      }else{
        inverse.p <- ipwpoint(exposure=group,family="binomial",link="logit",numerator=~1,
                              denominator=~1, data=data)

    }} else {
      if ("age" %in% variables){
        if ("ancestry" %in% variables){
          inverse.p <- ipwpoint(exposure=group,family="binomial",link="logit",
                                denominator=~age+ancestry, data=data)
          }
        else{inverse.p <-   ipwpoint(exposure=group,family="binomial",link="logit",
                                     denominator=~age, data=data)}
        } else if("ancestry" %in% variables){
          inverse.p <- ipwpoint(exposure=group,family="binomial",link="logit",
                                denominator=~ancestry, data=data)
        }else{
            inverse.p <- ipwpoint(exposure=group,family="binomial",link="logit",
                                  denominator=~1, data=data)
  }
  }

  data$sw <- inverse.p$ipw.weights

  p.val.ipw <- coef.per.celltype <- se.per.celltype <- t.per.celltype <- c()
  fits <- list()

  for (celltype in celltypes){
    subset <- data[,c(celltype, "group","sw")]
    colnames(subset)[1] <- "celltype"

    # Marginal Structural Model
    msm <- (svyglm(celltype ~ group, design = svydesign(~1, weights = ~ sw, data = subset)))
    # Keep the fits to make confidence interval for the desired alpha level
    fits[[length(fits)+1]] <- msm
    s <- summary(msm)

    coef.per.celltype <- c(coef.per.celltype, s$coefficients[2,1])
    se.per.celltype <- c(se.per.celltype, s$coefficients[2,2])
    t.per.celltype <- c(t.per.celltype, s$coefficients[2,3])
    p.val.ipw <- c(p.val.ipw, s$coefficients[2,4])
  }
    raw.p.ipw <- p.val.ipw
    p.val.ipw <- p.adjust(p.val.ipw, method=adjustment)
    names(p.val.ipw) <- celltypes

    return(list(data.frame(coef.per.celltype, se.per.celltype, t.per.celltype, p.val.ipw, raw.p=raw.p.ipw), fits))
}

standardization <- function(data, variables, celltypes=colnames(eurazia[,c(11:21)])){
  # Requires CLR counts and columns of the variables of interest in the data

  coef <- se <- pval  <-  c()
  fits <- rds <- list()

  formula_str <- paste(variables, collapse = " + ")
  for (celltype in celltypes){
    formula <- as.formula(paste("data[[celltype]]~", formula_str))
    fit1 <-glm(formula, family="gaussian", data=data)
    fit.std <- stdGlm(fit=fit1, data=data, X="group")
    rds[[length(rds)+1]] <- rd <- summary(fit.std, contrast="difference", reference=0)
    fits[[length(fits)+1]] <- fit.std

    coef <- c(coef, rd$est.table[2,1])
    se <- c(se, rd$est.table[2,2])

    pval <- c(pval, 2*pt(abs(rd$est.table[2,1]/rd$est.table[2,2]),df=nrow(data)-2, lower.tail=F))
    deviance <- rd[["input"]][["fit"]][["deviance"]]
  }
  raw.p <- pval
  adj.p <- p.adjust(pval, method="BH")
  names(coef) <- celltypes
  return (list(data.frame(coef, se, raw.p,adj.p), fits, deviance, rds))
}

evaluation <- function(signal.celltypes, nr.of.signal, tt, wilcox.TSS.pval,
                       wilcox.clr.pval, lindaRes, ipw.pval, pval.standardization,
                       pval.lm, alpha=0.05){
  # signal.celltypes: vector of signal cell types
  # nr.of.signal: k (number of signal cell types)
  # tt: topTable result from voomCLR (unsorted)
  # wilcox.TSS.pval/wilcox.clr.pval: vector of adjusted pvalues from each cell type from wilcoxon test
  # lindaRes: output table linDA (unsorted)
  # ipw.pval: vector of adjusted pvalues inverse probability weighting
  # pval.standardization: vector of adjusted pvalues standardization
  # lm.pval: adjusted pvalues linear regression
  # alpha: nominal level

  # First check if there are NAs -> replace by 1 (not significant)
  wilcox.TSS.pval[is.na(wilcox.TSS.pval)] <- wilcox.clr.pval[is.na(wilcox.clr.pval)] <-
    ipw.pval[is.na(ipw.pval)] <- pval.lm[is.na(pval.lm)] <- 1
  ###################### voomCLR ###################################
  # Are the signal introduced cell types in the top k list of (significant) celltypes?
  sorted.tt <- tt %>% arrange(adj.P.Val)
  top.k.match.voomclr <- ifelse(sum(signal.celltypes
          %in% rownames(sorted.tt)[1:nr.of.signal])==nr.of.signal, T,F)
  nr.significant.celltypes.voomclr <- sum(tt$adj.P.Val<alpha)
  nr.TP.findings.voomclr <- sum(rownames(tt)[tt$adj.P.Val<alpha] %in% signal.celltypes)
  nr.FP.findings.voomclr <- nr.significant.celltypes.voomclr - nr.TP.findings.voomclr

  ######################## Wilcoxon ########################
  sorted.by.significance.TSS <- sort(wilcox.TSS.pval)
  top.k.match.wilcox.TSS <- ifelse(sum(signal.celltypes %in%
  names(sorted.by.significance.TSS)[1:nr.of.signal])==nr.of.signal, T,F)
  nr.significant.celltypes.wilcox.TSS <- sum(wilcox.TSS.pval<alpha)
  nr.TP.findings.wilcox.TSS <- sum(names(sorted.by.significance.TSS[sorted.by.significance.TSS<alpha]
                               %in% signal.celltypes)
  nr.FP.findings.wilcox.TSS <- nr.significant.celltypes.wilcox.TSS - nr.TP.findings.wilcox.TSS

  sorted.by.significance.clr <- sort(wilcox.clr.pval)
  top.k.match.wilcox.clr <- ifelse(sum(signal.celltypes
      %in% names(sorted.by.significance.clr)[1:nr.of.signal])==nr.of.signal, T,F)
  nr.significant.celltypes.wilcox.clr <- sum(wilcox.clr.pval<alpha)
  nr.TP.findings.wilcox.clr <- sum(names(sorted.by.significance.clr)[sorted.by.significance.clr<alpha]
                               %in% signal.celltypes)
  nr.FP.findings.wilcox.clr <- nr.significant.celltypes.wilcox.clr - nr.TP.findings.wilcox.clr

  ###################### linDA ##############################
  sorted.linda <- lindaRes %>% arrange(padj)
  top.k.match.linda <- ifelse(sum(signal.celltypes
      %in% rownames(sorted.linda)[1:nr.of.signal])==nr.of.signal, T,F)
  nr.significant.celltypes.linda <- sum(lindaRes$padj < alpha)
  nr.TP.findings.linda <- sum(rownames(lindaRes)[lindaRes$padj<alpha] %in% signal.celltypes)
  nr.FP.findings.linda <- nr.significant.celltypes.linda - nr.TP.findings.linda

  ###################### Inverse probability weighting ################################
  sorted.by.significance.ipw <- sort(ipw.pval)
  top.k.match.ipw <- ifelse(sum(signal.celltypes
      %in% names(sorted.by.significance.ipw)[1:nr.of.signal])==nr.of.signal, T,F)
  nr.significant.celltypes.ipw <- sum(ipw.pval<alpha)
  nr.TP.findings.ipw <- sum(names(ipw.pval)[ipw.pval<alpha] %in% signal.celltypes)
  nr.FP.findings.ipw <- nr.significant.celltypes.ipw - nr.TP.findings.ipw

  ###################### Standardization ################################
  sorted.by.significance.std <- sort(pval.standardization)
  top.k.match.std <- ifelse(sum(signal.celltypes
      %in% names(sorted.by.significance.std)[1:nr.of.signal])==nr.of.signal, T,F)
  nr.significant.celltypes.std <- sum(pval.standardization<alpha)
  nr.TP.findings.std <- sum(names(pval.standardization)[pval.standardization<alpha]
                      %in% signal.celltypes)
  nr.FP.findings.std <- nr.significant.celltypes.std - nr.TP.findings.std

  ################### Linear regression ###############################
  sorted.by.significance.lm <- sort(pval.lm)
  top.k.match.lm <- ifelse(sum(signal.celltypes
      %in% names(sorted.by.significance.lm)[1:nr.of.signal])==nr.of.signal, T,F)
  nr.significant.celltypes.lm <- sum(pval.lm<alpha)
  nr.TP.findings.lm <- sum(names(pval.lm)[pval.lm<alpha]
                      %in% signal.celltypes)
  nr.FP.findings.lm <- nr.significant.celltypes.lm - nr.TP.findings.lm

    return (list(data.frame(top.k.match.voomclr, top.k.match.wilcox.TSS,
    top.k.match.wilcox.clr, top.k.match.linda, top.k.match.lm, top.k.match.ipw,
    top.k.match.std), data.frame(nr.significant.celltypes.voomclr,
    nr.significant.celltypes.wilcox.TSS,nr.significant.celltypes.wilcox.clr,
    nr.significant.celltypes.linda, nr.significant.celltypes.ipw,
    nr.significant.celltypes.std, nr.significant.celltypes.lm, nr.TP.findings.voomclr,
    nr.TP.findings.wilcox.TSS, nr.TP.findings.wilcox.clr, nr.TP.findings.linda,
    nr.TP.findings.ipw, nr.TP.findings.std, nr.TP.findings.lm, nr.FP.findings.voomclr,
    nr.FP.findings.wilcox.TSS, nr.FP.findings.wilcox.clr, nr.FP.findings.linda,
    nr.FP.findings.ipw, nr.FP.findings.std, nr.FP.findings.lm)))
}

######################### Simulation ##############################
non.parametric.simulation <- function(data=healthy.eurazians.duplicates.rm, k=3, swap.pct=0.3){
    # Function to create simulated data set nonparametric

  # Split the data in 2 groups
  groups <- sample.groups(data=data, swap.pct=swap.pct)
  group1 <- groups[[1]]
  group2 <- groups[[2]]
  meta1 <- groups[[4]]
  meta2 <- groups[[5]]

  # Introduce signal in group 2
  data.group2 <- comp.corr(data.new = group2, data.old=group2, k=k)

  group2 <- data.group2[[1]]
```

```r
  p <- data.group2[[2]]
  q <- data.group2[[3]]

  # Data prep
  group <- as.factor(c(rep("group1", ncol(group1)), rep("group2", ncol(group2))))
  age <- c(meta1$Age, meta2$Age)
  ancestry <- c(meta1$pop_cov, meta2$pop_cov)
  meta <- data.frame(group, age, ancestry)

  Y <- cbind(group1,group2)
  rownames(Y) <- celltypes
  colnames(Y) <- paste0("sample",1:ncol(Y))

  return(list(Y, meta, p, q))
}

parametric.simulation <- function(n=nrow(healthy.eurazians.duplicates.rm), P=11, k=3,
                        og.lib.sizes=rowSums(eurazians.duplicates.rm[,c(11:21)])){
   # Function to create simulated data set nonparametric
  # Make sure we are dividing in even groups
  n <- floor(n/2)*2

  # In each run, simulate cell type composition data
  # I noticed age has a bimodal structure and I want to model as close as possible to the real data
  # set.seed(1234)
  age <- c(ceiling(rmixgamma(n = n/2, weight = c(0.7,0.3), alpha = c(30,65) , beta = c(1,1))),
         ceiling(rmixgamma(n = n/2, weight = c(0.55,0.45), alpha = c(35,55) , beta = c(1,1))))

  ancestry <- c(sample(c("Asian","European"), size=n/2, prob=c(0.25,0.75), replace=T),
                sample(c("Asian","European"), size=n/2, prob=c(0.55,0.45), replace=T))

  # Mean/intercept (negative binomial - mixture of poissons)
  # We don't want a cell type with mean equal to 0 -> otherwise all counts will be zero in every sample
  mu0 <- rep(0, P)
  while (! all(mu0!=0)){
    mu0 <- rnbinom(n=P, size=1/2, mu=400)
  }

  # Sampling of confounding effects - fold change on log scale
  beta.age <- beta.ancestry <- rep(0, P)
  while (all(beta.age==0)){
    beta.age <- rlnorm(n=P, meanlog = -4, sdlog=0.5) * rbinom(n=P, size=1, prob=0.2) *
      sample(c(-1,1), size=P, replace=TRUE)
  }

  while (all(beta.ancestry==0)){
    beta.ancestry <- rlnorm(n=P, meanlog = -1, sdlog=0.5) * rbinom(n=P, size=1, prob=0.2) *
      sample(c(-1,1), size=P, replace=TRUE)
  }

  # Disease effect - fold change on log scale - make sure k cell types get a value different from 0
  beta <- rlnorm(n=P, meanlog = -1, sdlog=1) * sample(c(rep(1,k), rep(0,P-k)), replace=F) *
    sample(c(-1,1), size=P, replace=TRUE)

  # Simulate library sizes (not constant - not realistic)
  sim.lib.sizes <- rpois(n=n, lambda=mean(og.lib.sizes))

  # relative abundance information (observed data in typical experiment)
  Y0 <- Y1 <- matrix(NA, nrow=P, ncol=floor(n/2))
  for (i in 1:(n/2)){
    mu0.i <- mu0 * exp(beta.age*age[i]+beta.ancestry*(ifelse(ancestry[i]=="Asian",1,0)))
    rel.ab <- mu0.i/sum(mu0.i)
    Y0[,i] <- rmultinom(n=1, size=sim.lib.sizes[i], prob=rel.ab)

    mu1.i <- mu0 * exp(beta+beta.age*age[n/2+i]+beta.ancestry*(ifelse(ancestry[n/2+i]=="Asian",1,0)))
    rel.ab <- mu1.i/sum(mu1.i)
    Y1[,i] <- rmultinom(n=1, size=sim.lib.sizes[n/2+i], prob=rel.ab)
  }

  Y <- cbind(Y0, Y1)
  rownames(Y) <- paste0("celltype",1:P)
  colnames(Y) <- paste0("sample",1:n)
  group <- factor(rep(c("group1","group2"), each=n/2))
  meta <- data.frame(group, age, ancestry)

  signal.celltypes <- rownames(Y)[abs(beta)>0]
  betas <- data.frame(beta, beta.age, beta.ancestry)
  return(list(Y, meta, signal.celltypes, betas))}

simulation.confounding <- function(data=healthy.eurazians.duplicates.rm, k=3, B=250,
    seed=2001,  swap.pct=0.3, adjustment="BH", alpha=c(0.01,0.05,0.1),
    meanvar="analytical", distr="NB", stabilized=F, sim = "nonparametric", P=11,
    n=nrow(healthy.eurazians.duplicates.rm),og.lib.sizes=rowSums(eurazians.duplicates.rm[,c(11:21)])){

  set.seed(seed)
  nr.of.signal <- k

  if (sim == "parametric"){
    # Define the number of cell types in the data
    K <- P
    names <- paste0("celltype",1:P)
  } else {
    K <- 11
    names <- celltypes
  }

  ######### Initialize vectors and lists to keep track of results from different methods ############
  # Confounding
  top.k.match <- bias.voomclr <- bias.linda <- data.frame()
  signal <- data.stats <- ci.std <- ci.ipw <- ci.lm <- ci.voomclr <- ci.linda <- ci.linda.lfc <-
    number.findings <- beta <- list()

  coverage.linda <- coverage.voomclr <- coverage.lm <- coverage.ipw <- coverage.std <-
    coverage.linda.lfc <- matrix(0, nrow=K, ncol=length(alpha))

  wilcox.TSS.pval <- wilcox.clr.pval <- voomclr.pval <- linda.pval <- ipw.pval <- lm.pval <-
  wilcox.TSS.stat <- wilcox.clr.stat <- voomclr.coef <- voomclr.t <- voomclr.se <-
  linda.coef <- linda.se <-  ipw.coef <- ipw.se <- ipw.t <- standardization.coef <-
  standardization.se <- adj.p.standardization <- lm.coef <- lm.se <- lm.t <-
  raw.p.wilcox.TSS <- raw.p.wilcox.clr <- raw.p.voomclr <- raw.p.linda <- raw.p.lm <-
  raw.p.ipw <- raw.p.standardization <-  matrix(NA, nrow=B, ncol=K)

  colnames(wilcox.TSS.pval) <- colnames(wilcox.clr.pval) <- colnames(voomclr.pval) <-
  colnames(linda.pval) <- colnames(ipw.pval) <- colnames(lm.pval) <-
  colnames(wilcox.TSS.stat) <- colnames(wilcox.clr.stat) <- colnames(voomclr.coef) <-
  colnames(voomclr.t) <- colnames(voomclr.se) <- colnames(linda.coef) <-
  colnames(linda.se) <- colnames(ipw.coef) <- colnames(ipw.se) <- colnames(ipw.t) <-
  colnames(standardization.coef) <- colnames(adj.p.standardization) <- colnames(standardization.se) <-
  colnames(lm.coef) <- colnames(lm.se) <- colnames(lm.t) <- colnames(raw.p.wilcox.TSS) <-
  colnames(raw.p.wilcox.clr) <- colnames(raw.p.voomclr) <- colnames(raw.p.linda) <-
  colnames(raw.p.lm) <- colnames(raw.p.ipw) <- colnames(raw.p.standardization) <- names


  # Without confounding
  top.k.match.noconf <- bias.voomclr.noconf <- bias.linda.noconf <- data.frame()
  ci.std.noconf <- ci.ipw.noconf <- ci.lm.noconf <- ci.voomclr.noconf <-
  ci.linda.noconf <- ci.linda.noconf.lfc <- number.findings.noconf <- list()

  coverage.linda.noconf <- coverage.linda.noconf.lfc <- coverage.voomclr.noconf <- coverage.lm.noconf <-
  coverage.ipw.noconf <- coverage.std.noconf <- matrix(0, nrow=K, ncol=length(alpha))

 voomclr.pval.noconf <- linda.pval.noconf <- ipw.pval.noconf <- lm.pval.noconf <-
 voomclr.coef.noconf <- voomclr.t.noconf <- voomclr.se.noconf <- linda.coef.noconf <- linda.se.noconf <-
 ipw.coef.noconf <- ipw.se.noconf <- ipw.t.noconf <- standardization.coef.noconf <-
 standardization.se.noconf <- adj.p.standardization.noconf <- lm.coef.noconf <-
 lm.se.noconf <- lm.t.noconf <- raw.p.voomclr.noconf <- raw.p.linda.noconf <-
 raw.p.lm.noconf <- raw.p.ipw.noconf <- raw.p.standardization.noconf <- matrix(NA, nrow=B, ncol=K)

 colnames(voomclr.pval.noconf) <- colnames(linda.pval.noconf) <- colnames(ipw.pval.noconf) <-
 colnames(lm.pval.noconf) <-  colnames(voomclr.coef.noconf) <- colnames(voomclr.t.noconf) <-
 colnames(voomclr.se.noconf) <- colnames(linda.coef.noconf) <- colnames(linda.se.noconf) <-
 colnames(ipw.coef.noconf) <- colnames(ipw.se.noconf) <- colnames(ipw.t.noconf) <-
 colnames(standardization.coef.noconf) <- colnames(adj.p.standardization.noconf) <-
 colnames(standardization.se.noconf) <- colnames(lm.coef.noconf) <- colnames(lm.se.noconf) <-
 colnames(lm.t.noconf) <-colnames(raw.p.voomclr.noconf) <- colnames(raw.p.linda.noconf) <-
 colnames(raw.p.lm.noconf) <- colnames(raw.p.ipw.noconf) <-
 colnames(raw.p.standardization.noconf) <- names

  for (b in 1:B){
    number.findings[[b]] <- data.frame()
    number.findings.noconf[[b]] <- data.frame()

    if (sim == "nonparametric"){
      simulation <- non.parametric.simulation(data=data, k=k, swap.pct=swap.pct)
      Y <- simulation[[1]]
      meta <- simulation[[2]]
      p <- simulation[[3]]
      q <- simulation[[4]]
      betas <- matrix(NA, nrow=11, ncol=3)
    } else {
      simulation <- parametric.simulation(n=n, P=P, k=k,
                                          og.lib.sizes=og.lib.sizes)
      Y <- simulation[[1]]
      meta <- simulation[[2]]
      p <- simulation[[3]]
      betas <- simulation[[4]]
      beta[[b]] <- betas[,1]
    }

  # Data prep
    # clr uses geometric mean calculated only with non-zero counts, linDA and voomCLR use pseudocount 0.5
  # Y.clr <- clr(t(Y))
  # Calculate clr counts
  geoMeans <- exp(colMeans(log(Y+0.5)))
```

```r
Y.clr <- log(t(Y+0.5)/geoMeans)
Y.TSS <- as.data.frame(t(Y %>% microbiome::transform(transform="compositional")))

Y.lm <- as.data.frame(Y.clr)
# Make sure group is a binary vector
Y.lm$group <- ifelse(meta$group=="group1",0,1)
Y.lm$age <- meta$age
Y.lm$ancestry <- meta$ancestry

Y.std <- Y.ipw <- Y.lm

############# Wilcoxon #########################################
p.values.TSS <- p.values.clr <- c(1:K)

for (iter in 1:K){
  wilcox.celltype.TSS <- wilcox.test(x=Y.TSS[meta$group=="group1", iter],
                                     y=Y.TSS[meta$group=="group2", iter] )
  p.values.TSS[iter] <- wilcox.celltype.TSS$p.value
  wilcox.TSS.stat[b, iter] <- wilcox.celltype.TSS$statistic

  wilcox.celltype.clr <- wilcox.test(x=Y.clr[meta$group=="group1",iter],
                                     y=Y.clr[meta$group=="group2",iter] )
  p.values.clr[iter] <- wilcox.celltype.clr$p.value
  wilcox.clr.stat[b, iter] <- wilcox.celltype.clr$statistic
}
raw.p.wilcox.TSS[b,] <- p.values.TSS
raw.p.wilcox.clr[b,] <- p.values.clr
wilcox.TSS.pval[b,] <- p.adjust(p.values.TSS, method=adjustment)
wilcox.clr.pval[b,] <- p.adjust(p.values.clr, method=adjustment)

################################# LinDA ##############################
# Accounting for confounding
lindaRes <- linDA(Y, variables=c("group","age","ancestry"), meta, adjustment)
linda.res <- lindaRes[[1]]
bias.linda <- rbind(bias.linda, lindaRes[[2]])
linda.coef[b,] <- linda.res$log2FoldChange
linda.se[b,] <- linda.res$lfcSE
linda.pval[b,] <- linda.res$padj
raw.p.linda[b,] <- linda.res$pvalue

# Not accounting for confounding
lindaRes.noconf <- linDA(Y, variables=c("group"), meta, adjustment)
linda.res.noconf <- lindaRes.noconf[[1]]
bias.linda.noconf <- rbind(bias.linda.noconf, lindaRes.noconf[[2]])
linda.coef.noconf[b,] <- linda.res.noconf$log2FoldChange
linda.se.noconf[b,] <- linda.res.noconf$lfcSE
linda.pval.noconf[b,] <- linda.res.noconf$padj
raw.p.linda.noconf[b,] <- linda.res.noconf$pvalue

################ VoomCLR ########################################
# Accounting for confounding
voomclr <- voom(Y, variables=c("group","age","ancestry"), adjustment, meta, meanvar, distr)
tt <- voomclr[[1]]
bias.voomclr <- rbind(bias.voomclr,voomclr[[2]])
raw.p.voomclr[b,] <- tt$P.Value
voomclr.pval[b,] <- tt$adj.P.Val
voomclr.coef[b,] <- tt$logFC
voomclr.t[b,] <- tt$t
```

```r
  voomclr.se[b,] <- voomclr.coef[b,]/voomclr.t[b,]
  df.resid <- voomclr[[3]]
  df.prior <- voomclr[[4]]

  # Not accounting for confounding
  voomclr.noconf <- voom(Y, variables=c("group"), adjustment, meta, meanvar, distr)
  tt.noconf <- voomclr.noconf[[1]]
  bias.voomclr.noconf <- rbind(bias.voomclr.noconf, voomclr.noconf[[2]])
  raw.p.voomclr.noconf[b,] <- tt.noconf$P.Value
  voomclr.pval.noconf[b,] <- tt.noconf$adj.P.Val
  voomclr.coef.noconf[b,] <- tt.noconf$logFC
  voomclr.t.noconf[b,] <- tt.noconf$t
  voomclr.se.noconf[b,] <- voomclr.coef.noconf[b,]/voomclr.t.noconf[b,]
  df.resid.noconf <- voomclr.noconf[[3]]
  df.prior.noconf <- voomclr.noconf[[4]]

  ######################## Linear regression ######################
  # Accounting for confounders
  lin.reg <- linear.regression(Y.lm, variables=c("group","age","ancestry"), adjustment, names)
  lin.reg.fits <- lin.reg[[1]]
  lm.coef[b,] <- lin.reg[[2]]$coef
  lm.se[b,] <- lin.reg[[2]]$se
  lm.t[b,] <- lin.reg[[2]]$t
  lm.pval[b,] <- lin.reg[[2]]$pval
  raw.p.lm[b,] <- lin.reg[[2]]$raw.p

  # Not accounting for confounders
  lin.reg.noconf <- linear.regression(Y.lm, variables=c("group"), adjustment, names)
  lin.reg.fits.noconf <- lin.reg.noconf[[1]]
  lm.coef.noconf[b,] <- lin.reg.noconf[[2]]$coef
  lm.se.noconf[b,] <- lin.reg.noconf[[2]]$se
  lm.t.noconf[b,] <- lin.reg.noconf[[2]]$t
  lm.pval.noconf[b,] <- lin.reg.noconf[[2]]$pval
  raw.p.lm.noconf[b,] <- lin.reg.noconf[[2]]$raw.p


  ############### Inverse probability weighting ###################
  # Accounting for confounding
  inverse.p <- inverse.probability.weighting(Y.ipw, variables=c("group","age","ancestry"),
                                             adjustment, names, stabilized)
  ipw.coefs <- inverse.p[[1]]
  ipw.fits <- inverse.p[[2]]
  ipw.coef[b,] <- ipw.coefs$coef.per.celltype
  ipw.se[b,] <- ipw.coefs$se.per.celltype
  ipw.t[b,] <- ipw.coefs$t
  ipw.pval[b,] <- ipw.coefs$p.val.ipw
  raw.p.ipw[b,] <- ipw.coefs$raw.p

  # Not accounting for confounding
  inverse.p.noconf <- inverse.probability.weighting(Y.ipw, variables=c("group"),
                                                    adjustment, names, stabilized)
  ipw.coefs.noconf <- inverse.p.noconf[[1]]
  ipw.fits.noconf <- inverse.p.noconf[[2]]
  ipw.coef.noconf[b,] <- ipw.coefs.noconf$coef.per.celltype
  ipw.se.noconf[b,] <- ipw.coefs.noconf$se.per.celltype
  ipw.t.noconf[b,] <- ipw.coefs.noconf$t
  ipw.pval.noconf[b,] <- ipw.coefs.noconf$p.val.ipw
  raw.p.ipw.noconf[b,] <- ipw.coefs.noconf$raw.p
```

```r


############### Standardization ##################################
# Accounting for confounding
std <- standardization(Y.std, variables=c("group","age","ancestry"), names)
standardization.coefs <- std[[1]]
standardization.fits <- std[[2]]
standardization.deviance <- std[[3]]
standardization.coef[b,] <- standardization.coefs$coef
standardization.se[b,] <- standardization.coefs$se
raw.p.standardization[b,] <- standardization.coefs$raw.p
adj.p.standardization[b,] <- standardization.coefs$adj.p

# Not accounting for confounding
std.noconf <- standardization(Y.std, variables=c("group"), names)
standardization.coefs.noconf <- std.noconf[[1]]
standardization.fits.noconf <- std.noconf[[2]]
standardization.deviance.noconf <- std.noconf[[3]]
standardization.coef.noconf[b,] <- standardization.coefs.noconf$coef
standardization.se.noconf[b,] <- standardization.coefs.noconf$se
raw.p.standardization.noconf[b,] <- standardization.coefs.noconf$raw.p
adj.p.standardization.noconf[b,] <- standardization.coefs.noconf$adj.p

signal.celltypes <- p
signal[[b]] <- signal.celltypes

############ loop over alpha ##############
for (iter in seq_along(alpha)){
  a <- alpha[iter]
  ############## voomCLR #################
  ci <- ci.noconf <- data.frame()
  coverage <- coverage.noconf <- c(1:K)
  for (i in 1:K){
    LL <- voomclr.coef[b,i]-qt(1-a/2, df=df.resid[i]+df.prior)*voomclr.se[b,i]
    UL <- voomclr.coef[b,i]+qt(1-a/2, df=df.resid[i]+df.prior)*voomclr.se[b,i]
    ci <- rbind(ci, c(LL,UL))
    coverage[i] <- ifelse(ci[i,1] < betas[i,1] && betas[i,1] < ci[i,2], 1, 0)

    LL <- voomclr.coef.noconf[b,i]-qt(1-a/2, df=df.resid.noconf[i]+df.prior.noconf)*voomclr.se.noconf[b,i]
    UL <- voomclr.coef.noconf[b,i]+qt(1-a/2, df=df.resid.noconf[i]+df.prior.noconf)*voomclr.se.noconf[b,i]
    ci.noconf <- rbind(ci.noconf, c(LL,UL))
    coverage.noconf[i] <- ifelse(ci.noconf[i,1] < betas[i,1] && betas[i,1] < ci.noconf[i,2], 1, 0)
  }
  rownames(ci) <- rownames(ci.noconf) <- names
  colnames(ci) <- colnames(ci.noconf) <- c("lower","upper")
  ci.voomclr[[length(ci.voomclr)+1]] <- ci
  ci.voomclr.noconf[[length(ci.voomclr.noconf)+1]] <- ci.noconf
  coverage.voomclr[,iter] <- coverage.voomclr[,iter] + coverage
  coverage.voomclr.noconf[,iter] <- coverage.voomclr.noconf[,iter] + coverage.noconf

  ############## linDA #################
  # LinDA gives estimates on log2FC scale
  ci <- ci.noconf <- ci.lfc <- ci.noconf.lfc <- data.frame()
  coverage <- coverage.noconf <- coverage.lfc <- coverage.noconf.lfc <- c(1:K)
  for (i in 1:K){
    LL <- linda.coef[b,i]-qt(1-a/2, df=ncol(Y)-4)*linda.se[b,i]
    UL <- linda.coef[b,i]+qt(1-a/2, df=ncol(Y)-4)*linda.se[b,i]
    ci <- rbind(ci, c(LL,UL))
```

```r
      coverage[i] <- ifelse(ci[i,1] < betas[i,1] && betas[i,1] < ci[i,2], 1, 0)

      # Convert to log fold change scale
      LL <- log(2**LL)
      UL <- log(2**UL)
      ci.lfc <- rbind(ci.lfc, c(LL,UL))
      coverage.lfc[i] <- ifelse(ci.lfc[i,1] < betas[i,1] && betas[i,1] < ci.lfc[i,2], 1, 0)


      # Without accounting for confounding
      LL <- linda.coef.noconf[b,i]-qt(1-a/2, df=ncol(Y)-2)*linda.se.noconf[b,i]
      UL <- linda.coef.noconf[b,i]+qt(1-a/2, df=ncol(Y)-2)*linda.se.noconf[b,i]
      ci.noconf <- rbind(ci.noconf, c(LL,UL))
      coverage.noconf[i] <- ifelse(ci.noconf[i,1] < betas[i,1] && betas[i,1] < ci.noconf[i,2], 1, 0)

      # Convert to log fold change scale
      LL <- log(2**LL)
      UL <- log(2**UL)
      ci.noconf.lfc <- rbind(ci.noconf.lfc, c(LL,UL))
      coverage.noconf.lfc[i] <- ifelse(ci.noconf.lfc[i,1] < betas[i,1] &&
                                        betas[i,1] < ci.noconf.lfc[i,2], 1, 0)

    }
    rownames(ci) <- rownames(ci.noconf) <- rownames(ci.lfc) <- rownames(ci.noconf.lfc) <- names
    colnames(ci) <- colnames(ci.noconf) <- colnames(ci.lfc) <-
    colnames(ci.noconf.lfc) <- c("lower","upper")
    ci.linda[[length(ci.linda)+1]] <- ci
    ci.linda.noconf[[length(ci.linda.noconf)+1]] <- ci.noconf
    ci.linda.lfc[[length(ci.linda.lfc)+1]] <- ci.lfc
    ci.linda.noconf.lfc[[length(ci.linda.noconf.lfc)+1]] <- ci.noconf.lfc
    coverage.linda[,iter] <- coverage.linda[,iter] + coverage
    coverage.linda.noconf[,iter] <- coverage.linda.noconf[,iter] + coverage.noconf
    coverage.linda.lfc[,iter] <- coverage.linda.lfc[,iter] + coverage.lfc
    coverage.linda.noconf.lfc[,iter] <- coverage.linda.noconf.lfc[,iter] + coverage.noconf.lfc

    ############ Linear regression (ci)############
    ci <- ci.noconf <- data.frame()
    coverage <- coverage.noconf <- c(1:K)
    for (i in 1:K){
      ci <- rbind(ci,confint(lin.reg.fits[[i]], level=1-a)[rownames(confint(lin.reg.fits[[i]]))=="group",])
      coverage[i] <- ifelse(ci[i,1] < betas[i,1] && betas[i,1] < ci[i,2], 1, 0)

      ci.noconf <- rbind(ci.noconf,confint(lin.reg.fits.noconf[[i]], level=1-a)
                         [rownames(confint(lin.reg.fits.noconf[[i]]))=="group",])
      coverage.noconf[i] <- ifelse(ci.noconf[i,1] < betas[i,1] && betas[i,1] < ci.noconf[i,2], 1, 0)
    }
    rownames(ci) <- rownames(ci.noconf) <- names
    colnames(ci) <- colnames(ci.noconf) <- c("lower","upper")
    ci.lm[[length(ci.lm)+1]] <- ci
    ci.lm.noconf[[length(ci.lm.noconf)+1]] <- ci.noconf
    coverage.lm[,iter] <- coverage.lm[,iter] + coverage
    coverage.lm.noconf[,iter] <- coverage.lm.noconf[,iter] + coverage.noconf

    ###### Inverse probability weighting (ci) ######
    ci <- ci.noconf <- data.frame()
    coverage <- coverage.noconf <- c(1:K)
    for (i in 1:K){
      ci <- rbind(ci,confint(ipw.fits[[i]], level=1-a)[2,])
      coverage[i] <- ifelse(ci[i,1] < betas[i,1] && betas[i,1] < ci[i,2], 1, 0)

      ci.noconf <- rbind(ci.noconf,confint(ipw.fits.noconf[[i]], level=1-a)[2,])
      coverage.noconf[i] <- ifelse(ci.noconf[i,1] < betas[i,1] && betas[i,1] < ci.noconf[i,2], 1, 0)
    }
    rownames(ci) <- rownames(ci.noconf) <- names
    colnames(ci) <- colnames(ci.noconf) <- c("lower","upper")
    ci.ipw[[length(ci.ipw)+1]] <- ci
    ci.ipw.noconf[[length(ci.ipw.noconf)+1]] <- ci.noconf
    coverage.ipw[,iter] <- coverage.ipw[,iter] + coverage
    coverage.ipw.noconf[,iter] <- coverage.ipw.noconf[,iter] + coverage.noconf

    ################# Standardization ###############################
    ci <- ci.noconf <- data.frame()
    coverage <- coverage.noconf <- c(1:K)
    for (i in 1:K){
      summ <- summary(standardization.fits[[i]], CI.type="plain", CI.level=1-a,
                      contrast="difference", reference=0)
      ci <- rbind(ci, summ$est.table[2,3:4])
      coverage[i] <- ifelse(ci[i,1] < betas[i,1] && betas[i,1] < ci[i,2], 1, 0)

      summ.noconf <- summary(standardization.fits.noconf[[i]], CI.type="plain", CI.level=1-a,
                             contrast="difference", reference=0)
      ci.noconf <- rbind(ci.noconf, summ.noconf$est.table[2,3:4])
      coverage.noconf[i] <- ifelse(ci.noconf[i,1] < betas[i,1] && betas[i,1] < ci.noconf[i,2], 1, 0)
    }
    rownames(ci) <- rownames(ci.noconf) <- names
    colnames(ci) <- colnames(ci.noconf) <- c("lower","upper")
    ci.std[[length(ci.std)+1]] <- ci
    ci.std.noconf[[length(ci.std.noconf)+1]] <- ci.noconf
    coverage.std[,iter] <- coverage.std[,iter] + coverage
    coverage.std.noconf[,iter] <- coverage.std.noconf[,iter] + coverage.noconf

    ########## Evaluation ################

    eval <- evaluation(signal.celltypes, k, tt, wilcox.TSS.pval[b,], wilcox.clr.pval[b,], linda.res,
                       ipw.pval[b,], adj.p.standardization[b,], lm.pval[b,], alpha=a)
    eval.noconf <- evaluation(signal.celltypes, k, tt.noconf, wilcox.TSS.pval[b,], wilcox.clr.pval[b,],
                              linda.res.noconf, ipw.pval.noconf[b,], adj.p.standardization.noconf[b,],
                              lm.pval.noconf[b,], alpha=a)

    number.findings[[b]] <- rbind(number.findings[[b]], eval[[2]])
    number.findings.noconf[[b]] <- rbind(number.findings.noconf[[b]], eval.noconf[[2]])
  }
  # We only need to keep the last one as this doesnt change for alpha
  top.k.match <- rbind(top.k.match, eval[[1]])
  top.k.match.noconf <- rbind(top.k.match.noconf, eval.noconf[[1]])
  rownames(number.findings[[b]]) <- rownames(number.findings.noconf[[b]])<- alpha
  }

colnames(bias.voomclr) <- names(voomclr[[2]])
colnames(bias.voomclr.noconf) <- names(voomclr.noconf[[2]])
colnames(bias.linda) <- c("group","age","ancestry")
colnames(bias.linda.noconf) <- "group"

# proportion of runs where the true signal cell types were in the top k
prop.top.k <- apply(top.k.match, 2, function(x){sum(x)}/B)
prop.top.k.noconf <- apply(top.k.match.noconf, 2, function(x){sum(x)}/B)
```

```r

###### FDR and sensitivity #######
# Accounting for confounding
FDR.wilcox.TSS <- FDR.wilcox.clr <- FDR.voomclr <- FDR.linda <- FDR.lm <- FDR.ipw <-
FDR.standardization <- data.frame()
sensitivity.wilcox.TSS <- sensitivity.wilcox.clr <- sensitivity.voomclr <-
sensitivity.linda <- sensitivity.lm <- sensitivity.ipw <-
sensitivity.standardization <- data.frame()
for (i in 1:B){
  FDR.wilcox.TSS <- rbind(FDR.wilcox.TSS,
  number.findings[[i]]$nr.FP.findings.wilcox.TSS/number.findings[[i]]$nr.significant.celltypes.wilcox.TSS)
  FDR.wilcox.clr <- rbind(FDR.wilcox.clr,
  number.findings[[i]]$nr.FP.findings.wilcox.clr/number.findings[[i]]$nr.significant.celltypes.wilcox.clr)
  FDR.voomclr <- rbind(FDR.voomclr,
  number.findings[[i]]$nr.FP.findings.voomclr/number.findings[[i]]$nr.significant.celltypes.voomclr)
  FDR.linda <- rbind(FDR.linda,
  number.findings[[i]]$nr.FP.findings.linda/number.findings[[i]]$nr.significant.celltypes.linda)
  FDR.lm <- rbind(FDR.lm,
number.findings[[i]]$nr.FP.findings.lm/number.findings[[i]]$nr.significant.celltypes.lm)
  FDR.ipw <- rbind(FDR.ipw,
  number.findings[[i]]$nr.FP.findings.ipw/number.findings[[i]]$nr.significant.celltypes.ipw)
  FDR.standardization <- rbind(FDR.standardization,
  number.findings[[i]]$nr.FP.findings.std/number.findings[[i]]$nr.significant.celltypes.std)

  sensitivity.wilcox.TSS <- rbind(sensitivity.wilcox.TSS,
    number.findings[[i]]$nr.TP.findings.wilcox.TSS/nr.of.signal)
  sensitivity.wilcox.clr <- rbind(sensitivity.wilcox.clr,
    number.findings[[i]]$nr.TP.findings.wilcox.clr/nr.of.signal)
  sensitivity.voomclr<- rbind(sensitivity.voomclr,
    number.findings[[i]]$nr.TP.findings.voomclr/nr.of.signal)
  sensitivity.linda <- rbind(sensitivity.linda,
    number.findings[[i]]$nr.TP.findings.linda/nr.of.signal)
  sensitivity.lm <- rbind(sensitivity.lm,
    number.findings[[i]]$nr.TP.findings.lm/nr.of.signal)
  sensitivity.ipw <- rbind(sensitivity.ipw,
    number.findings[[i]]$nr.TP.findings.ipw/nr.of.signal)
  sensitivity.standardization <- rbind(sensitivity.standardization,
    number.findings[[i]]$nr.TP.findings.std/nr.of.signal)
}

colnames(FDR.wilcox.TSS) <- colnames(FDR.wilcox.clr)<- colnames(FDR.voomclr) <-
colnames(FDR.linda) <- colnames(FDR.lm) <- colnames(FDR.ipw) <- colnames(FDR.standardization) <-
colnames(sensitivity.wilcox.TSS) <- colnames(sensitivity.wilcox.clr)<- colnames(sensitivity.voomclr) <-
colnames(sensitivity.linda) <- colnames(sensitivity.lm) <- colnames(sensitivity.ipw) <-
colnames(sensitivity.standardization)<- alpha

# Make sure that NAs are set to 0 (no significant cell types in that iteration)
FDR.wilcox.TSS[is.na(FDR.wilcox.TSS)] <- FDR.wilcox.clr[is.na(FDR.wilcox.clr)]<-
  FDR.voomclr[is.na(FDR.voomclr)] <- FDR.linda[is.na(FDR.linda)] <- FDR.lm[(is.na(FDR.lm))] <-
  FDR.ipw[is.na(FDR.ipw)] <- FDR.standardization[is.na(FDR.standardization)] <- 0

FDR.wilcox.TSS <- colMeans(FDR.wilcox.TSS)
FDR.wilcox.clr <- colMeans(FDR.wilcox.clr)
FDR.voomclr <- colMeans(FDR.voomclr)
FDR.linda <- colMeans(FDR.linda)
FDR.lm <- colMeans(FDR.lm)
FDR.ipw <- colMeans(FDR.ipw)
FDR.standardization <- colMeans(FDR.standardization)
```

```r

sensitivity.wilcox.TSS <- colMeans(sensitivity.wilcox.TSS)
sensitivity.wilcox.clr <- colMeans(sensitivity.wilcox.clr)
sensitivity.voomclr <- colMeans(sensitivity.voomclr)
sensitivity.linda <- colMeans(sensitivity.linda)
sensitivity.lm <- colMeans(sensitivity.lm)
sensitivity.ipw <- colMeans(sensitivity.ipw)
sensitivity.standardization <- colMeans(sensitivity.standardization)

# Not accounting for confounding
FDR.voomclr.noconf <- FDR.linda.noconf <- FDR.lm.noconf <- FDR.ipw.noconf <-
FDR.standardization.noconf <- data.frame()
sensitivity.voomclr.noconf <- sensitivity.linda.noconf <- sensitivity.lm.noconf <-
sensitivity.ipw.noconf <- sensitivity.standardization.noconf <- data.frame()

for (i in 1:B){
  FDR.voomclr.noconf <- rbind(FDR.voomclr.noconf,
  number.findings.noconf[[i]]$nr.FP.findings.voomclr/number.findings.noconf[[i]]$nr.significant.celltypes.voomclr)
  FDR.linda.noconf <- rbind(FDR.linda.noconf,
  number.findings.noconf[[i]]$nr.FP.findings.linda/number.findings.noconf[[i]]$nr.significant.celltypes.linda)
  FDR.lm.noconf <- rbind(FDR.lm.noconf,
  number.findings.noconf[[i]]$nr.FP.findings.lm/number.findings.noconf[[i]]$nr.significant.celltypes.lm)
  FDR.ipw.noconf <- rbind(FDR.ipw.noconf,
  number.findings.noconf[[i]]$nr.FP.findings.ipw/number.findings.noconf[[i]]$nr.significant.celltypes.ipw)
  FDR.standardization.noconf <- rbind(FDR.standardization.noconf,
  number.findings.noconf[[i]]$nr.FP.findings.std/number.findings.noconf[[i]]$nr.significant.celltypes.std)

  sensitivity.voomclr.noconf <- rbind(sensitivity.voomclr.noconf,
    number.findings.noconf[[i]]$nr.TP.findings.voomclr/nr.of.signal)
  sensitivity.linda.noconf <- rbind(sensitivity.linda.noconf,
    number.findings.noconf[[i]]$nr.TP.findings.linda/nr.of.signal)
  sensitivity.lm.noconf <- rbind(sensitivity.lm.noconf,
    number.findings.noconf[[i]]$nr.TP.findings.lm/nr.of.signal)
  sensitivity.ipw.noconf <- rbind(sensitivity.ipw.noconf,
    number.findings.noconf[[i]]$nr.TP.findings.ipw/nr.of.signal)
  sensitivity.standardization.noconf <- rbind(sensitivity.standardization.noconf,
    number.findings.noconf[[i]]$nr.TP.findings.std/nr.of.signal)
}

colnames(FDR.voomclr.noconf) <- colnames(FDR.linda.noconf) <- colnames(FDR.lm.noconf) <-
colnames(FDR.ipw.noconf) <- colnames(FDR.standardization.noconf) <-
colnames(sensitivity.voomclr.noconf) <- colnames(sensitivity.linda.noconf) <- colnames(sensitivity.lm.noconf) <-
colnames(sensitivity.ipw.noconf) <- colnames(sensitivity.standardization.noconf) <- alpha

FDR.voomclr.noconf[is.na(FDR.voomclr.noconf)] <- FDR.linda.noconf[is.na(FDR.linda.noconf)] <-
FDR.lm.noconf[(is.na(FDR.lm.noconf))] <- FDR.ipw.noconf[is.na(FDR.ipw.noconf)] <-
FDR.standardization.noconf[is.na(FDR.standardization.noconf)] <- 0

FDR.voomclr.noconf <- colMeans(FDR.voomclr.noconf)
FDR.linda.noconf <- colMeans(FDR.linda.noconf)
FDR.lm.noconf <- colMeans(FDR.lm.noconf)
FDR.ipw.noconf <- colMeans(FDR.ipw.noconf)
FDR.standardization.noconf <- colMeans(FDR.standardization.noconf)

sensitivity.voomclr.noconf <- colMeans(sensitivity.voomclr.noconf)
sensitivity.linda.noconf <- colMeans(sensitivity.linda.noconf)
sensitivity.lm.noconf <- colMeans(sensitivity.lm.noconf)
sensitivity.ipw.noconf <- colMeans(sensitivity.ipw.noconf)
sensitivity.standardization.noconf <- colMeans(sensitivity.standardization.noconf)

return (list(signal, prop.top.k, number.findings,
             data.frame(sensitivity.wilcox.TSS, sensitivity.wilcox.clr,
                        sensitivity.voomclr, sensitivity.linda, sensitivity.lm,
                        sensitivity.ipw, sensitivity.standardization,
                        FDR.wilcox.TSS, FDR.wilcox.clr, FDR.voomclr,
                        FDR.linda, FDR.lm, FDR.ipw, FDR.standardization),
             wilcox.TSS.pval, wilcox.clr.pval, wilcox.TSS.stat, wilcox.clr.stat,
             voomclr.pval, voomclr.coef, voomclr.t, ci.voomclr, coverage.voomclr, bias.voomclr,
             linda.pval, linda.coef,linda.se, ci.linda, coverage.linda, bias.linda,
             ipw.pval, ipw.coef, ipw.se, ipw.t, ci.ipw, coverage.ipw,
             standardization.coef, adj.p.standardization, ci.std, coverage.std,
             lm.pval, lm.coef, lm.se, lm.t, ci.lm, coverage.lm,
             raw.p.wilcox.TSS, raw.p.wilcox.clr, raw.p.voomclr, raw.p.linda, raw.p.lm, raw.p.ipw,
             raw.p.standardization, prop.top.k.noconf, number.findings.noconf,
             data.frame(sensitivity.voomclr.noconf, sensitivity.linda.noconf,  sensitivity.lm.noconf,
                        sensitivity.ipw.noconf, sensitivity.standardization.noconf,FDR.voomclr.noconf,
                        FDR.linda.noconf, FDR.lm.noconf, FDR.ipw.noconf, FDR.standardization.noconf),
             voomclr.pval.noconf, ci.voomclr.noconf, coverage.voomclr.noconf, bias.voomclr.noconf,
             linda.pval.noconf, ci.linda.noconf,  coverage.linda.noconf, bias.linda.noconf,
             ipw.pval.noconf, ci.ipw.noconf, coverage.ipw.noconf,
             adj.p.standardization.noconf, ci.std.noconf, coverage.std.noconf,
             lm.pval.noconf, ci.lm.noconf, coverage.lm.noconf,
             raw.p.voomclr.noconf, raw.p.linda.noconf, raw.p.lm.noconf, raw.p.ipw.noconf,
             raw.p.standardization.noconf, beta, voomclr.coef.noconf, linda.coef.noconf, ci.linda.lfc,
             coverage.linda.lfc, ci.linda.noconf.lfc, coverage.linda.noconf.lfc
))
}


###### Non parametric simulation ########
B <- 250
k=3
nonparam <- simulation.confounding(B=B, k=k, sim="nonparametric", alpha=c(0.01,0.05,0.1))

# Generation of sensitivity and FDR plot
library(reshape2)
colnames(accuracy.noconf) <- sub(".noconf","*",colnames(accuracy.noconf))
accuracy.data <- cbind(accuracy, accuracy.noconf)

accuracy.data$alpha <- rownames(accuracy)

sensitivity_data <- melt(accuracy.data, id.vars = 'alpha',
    measure.vars = grep('sensitivity', names(accuracy.data), value = TRUE))
sensitivity_data$alpha <- as.numeric(paste(sensitivity_data$alpha))
sensitivity_data$value <- as.numeric(paste(sensitivity_data$value))
sensitivity_data$variable <- sub("sensitivity.","", sensitivity_data$variable)

fdr_data <- melt(accuracy.data, id.vars = 'alpha',
    measure.vars = grep('FDR', names(accuracy.data), value = TRUE))
fdr_data$value <- as.numeric(paste(fdr_data$value))
fdr_data$alpha <- as.numeric(paste(fdr_data$alpha))
fdr_data$variable <- sub("FDR.","", fdr_data$variable)

# FDR
fdr_data <- fdr_data %>% arrange(value)
fdr_data$group <- factor(fdr_data$variable, levels = rev(unique(fdr_data$variable)))
```

```r

ggplot(fdr_data, aes(x = alpha, y = value, color = group)) +
  geom_line(linewidth=1) +geom_point(aes(x = alpha, y = alpha), size = 4, shape = 21, fill = "white") +
  scale_x_continuous(breaks=c(0.01,0.05,0.1))+labs(x="Significance level",y="FDR",col="Method") +
  theme_bw(base_size=12)

# Sensitivity
sensitivity_data <- sensitivity_data %>% arrange(value)
sensitivity_data$group <- factor(sensitivity_data$variable, levels = rev(unique(sensitivity_data$variable)))

ggplot(sensitivity_data, aes(x = alpha, y = value, color = group)) +  geom_line(linewidth=1)  +
  scale_x_continuous(breaks=c(0.01,0.05,0.1))+labs(x="Significance level",y="Sensitivity",col="Method") +
  theme_bw(base_size=12)

# Bias terms linDA and voomCLR
bias <- data.frame(bias=c(voomclr.bias$groupgroup2, linda.bias$group,
                          voomclr.bias.noconf$groupgroup2, linda.bias.noconf$group),
                   method=c(rep(c(rep("voomCLR",B), rep("linDA",B)),2)),
                   confounding=c(rep("Accounting for confounding",2*B),
                                 rep("Not accounting for confounding",2*B)))

means <- bias %>%
  group_by(method,confounding) %>%
  summarise(mean = mean(bias), sd=sd(bias))


ggplot(bias, aes(x=bias, col=method)) +
  facet_wrap(~confounding) +
  geom_density(linewidth=1) +
  theme_bw(base_size=15)

# ROC curves
names <- paste(celltypes, rep(1:B, each=11), sep="_")

pval <- data.frame(wilcox.TSS = as.vector(t(raw.p.wilcox.TSS)),
                   wilcox.clr = as.vector(t(raw.p.wilcox.clr)),
                   voomclr = as.vector(t(raw.p.voomclr)),
                   linda = as.vector(t(raw.p.linda)),
                   lm = as.vector(t(raw.p.lm)),
                   ipw = as.vector(t(raw.p.ipw)),
                   std = as.vector(t(raw.p.std)),
                   row.names=names)

padj <- data.frame(wilcox.TSS=as.vector(t(wilcox.TSS.adj.pval)),
                   wilcox.clr=as.vector(t(wilcox.clr.adj.pval)),
                   voomclr=as.vector(t(voomclr.adj.pval)),
                   linda= as.vector(t(linda.adj.pval)),
                   lm = as.vector(t(lm.adj.pval)),
                   ipw=as.vector(t(ipw.adj.pval)),
                   std=as.vector(t(std.adj.pval)),
                   row.names=names)

pval.noconf <- data.frame(wilcox.TSS = as.vector(t(raw.p.wilcox.TSS)),
                   wilcox.clr = as.vector(t(raw.p.wilcox.clr)),
                   voomclr = as.vector(t(raw.p.voomclr.noconf)),
                   linda = as.vector(t(raw.p.linda.noconf)),
                   lm = as.vector(t(raw.p.lm.noconf)),
                   ipw = as.vector(t(raw.p.ipw.noconf)),
                    std = as.vector(t(raw.p.std.noconf)),
                    row.names=names)

padj.noconf <- data.frame(wilcox.TSS=as.vector(t(wilcox.TSS.adj.pval)),
                    wilcox.clr=as.vector(t(wilcox.clr.adj.pval)),
                    voomclr=as.vector(t(voomclr.adj.pval.noconf)),
                    linda= as.vector(t(linda.adj.pval.noconf)),
                    lm = as.vector(t(lm.adj.pval.noconf)),
                    ipw=as.vector(t(ipw.adj.pval.noconf)),
                    std=as.vector(t(std.adj.pval.noconf)),
                    row.names=names)

truth <- c()
for (i in 1:B){
  index <- which(celltypes%in%signal[[i]])
  seq <- rep(0, 11)
  seq[index] <- 1
  truth <- c(truth, seq)
}
truth=data.frame(status=truth, row.names=names)


cobra <- COBRAData(pval = pval,
                   padj = padj,
                   truth = truth)
perf <- calculate_performance(cobra, binary_truth="status")
cobraplot <- prepare_data_for_plot(perf, facetted=F)
plot_roc(cobraplot, title=paste0("Nonparametric simulation\nk=",k))+ylim(c(0.5,1))

cobra.noconf <- COBRAData(pval = pval.noconf,
                   padj = padj.noconf,
                   truth = truth)
perf.noconf <- calculate_performance(cobra.noconf, binary_truth="status")
cobraplot.noconf <- prepare_data_for_plot(perf.noconf, facetted=F)
plot_roc(cobraplot.noconf, title=paste0("Nonparametric simulation (no adjustment)\nk=",k))+ylim(c(0.5,1))



##### Parametric simulation #####
B <- 250

# Setting A
n <- 90
P <- 11
k <- 3
paramA <- simulation.confounding(B=B, sim="parametric", P=P, n=n, k=k, alpha=c(0.01,0.05,0.1))

# # Setting B
n <- 90
P <- 11
k <- 6
paramB <- simulation.confounding(B=B, sim="parametric", P=P, n=n, k=k, alpha=c(0.01,0.05,0.1))

# Setting C
n <- 90
P <- 30
k <- 6
paramC <- simulation.confounding(B=B, sim="parametric", P=P, n=n, k=k, alpha=c(0.01,0.05,0.1))
```

```r

# Setting D
n <- 20
P <- 11
k <- 3
paramD <- simulation.confounding(B=B, sim="parametric", P=P, n=n, k=k, alpha=c(0.01,0.05,0.1))

# Coverage
conf.level <- c(0.99,0.95,0.90)

voomclr.coverage.percentage <- voomclr.coverage/B
voomclr.coverage.percentage.noconf <- voomclr.coverage.noconf/B
linda.coverage.percentage <- linda.coverage.lfc/B
linda.coverage.percentage.noconf <- linda.coverage.noconf.lfc/B
lm.coverage.percentage <- lm.coverage/B
lm.coverage.percentage.noconf <- lm.coverage.noconf/B
ipw.coverage.percentage <- ipw.coverage/B
ipw.coverage.percentage.noconf <- ipw.coverage.noconf/B
std.coverage.percentage <- std.coverage/B
std.coverage.percentage.noconf <- std.coverage.noconf/B

rownames(voomclr.coverage.percentage) <- rownames(voomclr.coverage.percentage.noconf) <-
rownames(linda.coverage.percentage) <- rownames(linda.coverage.percentage.noconf) <-
rownames(lm.coverage.percentage) <- rownames(lm.coverage.percentage.noconf) <-
rownames(ipw.coverage.percentage) <- rownames(ipw.coverage.percentage.noconf) <-
rownames(std.coverage.percentage) <- rownames(std.coverage.percentage.noconf) <-
paste("Celltype", 1:nrow(voomclr.coverage))

colnames(voomclr.coverage.percentage) <- colnames(voomclr.coverage.percentage.noconf) <-
colnames(linda.coverage.percentage) <- colnames(linda.coverage.percentage.noconf) <-
colnames(lm.coverage.percentage) <- colnames(lm.coverage.percentage.noconf) <-
colnames(ipw.coverage.percentage) <- colnames(ipw.coverage.percentage.noconf) <-
colnames(std.coverage.percentage) <- colnames(std.coverage.percentage.noconf) <- conf.level

names <- paste("Celltype",1:P,sep="")
coverage <- data.frame(coverage = c(as.vector(voomclr.coverage.percentage),
                                    as.vector(voomclr.coverage.percentage.noconf),
                                    as.vector(linda.coverage.percentage),
                                    as.vector(linda.coverage.percentage.noconf),
                                    as.vector(lm.coverage.percentage),
                                    as.vector(lm.coverage.percentage.noconf),
                                    as.vector(ipw.coverage.percentage),
                                    as.vector(ipw.coverage.percentage.noconf),
                                    as.vector(std.coverage.percentage),
                                    as.vector(std.coverage.percentage.noconf)),
                       method = c(rep("voomCLR", length(alpha)*P), rep("voomCLR", length(alpha)*P),
                                  rep("linDA", length(alpha)*P), rep("linDA", length(alpha)*P),
                                  rep("Linear regression", length(alpha)*P),rep("Linear regression", length(alpha)*P),
                                  rep("IPW", length(alpha)*P), rep("IPW", length(alpha)*P),
                                  rep("Standardization", length(alpha)*P),rep("Standardization", length(alpha)*P)),
                       confounding = rep(c(rep("Accounting for confounding",length(alpha)*P),
                       rep("Not accounting for confounding",length(alpha)*P)), 5),
                       conf.level = rep(conf.level, 10, each=P),
                       row.names = paste(names, rep(1:30, each=P), sep="_"))


colors <- c("voomCLR" = "red", "linDA" = "pink",
             "Linear regression" = "green",
             "IPW" = "purple",
             "Standardization" = "orange")

order <- coverage %>% filter(confounding=="Accounting for confounding") %>% group_by(method) %>%
summarize(mean=mean(coverage)) %>% arrange(-mean)
coverage$group <- factor(coverage$method, levels=order$method)

ggplot(coverage, aes(x=conf.level, y=coverage, color=group)) +
  geom_point() +
  geom_point(aes(x = conf.level, y = conf.level), size = 4, shape = 21, fill = "white", col="black") +
  scale_x_continuous(breaks=conf.level) +
  geom_smooth(se=F) +
  scale_color_manual(values=colors) +
  facet_wrap(~confounding) +
  theme_bw(base_size=12) +
  labs(title=paste0("Parametric simulation \nP=", P, ",n=", n, ",k=", k),
       x="Confidence level", y="Coverage percentage", color="Method") +
  lims(y=c(0,1))
```

F R code implementation case study

```r

########## VoomCLR ##########
library(limma)
library(voomCLR)
group <- factor(ifelse(eurazians.duplicates.rm$SLE_status=="SLE",1,0))
ancestry <- factor(ifelse(eurazians.duplicates.rm$pop_cov=="European",1,0))
age <- eurazians.duplicates.rm$Age
design <- model.matrix(~group+ancestry+age)
v <- voomCLR(counts = t(eurazians.duplicates.rm[,c(11:21)]),
             design = design,
             varCalc = "analytical",
             varDistribution = "NB",
             plot = TRUE,
             span = 0.8)
fit <- lmFit(v, design)
fit <- eBayes(fit)
tt <- topTableBC(fit, coef=2, n=Inf, bootstrap="nonparametric")
head(tt)

######### linDA ##########
linDA <- MicrobiomeStat::linda(feature.dat = t(eurazians.duplicates.rm[,c(11:21)]),
                                meta.dat = eurazians.duplicates.rm[,-c(11:21)],
                                formula = '~SLE_status+Age+pop_cov',
                                feature.dat.type = 'count',
                                adaptive=TRUE,
                                zero.handling = 'pseudo-count',
                                p.adj.method="BH")
res <- linDA$output$SLE_statusSLE %>% arrange(padj)
lfc <- log(2**(res[["log2FoldChange"]]))
names <- rownames(linDA$output$SLE_statusSLE %>% arrange(padj))
```