MEASUREMENT ERROR DECONVOLUTION METHODS AND RANK SELECTION FOR NON-NEGATIVE MATRIX FACTORIZATION WITH APPLICATIONS IN MICROBIOME DATA

by

Yun Cai

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
August 2022

Table of Contents

List of Tables

List of Figures

Abstract

Learning the structure of microbial communities is critical in understanding the different community structures and functions of microbes in distinct individuals. We view microbial communities as consisting of many subcommunities which are formed by certain groups of microbes functionally dependent on each other. This work studies the structure of microbial community data using the technique Non-negative Matrix Factorisation (NMF).

The supervised NMF method for detecting the differences between microbial communities was developed in my MSc. thesis. However, the interpretation of the resulting factorizations were not considered, and the study of the performance of the method was very limited. In Chapter 2 of this thesis, we review the supervised NMF from my MSc. thesis, then perform extensive simulation studies and real data analyses to better understand the interpretation and the performance of the method under a wide range of scenarios.

One difficulty involved in using NMF is that there is not an accurate method to select the rank for NMF. The rank corresponds to the number of subcommunities, and is thus fundamentally important in interpreting the microbiome data. In order to develop a suitable method to infer the number of ranks for NMF, we further developed a deconvolution method to remove the convergence error in NMF results.

Chapter 3 develops a new method for the deconvolution problem. Deconvolution is the problem of estimating the distribution of a quantity from a sample with additive measurement error. Deconvolution has a wide number of applications, so this work is of very general interest. Our new deconvolution method is based on maximizing log likelihood with a smoothness penalty (PMLE-decon). We develop both the method and the associated asymptotic theory for PMLE deconvolution, and provide an R package for general deconvolution distribution estimation. Through simulations and real data examples, we show that our new method has much better performance than existing methods, particularly for small sample size or low signal-noise ratio. Our method can be applied both with known or parametrically estimated error distribution, and with empirical error distribution, estimated from a pure error sample.

Finally, we develop a novel rank selection method based on hypothesis testing, using a deconvolved bootstrap distribution to assess the significance level accurately despite the large amount of optimisation error. Through simulations, we demonstrate that our method is not only accurate at estimating the true ranks for NMF but also efficient at computation compared with other methods, especially when the features are hard to distinguish. With the newly developed more accurate rank selection method for NMF, we re-analyze the microbiome data we worked on earlier and improve our understanding of microbial sub-communities.

Acknowledgements

I first and foremost want to thank my supervisor Dr. Hong Gu and Dr. Toby Kenney for their support and guidance during my graduate study. I would like to thank Hong for her trust in my potential and offering an opportunity for me to work on statistics. Her generous support in both time and finance allowed me to be fully engaged in my research. She is not only an advisor in academic but also a mentor in life who walk me through the journey of my master and PhD. program. Her patience and optimism encouraged me to overcome all the difficulties I've met over these years. I am also grateful for Toby's inspiration in the thesis. He is objective and inventive all the time during our weekly meetings and discussions. I am always amazed by his talent in many fields including mathematics and statistics. It's impossible to finish my thesis without his extensive knowledge and unreserved help, especially for the Theory section in Chapter 3, which is mainly led by him. I owe a big thank for his hand-on supervision in the proof of the theorems.

I would also like to thank my committee members Dr. Edward Susko and Dr. Andrew Irwin for reading my thesis and their insightful suggestions and comments.

Finally, I want to acknowledge my parents. They always support me in pursuing my interest in statistics. They believe in me and are proud of me no matter what choice I've made. Without their love and care, I will never achieve thus far.

Chapter 1

Introduction

Microbes affect human physiology and global nutrient cycling, through the action of microbial communities [2] [32] [93]. A microbial community usually consists of hundreds or even thouands of different microorganisms [14] [35] which survive through the interaction with each other and environments and form metabolically integrated communities [86]. Although in some cases the abundance of a single species can have a big effect on the overall state of the community, for example some species of pathogens are believed to single-handedly cause illnesses, in many cases, differences between different types of microbial communities (for example, the communities in the guts of healthy and IBD people) are attributable to the overall structure of the community. It is therefore critical to devise models which take into account this overall structure.

Next generation sequencing has generated a large amount of microbial metagenomics data for the study of microbial diversity of different environments. These data consist of either marker-gene data (counts of OTUs) or functional metagenomic data, i.e. counts of reaction-coding enzymes. The OTUs or gene counts will be referred to as variables, and a sample will be also referred to as an observation in this thesis. Considering the difficulty of collecting data and the large number of variables, the data always consist of hundreds or even thousands of variables but only a few observations, which means $p \gg n$ (p is the number of variables and n is the number of observations). In addition, many species are only observed in a few samples, thus the data are highly sparse [40] [54]. This makes it challenging to apply classical statistical analysis methods.

Exploratory data analysis, such as PCA [90], based on the Euclidean distance between samples, is not appropriate for count data and has largely been replaced by clustering analysis or principal coordinates analysis based on Unifrac [72]. The

unifrac distance measures the abundance difference between two samples, incorporating phylogenetic tree information between the organisms. Although Unifrac is widely used, it has some drawbacks. One is that it doesn't address the heterogeneity between samples due to the different sequencing depths for different samples. Sequencing depth in the high-throughput next generation sequencing (NGS) is typically used to describe the total read of DNAs from a sample. Thus this quantity is related to the total DNAs extracted from the sample, the configuration of the sequencer and the upstream bioinformatics tools used to extract the counts of different microbes. It is an unknown quantity which is different for different samples, but most often being approximated by the total count of all microbes in a sample in the down stream data analysis. Thus in this thesis we refer sequencing depth to the total count of all microbes in a sample. Subsampling techniques are sometimes used to attempt to remedy this problem, but these do not fully resolve the problem and involve throwing away a large amount of information in the data and so are not recommended [75]. Unifrac based methods are only applicable to OTU data, not whole metagenome sequence data. Furthermore, Unifrac is an *ad hoc* method in that it is not based on a probablistic model, thus does not provide as much insight as an explicit statistical model-based approach.

Early work on the probabilistic modeling of microbial metagenomics data by Holmes, Harris and Quince [44] has represented the data as multinomial samples from an underlying multinomial distribution which in turn is generated from one of several Dirichlet mixture components. The hyperparameters of each of the Dirichlet mixture components have been assumed to follow a Gamma prior. This Bayesian probability framework seemed to be reasonable, though some assumptions such as choice of prior are arbitrary; however the analysis results of the two examples based on this probability framework in [44] are not totally convincing in that the clustering results of lean and obese samples don't really show clustering patterns, and the method underperforms existing methods at classification. Another Bayesian probabilistic framework models the contaminated sample as a mixture of several known microbial community sources [55]. Recently a Bayesian hierarchical mixed-membership model, called BioMiCo [95], for Bayesian inference of microbial communities, has been developed. BioMiCo takes OTU abundances as

input, and models each sample as a two-level hierarchy of mixtures of multinomial distributions which are constrained by Dirichlet priors. This model can be used to infer both the mixing of OTUs within assemblages, and the mixing of assemblages within samples relative to features shared by multiple samples. Unlike the Gamma prior used in [44], the Dirichlet priors are used to control the sparsity of mixing probabilities for both levels of the multinomial distributions which results in more interpretable assemblages and a more parsimonious model.

Although the above probability frameworks have been mainly applied to marker-gene data, in principle, they can be applied to metagenomic data as well. Unlike the Unifrac based approach, the heterogeneity among samples due to the different sequencing depths is incorporated into the likelihood. Some studies (for example Leibold, Holyoak and Mouquet [65]) suggested that the assembly and structure of bacterial communities should be based on functional genes rather than species. However, directly applying the above models will not utilize the information of the metabolic reactions that can be inferred from those metagenome sequences that encode an enzyme. BiomeNet [96] is another Bayesian hierarchical mixed-membership model to exploit abundance data for metabolic reactions inferred from the metagenome sequences. The corresponding reactions are decomposed into substrate-product pairs, with dependence modelled via shared membership in a subnetwork. Metabolism is then modelled as a three-level hierarchy, with components from one level (e.g., reactions) contributing to other structures (e.g., different subnetworks, metabosystems, and samples) to different degrees.

A common theme in these Bayesian probability modelling frameworks is that each sample is modeled as a mixture of several typical "types". These typical "types" are mostly inferred from data by computational methods. The Bayesian framework provides a natural vehicle for fitting complicated models, but the resulting models are generally not easy to interpret and the computation usually takes a very long time. In addition, the validity of Bayesian inference, just like the likelihood based inference, highly relies on the model checking after calculating the posterior distributions of the model parameters. Due to the high dimensionality of the parameters in such a modeling framework, the model checking is actually very difficult. Due to this, or for other reasons, model checking in the above cited

literature is generally lacking.

In order to provide an effective exploratory data analysis method that is suitable for both marker-gene and functional metagenomic data and is based on a probability model that can capture the subcommunity structure information and can address the issues of heterogeneity among samples, we explore the application of Non-negative Matrix Factorization (NMF) to microbiome data in a likelihood framework. NMF has been widely applied in many areas, such as image and natural language processing, and also has found many applications in computational biology [25]. More recently, it was applied in the ocean microbes data to investigate the biogeography of microbial function and its correlation to environmental distance [48]. Conceptually, similar to the above Bayesian modeling frameworks, NMF models each sample as a mixture of different types. These types represent the structure of subcommunities. Instead of using a multi-level hierarchical structure as in BioMiCo [95] and BiomeNet [96], NMF uses one level of subcommunities as building blocks which makes the connection between the sample microbiome composition and the OTUs or reaction-coding enzymes more direct, this will provide better interpretability for the analysis results. In addition, NMF is a natural method to use for dimension reduction and feature selection in microbiome data. The commonly used unsupervised learning methods such as PCA and vector quantization (VQ) for reducing dimension and picking up the main features of the data usually result in linear combinations with negative coefficients which are hard to interpret naturally in this context. We want to find the main features (subcommunities) of the data and at the same time keep all the elements in these features positive. As demonstrated by Lee and Seung [63], NMF also tends to identify sparse features, and thus each sample is expressed as a non-negative linear combination of a few sparse features (types), which further facilitates the interpretation of the results.

Like PCA or BiomeNet, NMF is an unsupervised method. Although NMF can extract the main features from the data, it can't guarantee that these features are the best discriminant features to distinguish different classes. For example, if two classes are described by similar features, NMF will extract an average of these features to fit both classes, rather than separate features for the two classes. For

the purpose of identifying differences between different types of communities, we developed a supervised version of NMF.

There are many off-the-shelf supervised learning methods that can perform a classification directly on such data (a review is given by [54]). Since typically $p \gg n$ (the number of predictor variables is substantially larger than the number of data points), we need to choose methods designed for the $p \gg n$ case. Directly applying these classification methods often results in quite good classification. With some classification methods (for example random forest and the elastic net), variable selection is also possible. However the selected variables are often difficult to map back to some discriminating community level features between classes, particularly if the true discriminating feature is not a single variable. Some classification methods (such as support vector machine, boosted trees or Neural network) can construct a very good classifier for such data, but without any possibility of interpretation and thus can't provide any insight for the underlying community structure. BioMiCo [95] builds a classifier on the discriminant assemblages of the OTUs to predict the class labels with these assemblages showing the subcommunity structures. The model complexity of BioMiCo is controlled by the number of assemblages and the Dirichlet priors which are both pre-specified. These pre-specified parameters in principle can be adapted to the data through cross-validation on the training data, but running these Bayesian models needs a long time for each run which hurts the wide applicability of BioMiCo to different data.

Since we are interested in the community level features or systematic differences between different classes, we first use NMF to identify features from each class, and then we build a classifier based on the weights distribution of each sample on the combined features from different classes. The features selected by this method will describe the original data well and also contain classification information. We can measure how well the features identified relate to the differences between different types of communities by looking at the misclassification error of classifiers. As mentioned above, the purpose of NMF is to provide insights into the structural differences between different types of microbial communities, rather than to produce the most accurate classification possible. Classification is however

a good measure to gauge the extent to which the subcommunities identified have important biological roles in the overall community structure.

Supervised NMF has similar model structures to BioMiCo, but is fast to compute and the only tuning parameters are the number of features that are extracted from different classes. Unlike BioMiCo which controls the sparsity of the features by the Dirichlet priors, the sparsity of NMF is decided by the number of features. With fewer features used in the model, each feature tends to be less sparse and conversely more features means each feature is more sparse. This flexibility of NMF results means that there may be a range of proper values for the number of features in the model. This makes the model more robust.

The only tuning parameter in the NMF algorithm is the rank of the feature matrix k, which is also the number of features. The interpretation of k is the number of underlying networks or subcommunities extracted from the data. The selection of k not only affects the performance of NMF but also relates to the interpretation of NMF results. Too small k value may lead to key features missing and too large k value may have over-fitting issues. It is therefore necessary to select an appropriate rank for the NMF model. We develop a goodness-of-fit test based on the likelihood ratio statistic. Because NMF is not a standard unconstrained MLE, the asymptotic distribution of the likelihood ratio statistic cannot be derived. We therefore use a bootstrap to determine the critical value. Furthermore, the optimisation in NMF is very unreliable, so our bootstrap values are subject to convergence error. To obtain a better estimate of the null distribution, we perform deconvolution to estimate the null distribution from the sample with measurement error.

Most existing deconvolution techniques like `decon` [105] and `deamer` [49] are based on Fourier transformation, whose division form can cause instability in the estimates when the denominator is small. We develop a more stable deconvolution method by maximizing the penalized log-likelihood (penalized MLE) with the common smoothness penalty. Although we target on removing the convergence errors in NMF rank selection procedure, the deconvolution method can also be applied to remove additive measurement errors in several other areas, e.g. the clinical research, bioinformatics and microscopy. Comte et al. [20] estimate the true

probability distribution of date of onset of pregnancy by applying deconvolution method to date of pregnancies estimated by ultrasound. Mendelsohn and Rice [79] consider to retrieve the true DNA distribution from the DNA content measurements obtained by flow microfluorometry technique through deconvolution. They are concerned that the true distribution is not the recorded distribution due to error from variable stain absorption, light scattering and other sources. Odiachi and Peirve [84] explore the idea of deconvolution for removing background noise from light scattering intensity obtained in total internal reflection microscopy experiments. In addition to use our penalized MLE deconvolution method to build a new method for selecting the number of NMF types, we also show its general application by an additional example in Chapter 3 to estimate the true density of the male subjects' blood pressure from the contaminated Framingham data [15].

1.1 Organization of the Thesis

Chapter 2 details the application of supervised and unsupervised NMF in microbiome data. We apply the existing unsupervised NMF method and the supervised NMF method from my MSc. thesis for extracting interpretable information from classification problems in real data. Then we compare the predictive accuracy for classification of supervised NMF with Random Forest and Support Vector Machine in both simulated and real data examples. We apply supervised NMF on three real data sets: the whole metagenomes of various mammals [81], the time series microbiome data from various human body sites [14] and whole metagenome data of IBD patients and healthy controls [88]. Results of these real data analysis are interpreted in terms of Biosynthesis pathways.

Chapter 3 proposes a new deconvolution method based on maximizing log-likelihood with a smoothness penalty (P-MLE). Deconvolution is the problem of estimating the distribution of a quantity from a sample of observations that are subject to additive measurement error. Our approach is to apply penalized MLE with penalty on the smoothness. We then show the consistency of the estimators and compare our method with existing deconvolution methods in simulation experiments and real data analysis.

In Chapter 4, we develop a goodness-of-fit test to select the rank of NMF based

on a deconvolved bootstrapping distribution. The null hypothesis is that the data are drawn from an NMF with the current rank, while the alternative hypothesis is that increasing the rank by 1 will improve the fit to the data. We use the likelihood ratio statistic for this test. However, the null distribution cannot be obtained in theory, so we obtain it via a parametric bootstrap. Because the optimisation of NMF is unreliable, our bootstrap samples are subject to convergence error. Assuming this convergence error is additive, we can use deconvolution to estimate the null distribution of the test statistic with this convergence error removed. We compare our method with a bootstrap without deconvolution for finding the critical value, and with a cross-validated imputation method on a large number of simulated datasets. We also applied the new rank selection method to real microbiome data (e.g. OTU data and functional metagenomic data) in Section 4.4 and interpret the features captured.

Chapter 2

Learning Microbial Community Structures with Supervised and Unsupervised Non-negative Matrix Factorization

2.1 Review of my MSc. thesis work

This section is a necessary review for the methods developed in my MSc. thesis for the completeness of the results presented in this chapter. We first give a review of NMF. Then we describe the idea of supervised NMF based on unsupervised NMF, with the computation of the weight matrix over the combined features, followed by the method used to choose the tuning parameters for the supervised NMF.

2.1.1 The NMF model

Non-negative matrix factorization [63] is a dimension reduction method for non-negative data. The idea is to represent each data point as a linear combination of non-negative features which are computed from the data. Given a non-negative $p \times n$ matrix X, we approximate X by TW through minimizing the Kullback-Leribler divergence between distributions with mean as X and distributions with mean as TW, where T is a non-negative $p \times k$ matrix, referred to as the *type* (feature) matrix, and W is a non-negative $k \times n$ weight matrix. Each column of X is approximated by a non-negative linear combination of the types (columns of T). Here k is the NMF rank or number of features which determines the complexity of the model. It is a tuning parameter in this context. Usually k is chosen such that $(p+n)k \ll np$, so that we reduce the dimension significantly.

In our analysis, X is microbe data with counts of OTUs or genes. Specifically, X_{ij} is the number of times the ith OTU or gene is observed in the jth sample. Thus each feature (column) in T describes a subcommunity and each column in W contains the linear coefficients for the corresponding sample (column) in X.

The whole community in a sample is thus approximated by a mixture of the subcommunities. For count data, such as our X, we model each element as a conditional independent Poisson observation given its mean in the matrix TW, i.e. the dependency in data is expressed in matrix W and matrix T. Thus the covariance structure between the variables in X is implicitly given by the patterns in the type matrix T. The columns of the type matrix T are constrained to have sum 1, and in this context each column in T can be interpreted as the composition of OTUs or genes for each *type*. The different sequencing depths for the samples in X are absorbed in the weight matrix W. To compute T and W we use an iterative algorithm where we iterate between maximizing

$$L(T, W) = \sum_{i,j} (X_{ij} \log(TW)_{ij} - (TW)_{ij})$$

over W, holding T fixed and then maximizing over T holding W fixed [94],

In most literature (e.g. Lee and Seung [94]) Euclidean distance is used as a criterion, which is appropriate when the observations are assumed to follow a Gaussian distribution with the mean given by TW.

Many authors have extended NMF to include additional constraints. For example, sparsity constraints are useful when the degree of sparseness in the T or W matrix needs to be controlled [52].

There has been a large body of work on the computational issues of NMF. The traditional NMF criterion depends only on the product TW. Even with the requirement that T and W be non-negative, the solution is not always unique because there can be many factorizations that give the same product [61]. $X \approx TW = TAA^{-1}W$ where A is an invertible matrix. If there exists such invertible matrix A that TA and $A^{-1}W$ are nonnegative matrices, the solution of NMF is not unique. If TA and $A^{-1}W$ are nonnegative matrices only if A is a diagonal matrix or a permutation matrix, then the solution of NMF is unique up to scaling and permutation. Although the general theory on the uniqueness of the solution of NMF is unknown, there exist different approaches to obtain unique NMF solutions and most of them are based on the incorporation of additional constraints into the NMF model. One approach is to add a sparseness penalty to the T and W matrices [46]. Another popularly used constraint is the minimum volume.

The simplex volume determined by T is constrained to be the minimum among all possible NMF outcomes [80]. Requiring each column in T to be orthogonal has also been proved to achieve unique NMF results although the condition is restrictive [28]. In this thesis, instead of adding constrains to the original NMF model, we only consider the choice of NMF algorithm that have an effect on the results of the analysis. The original multiplicative algorithms of Lee and Seung [63] are reliable but slow. A number of authors have developed faster but less reliable methods, for example: Gonzalez and Zhang [37], Lin [67], Hoyer [45] and Shahnaz [97]. For a more thorough discussion of the algorithms available and their merits, see Berry and Browne [7]. We used the R package NMF by Renaud [33], which implements the algorithm of Lee and Seung [63].

Another challenge in applying NMF is to choose, k, the NMF rank. We develop a new method for this in Chapter 4. For the current chapter, we use a simple heuristic approach. Generally, the log-likelihood increases with k increasing. We can plot the log-likelihood values versus k to find the "elbow point" after which the log-likelihood increases more slowly. This means the increase in the NMF rank will not add as much in modeling the data. Thus we should choose the k value at the "elbow point". This is equivalent to choosing k such that the loglikelihood increase between $K+1$ and K is smaller than some preset threshold value. In cases where there is no such "elbow point", exploring multiple different k values by using an interactive data exploration tool, *SimplePlot* which is described later in this section, could help to find the k value based on which some meaningful data structure can be shown. While this is an ad-hoc method, we will develop a more accurate rank selection method with solid statistical inference in Chapter 4.

2.1.2 Supervised NMF

For a supervised learning problem, we have observations from different classes. Our objective is usually to find the differences between the structures from the different classes. We will approach this by separately identifying the subcommunities in each class first, and then combine them into a single matrix of subcommunities. Each sample now can be expressed as a mixture of all these subcommunities. For example, if we have data X from g classes,

$$X = \left(X^{(1)}, X^{(2)}, \cdots, X^{(g)}\right)$$

where $X^{(1)}, X^{(2)}, \cdots, X^{(g)}$ are g classes of observations. From $X^{(i)}$, we can calculate the non-negative type matrix $T^{(i)}$ and weight matrix $W^{(i)}$ $(i = 1, \cdots, g)$ by NMF, which uses the multiplicative update rule to minimize the distance $D(X\|TW) = \sum_{ij}(X_{ij} \log \frac{X_{ij}}{(TW)_{ij}} - X_{ij} + (TW)_{ij})$ [94]. The algorithm update each element in T and W by multiplying the current value by some factor that depends on the quality of the approximation of $X \approx TW$ in each iteration. To get the hidden structure of different classes in the whole data, we combine these type matrices together and denote this combined type matrix for the whole data as

$$T = (T^{(1)}, T^{(2)}, \cdots, T^{(g)})$$

It is straightforward that T is non-negative since each $T^{(i)}$ is non-negative and each column of T is normalized to have unit sum. For fixed T, to maximize the Poisson log-likelihood for the whole data X is equivalent to maximizing the Poisson log-likelihood for each sample, because the weight vectors in W associated with different samples are independent. Thus calculating the weight matrix W can be reduced to performing a non-negative Poisson regression of each sample in X on T.

Our purpose is to find the non-negative coefficients for a Poisson regression with identity link and without intercept, by maximizing the Poisson log-likelihood. We now focus on the regression of one sample $X_j = (X_{1j}, X_{2j}, \cdots, X_{pj})$ on T. The resulting coefficients $W_j = (w_{1j}, \cdots, w_{kj})$ thus will be either positive or 0, with 0 coefficients corresponding to the variables in T removed from this regression. We aim to find a list of positive coefficients with the corresponding variables, so that adding another variable to the list cannot improve the likelihood and still maintain the non-negative constraint. This is achieved through a backwards-forwards Poisson regression procedure as follows.

We start by recursively fitting a Poisson regression on T and removing the variables corresponding to the negative coefficients in $W_j = (w_{1j}, \cdots, w_{kj})$ until all the coefficients are positive. Using the remaining variables, we calculate the

log-likelihood value. Then we test each removed variable by adding it back with a small positive coefficient. If this increases the log-likelihood value, we add this variable back to the remaining variables and repeat the above steps, otherwise we remove this variable and test the next one.

The algorithm follows these steps:

1. Fit a Poisson regression with identity link but without intercept on T with the initial value of W_j set as the coefficients of linear least square regression of X_j on T. Eliminate those variables corresponding to negative coefficients.

2. If any variables were removed, go back to step 1 until all the coefficients are positive. In the end, the matrix consisting of remaining variables is T_j^+. Since X, T and W are all non-negative, the resulting T_j^+ cannot be empty unless X is a zero vector.

3. Calculate the log-likelihood for T_j^+.

$$L(T_j^+) = \sum_{i=1}^{p} \left(X_{ij} \log(T_j^+ W_j)_i - (T_j^+ W_j)_i \right),$$

   where $(T_j^+ W_j)_i$ denotes the ith element of the vector $T_j^+ W_j$.

4. Add one variable in the removed pool to T_j^+, denote the new feature matrix as $T_{j\ new}^+$ and calculate the log-likelihood again.

$$L(T_{j\ new}^+) = \sum_{i=1}^{p} \left(X_{ij} \log(T_{j\ new}^+ W_{j new})_i - (T_{j\ new}^+ W_{j new})_i \right),$$

   where $W_{j new} = (W_j(1-\varepsilon), \varepsilon)$, ε is a very small positive number close to 0. In our practice, we use 10^{-7} as the value of ε.

5. Compare $L(T_j^+)$ with $L(T_{j\ new}^+)$, if $L(T_j^+) < L(T_{j\ new}^+)$, use this new $T_{j\ new}^+$ composed of T_j^+ and the new variable to repeat steps 1 to 5. Otherwise remove this variable and try to add another variable in the removed pool to T_j^+ and repeat steps 4 to 5, until all removed variables have been tested.

In step 4, we add back one removed variable each time into the positive T matrix and calculate the new log-likelihood value. To decide if this variable should

be added back, we do not need to refit the Poisson regression when calculating the new log-likelihood value. As the old coefficient matrix is a local maximization for the log-likelihood function with the remaining variables, the derivative of the log-likelihood at that point should be 0 with respect to all remaining variables. When we add another variable with a small positive coefficient into the system, if we are near to the original maximum, the log-likelihood for the new point will either increase or decrease, depending whether the derivative with respect to the newly added variable is positive or negative. So if we want to see whether a variable could increase the log-likelihood, we can just add a very small weight ε for the new variable, then calculate the new log-likelihood with the new rescaled weight matrix. We need to rescale the W_j vector, so that $W'_{j_{new}}1 = X'_j1$, where $1 = (1, \cdots, 1)$. This is because we assume the data follow the Poisson distribution, so the sum of the observations X_j also follows a Poisson distribution with the mean given by the sum of the mean vector TW_j. As each column of T has unit sum, $W'_j1 = W'_jT'1$. Maximizing the Poisson loglikelihood is equivalent to maximizing the sum of a multinomial loglikelihood and a Poission loglikelihood. Denote $\frac{(TW)_{ij}}{(TW)'_j1} = P_i$, we have

$$\begin{aligned}\sum_{i=1}^{p}\left(X_{ij}\log(TW)_{ij} - (TW)_{ij}\right) &= \sum_{i=1}^{p}\left(X_{ij}\log P_i + X_{ij}\log((TW)'_j1) - (TW)_{ij}\right)\\ &= \sum_{i=1}^{p}X_{ij}\log P_i + \left(\sum_{i=1}^{p}X_{ij}\log(W'_j1) - W'_j1\right)\end{aligned}$$

The first term is the multinomial loglikelihood and the second term is the Poisson loglikelihood on the total count of the vector X_j. Only the first term is related to matrix T in the optimization. Rescaling vector W_j doesn't influence the estimates of P_i, thus by maximizing the second term only, we can get $W'_j1 = X'_j1$.

We compare this new log-likelihood value with the old one. If it decreases, the derivative is negative which means points with positive weight on the new variable will decrease the log-likelihood. Then the new variable should not be added. If the new one is larger than the old one, add this variable into the positive T matrix and do a Poisson regression on this new positive T matrix again and repeat the

above steps until no variable can be added. In this way, we can make sure that each time we decide to add a new variable to the positive T matrix, the likelihood becomes larger. This procedure keeps the log-likelihood function increasing under the constraints that all elements in W_j remain non-negative.

To see that the algorithm will converge, a key point is that our algorithm is only dealing with the discrete part of the optimization, and the Poisson regression takes care of the continuous optimization. Since we are optimizing over a finite number of possible sets of positive variables, convergence to at least a local optimum is guaranteed by the fact that each step increases the likelihood.

2.1.3 Method for choosing the NMF rank

The NMF rank for each class of observations should be chosen to best describe its own class but not to describe other classes or noise. For discrimination purposes, the NMF rank for each class should be chosen to best separate the classes in combination with the number(s) of types in other classes. The most direct way to choose NMF rank for all classes is to find the model mis-classification errors on the validation sets for each combination of the numbers of types for different classes. However the computation burden is heavy in such an effort. Thus we propose to choose the NMF rank for each class separately as below first and try the selected combinations of NMF ranks for different classes if the results are not clear-cut.

In order to choose the best NMF rank for the first class, we will look at the deviance statistics to see how well the chosen types will fit the first class better than other classes. (Deviance is a measure of fit between data and model, given by the difference in log-likelihood between the current model, and a saturated model. Smaller deviance corresponds to better fit). Since the types are chosen from the first class, to make the comparison objective, the deviance statistics need to be calculated on a test set of the first class. We obtain one deviance statistic for each data point in the test set. We use cross-validation, so that every data point is in one test set. The deviance statistics are not normally distributed, thus we will use the Wilcoxon Rank-Sum test [106] based on the deviance statistics to test how well the classes are separated. The idea is to rank the deviance statistics from the test data points. If there is no discrimination between the classes, then the ranks

should be distributed randomly between the classes. The Wilcoxon Rank-Sum test computes a statistic which measures how unevenly the ranks are distributed between the classes. This statistic is then standardised so that it (approximately) follows a standard normal distribution under the assumption that the ranks are randomly distributed between classes. We refer to this standardised statistic as a Z-value. We obtain one Z-value for each fold of the cross-validation. Our overall measure of difference is the sum of the Z-values for each fold, divided by $\sqrt{r}$, where r is the number of folds. (Dividing by $\sqrt{r}$ ensures that if the model is equally good at fitting the data from the two classes, then this overall measure follows a standard normal distribution.) We have one Z-value from each fold of the cross-validation, so by calculating the standard deviation of these Z-values, we are able to obtain a standard error for our overall statistic. For each class, we will try a sequence of values for the NMF rank and find the best value to discriminate this class from other classes.

We use a 2-class data case as an example to illustrate the ideas. We use an r-fold cross-validation on training data for both classes. In each cross-validation, we separate the training data into a training fold and a test fold. To choose the NMF rank for class 1, we apply the following steps to a range of values for k:

1. For each fixed value k, fit k types on the training folds from class 1 to get the type matrix T.

2. Fit the remaining test fold data from class 1 and one fold of data from class 2 on T.

3. Calculate the deviance for each fitting. (One deviance value for each data point in the test folds).

4. Use a Wilcoxon Rank-Sum test on these deviances to get one Z-value for each fold.

5. Sum the values of Z statistics from all different cross-validations and divide by $\sqrt{r}$. Let Z_{all} denote this statistic. This statistic should approximately follow a normal distribution with mean of zero and standard deviation of 1

under the null hypothesis that the distributions of deviance values from both classes are the same.

6. Choose the smallest k for which Z_{all} is within one standard deviation of the largest Z_{all}-value, where the standard deviation is calculated as the sample standard deviation of the Z-values from the different folds for each k.

Note that the purpose is to choose k such that the deviances from two classes are best separated, not a hypothesis test to test the equality of means. Thus the sample standard deviation of Z_{all} is calculated from the different folds in the last step, instead of using 1. By using r-fold cross-validation and combined Z-values, we can effectively increase the power of this test, which is particularly important when the number of observations is small.

When the classification problem is an easy one, there is a clear separation for the deviances resulting from the class for which we are selecting the NMF rank and that from other classes. The near complete separation often results in the almost equal Z-values from the different folds, thus the sample standard deviation of Z_{all} is small. When the classification problem is hard, the resulting Z-values from different folds tend to have larger variance. The rank selected in the easy case usually is small and clear cut, the rank selected in the harder case usually tends to be large. After we run the above procedure to select numbers of types for all classes, we will fix the NMF rank for the easy case and select the best matching NMF rank for the other class so that the misclassification error is minimized.

2.1.4 Prediction

For fixed $T = (T^{(1)}, T^{(2)}, \cdots, T^{(g)})$, we apply the non-negative Poisson regression algorithm on training data to calculate the training W and on test data to calculate the test W. After getting the W matrix, we have effectively reduced the dimension from p to k, in the sense that for the fixed T feature matrix, each observation is best approximated by the corresponding k vector in the W matrix. We can use an off-the-shelf supervised learning method to predict the class labels since $k < n$. Note that the sum of each column in the W matrix is the same as the sum of the corresponding column in the X matrix, which means sequencing depth

in the microbes data. When we perform a supervised learning, the transpose of the W matrix will be used as input for each observation. Geometrically this corresponds to projecting all the data into the space spanned by the vectors in the T matrix. The entries for different individuals on the same input vector of T are not comparable due to the different sequencing depth for the original data. We normalize the W matrix so that its column sum is 1 before performing a supervised learning method. This makes the entries in each row of W comparable and also makes it possible to show all the data in a plot. The normalization at this step is different from the normalization on the X matrix directly, because different sequencing depths result in heterogeneity in the original observations, and this has to be taken care of in the likelihood calculation and in the estimation of T and W.

We choose a suitable supervised learning method based on the graphical display of NMF results as described below. In the following sections, we most often perform a logistic regression on W. We choose logistic regression because our interactive exploration of the data suggests that a linear classifier is appropriate for this classification, and logistic regression is one of the simplest linear classification methods. The trained logistic regression model can then be used to do prediction on the test W.

2.1.5 Graphical display of NMF results

In both unsupervised and supervised NMF methods, we estimate the mean of each observation by a linear combination of k types. Thus if we fix the positions of the k types, the location of the mean of each observation can be fixed relative to the positions of the types. To properly display the NMF results and better facilitate the interactive exploration, a software package, *SimplePlot*, has been developed by supervisor Toby Kenney. It is available from Toby Kenney's website `www.mathstat.dal.ca/~tkenney/SimplePlot/`. Using *SimplePlot*, we can manually move the positions of the types (represented by crosses on the figure) around the plane and watch how the relative positions of the means of the observations change. We can get a snapshot of a satisfactory projection of the data when the data show the best separation. The advantage of using interactive software is

that it is easier to identify non-linear separation if that is more appropriate for a particular dataset.

2.2 Simulations

2.2.1 Simulation Based on NMF

The design for the simulation studies in section 2.2.1 and the NMF results were in my MSc. thesis. Random Forest and Support Vector Machine simulation results have been added during my PhD. study.

We perform simulations in this section to evaluate the performance of our proposed method with regard to the NMF rank selected and prediction accuracy. We use types estimated from the Qin data [88] to do the simulation. We simulate data according to our proposed model. The data follows a Poisson distribution with mean $(TW)_{ij}$. To generate these data, we first generate the mean TW.

The mean is a linear combination of different features (different columns of T). We fix T to be the features obtained by applying NMF to the two classes in the Qin dataset [88].

We generate the W matrix by generating each entry from a uniform distribution on $[0, 1]$ then normalizing the column vectors so that the column sums of W are equal to the column sums of the IBD data.

The product TW gives us the mean and we add four levels of noise to the product TW. The noise is normally distributed with mean 0 and four different standard deviations, to study the effects of different signal-noise ratios (SNR).

$$SNR = +\infty : sd_0 = 0$$

$$SNR = 4 : sd_1 = sd(T)/4$$

$$SNR = 2 : sd_2 = sd(T)/2$$

$$SNR = 1 : sd_3 = sd(T)$$

Here the $sd(T)$ is a vector of standard deviations for each row of T. This is a vector of length p (the number of genes or OTUs) which measures the variability for each gene or OTU across different features in T.

The column of TW plus the noise is the Poisson mean we use in the simulation. Each element of X is generated following an independent Poisson distribution with the mean given by the mean matrix described above.

We simulate data with NMF ranks equal to 2, 5, 10 for class 1 and 3, 6, 9 for class 2. So the number of different combinations is 9 in total. They are 2&3, 2&6, 2&9, 5&3, 5&6, 5&9, 10&3, 10&6, 10&9. Considering the different noise levels, we have 36 scenarios. For each scenario, we simulate 25 replicates. In each replicate, we simulate 200 observations for each class. Then we separate the data into two parts: the first 200 observations (100 from each class) as the training data and the other 200 as the test data.

We choose the NMF ranks from the training data using a 10-fold cross-validation. After the NMF rank is chosen, we perform a prediction on the test data using the trained logistic regression model on the training data based on the chosen NMF rank for each simulated data set.

The NMF, RF and SVM misclassification errors are shown in Table 2.1, Table 2.2 and Table 2.3 respectively for different noise levels. We find when the true numbers of types get larger, the NMF misclassification errors tend to increase but the RF misclassification errors tend to decrease. That may be because we have more accurately estimated the NMF rank in the cases when the true numbers of types are small. This can be expected as when the number of types increases, the data is more complicated and estimating the ranks becomes much challenging. But overall, the misclassification errors are quite small for all cases which means our supervised NMF method works well in prediction. NMF performs better in prediction than RF when rank value is small and better than SVM in all scenarios.

Table 2.4 summarizes the results of the NMF rank chosen. It shows that the algorithm tends to output slightly larger values than the true NMF rank in most scenarios, but the true numbers of types mostly are within one standard deviation of the mean of the chosen NMF rank. Note also, the NMF ranks are chosen only by performing the Wilcoxon Rank-Sum test as proposed earlier for each class, the results are not modified through optimizing the classification results based on combined types.

Table 2.4 also shows that in most replicates, when the noise level becomes

Table 2.1: NMF mean misclassification test errors for 25 data sets (the value before /) with the standard errors of the mean misclassification error (the value after /). The rows are the true NMF rank for class 2 and the columns are the true NMF rank for class 1.

SNR	class 2 \ class 1	2 types	5 types	10 types
$+\infty$	3 types	0.0002/0.0010	0.004/0.0066	0.0104/0.0126
4		0.0002/0.0010	0.0050/0.0085	0.0092/0.0102
2		0.0002/0.0010	0.0048/0.0071	0.0102/0.0104
1		0.0002/0.0010	0.0022/0.0041	0.0084/0.0079
$+\infty$	6 types	0.0012/0.0036	0.0054/0.0071	0.0082/0.0089
4		0.0012/0.0030	0.0052/0.0076	0.0154/0.0114
2		0.0010/0.0029	0.0056/0.0082	0.0058/0.0067
1		0.0014/0.0037	0.0058/0.0081	0.0126/0.0107
$+\infty$	9 types	0.0106/0.0132	0.0068/0.0084	0.0108/0.0126
4		0.0088/0.0102	0.0066/0.0100	0.0132/0.0100
2		0.0078/0.0101	0.0072/0.1011	0.0128/0.0110
1		0.0046/0.0058	0.0062/0.0092	0.0124/0.0089

higher, the difference between the mean and the true NMF rank will increase. Nevertheless, these results demonstrate that our method is quite effective in finding the appropriate NMF rank.

Further simulation results (not shown in this thesis) have shown that when we apply NMF with the true NMF rank on the simulated data, the features computed from the data can match very closely with the true features that were used to generate the data. Applying NMF with the wrong number of features can recover a space with the true features embedded in it. The study of consistency of the NMF method is not a trivial topic and deserves further research.

2.2.2 Simulation with outliers

We designed this simulation to measure how our method performs when the data contain outliers. We perform this simulation based on data generated in the last section. We use the generated data from scenarios 2&3 types, 5&6 types and 10&9 types, with $SNR = 1$. We generate outliers by mislabeling the class of observations in the training data. We run simulations with 5 percent, 10 percent and 20 percent of observations in the training data mislabeled. We used the same

Table 2.2: RF mean misclassification test errors for 25 data sets (the value before /) with the standard errors of the mean misclassification error (the value after /).

SNR	class 1 / class 2	2 types	5 types	10 types
$+\infty$	3 types	0.0024/0.0010	0.0012/0.0007	0.0012/0.0005
4		0.0014/0.0004	0.0002/0.0002	0.0012/0.0005
2		0.0016/0.0006	0.0016/0.0006	0.0006/0.0003
1		0.0018/0.0006	0.0008/0.0004	0.0010/0.0005
$+\infty$	6 types	0.0022/0.0007	0.0026/0.0009	0.0008/0.0005
4		0.0020/0.0007	0.0018/0.0008	0.0010/0.0006
2		0.0020/0.0008	0.0014/0.0005	0.0010/0.0005
1		0.0020/0.0008	0.0024/0.0008	0.0006/0.0004
$+\infty$	9 types	0.0028/0.0010	0.0014/0.0007	0.0010/0.0006
4		0.0022/0.0010	0.0014/0.0006	0.0014/0.0007
2		0.0022/0.0009	0.0010/0.0005	0.0008/0.0004
1		0.0022/0.0007	0.0008/0.0004	0.0010/0.0005

procedure as in the previous section to calculate the misclassification errors. The results in Table 2.5 show that while RF is more robust in this simulation, NMF still predicts fairly well when there are outliers in the data.

2.2.3 Simulation with zero inflated weight matrix

In the previous simulations, we generated the weight matrix of the Poisson mean from the uniform distribution. The sparsity of the generated datasets is around 24 percent, which is less than is typically observed in practice. We therefore use a Dirichlet distribution with all parameters 0.005 for the weights, in order to generate zero-inflated data. This results in a sparsity (proportion of zeros in the data) of around 39 percent. We follow the same steps from the first section of the simulations, to generate 36 scenarios and 25 replicates in each scenario. The misclassification errors and the associated standard errors of NMF, RF and SVM are shown in Table 2.6, Table 2.7 and Table 2.8. The results show that NMF and SVM are robust when the data become more sparse. RF performs worse in this simulation than in the original simulation.

Table 2.3: SVM mean misclassification test errors for 25 data sets (the value before /) with the standard errors for the mean misclassification errors (the value after /).

SNR	class 1 / class 2	2 types	5 types	10 types
$+\infty$	3 types	0.1462/0.0173	0.1300/0.0167	0.1276/0.0156
4		0.1356/0.0176	0.1784/0.0169	0.1616/0.0153
2		0.1356/0.0176	0.1784/0.0168	0.1460/0.0124
1		0.1356/0.0176	0.1850/0.0151	0.1460/0.0125
$+\infty$	6 types	0.1794/0.0180	0.1214/0.0146	0.1620/0.0154
4		0.1774/0.0191	0.1626/0.0154	0.1738/0.0164
2		0.1772/0.0191	0.1498/0.0147	0.1590/0.0172
1		0.1392/0.0173	0.1632/0.0154	0.1248/0.0151
$+\infty$	9 types	0.1344/0.0193	0.1656/0.0179	0.1740/0.0177
4		0.1210/0.0158	0.1558/0.0166	0.1846/0.0162
2		0.1210/0.0158	0.1552/0.0181	0.1824/0.0177
1		0.1210/0.0159	0.1652/0.0198	0.1794/0.0178

2.2.4 Simulation Based on Dynamic Ecology Models

The interpretability of NMF is based on the assumption that the microbial community can be interpreted as a mixture of subcommunities. In this section, we study the question of whether realistic community dynamics can give rise to this assumption. Current knowledge of the community dynamics of the microbiome is woefully inadequate; with a few available suggested models, none of which fit the data very well. In this section, we simulate community dynamics under a Holling type II model [43], given by

$$\frac{dM_i}{dt} = M_i \left(r_i(1 - c_i M_i) + \sum_{j \neq i} \frac{b_{ij} a_{ij} M_j}{1 + a_{ij} T_{Hij} M_j} \right) \tag{2.1}$$

Here, for OTU i, r_i is the intrinsic growth rate; c_i is the coefficient of negative intraspecific interaction, which is the inverse of the carrying capacity of this OTU in isolation; a_{ij} is attack rate; T_{Hij} is handling time; and b_{ij} is the interaction coefficient between OTUs. When $a_{ij} T_{Hij} M_j$ is very small, the 1 term dominates the denominator, so the derivative approximately follows generalised Lotka-Volterra type dynamics for these OTUs; when $a_{ij} T_{Hij} M_j$ is large such that it dominates the denominator of the fraction, then the term becomes approximately $\frac{b_{ij}}{T_{Hij}}$, and

Table 2.4: Simulation summary of the estimated numbers of types. For example, the first entry 2/3.4 means when the true NMF rank is 2 for class 1, 3 for class 2 and $SNR = +\infty$, the mean numbers of types our method chooses are 2 for class 1 and 3.4 for class 2.

SNR		2		5		10	
		mean	sd	mean	sd	mean	sd
$+\infty$		2.0/3.4	0.0/1.0	5.0/3.1	0.3/0.3	8.9/3.5	1.9/1.3
4	3	2.0/3.8	0.0/1.5	5.0/3.3	0.3/1.2	8.9/3.7	2.2/2.0
2		2.0/3.7	0.0/1.4	5.0/3.1	0.3/0.3	8.9/3.7	2.2/2.0
1		2.0/3.6	0.0/1.4	5.1/3.0	0.5/0.2	8.5/3.8	1.9/2.0
$+\infty$		2.0/6.4	0.0/0.8	5.0/6.4	0.5/0.7	9.8/6.6	1.2/1.2
4	6	2.0/6.4	0.0/0.6	5.2/6.6	0.8/0.9	9.5/6.4	1.3/0.6
2		2.0/6.4	0.0/0.6	5.2/6.6	0.5/0.8	9.4/6.3	1.5/0.5
1		2.0/6.3	0.0/0.6	5.0/6.6	0.5/0.8	9.4/6.2	1.1/0.5
$+\infty$		2.4/8.4	1.5/1.5	5.1/8.9	1.7/0.8	8.7/9.2	1.8/0.8
4	9	2.0/8.2	0.0/1.7	5.0/9.8	0.4/1.1	8.8/9.2	1.8/0.6
2		2.4/9.1	1.5/0.7	7.0/9.3	1.5/0.8	11.0/10.9	1.2/1.4
1		2.5/9.0	1.1/1.1	6.9/8.7	1.8/0.9	9.7/10.8	1.7/1.8

the influence of OTU j on OTU i is limited by this quantity.

The reason we choose the Holling Type II model, rather than the more commonly used generalised Lotka-Volterra dynamics is that the Holling model seems to have more capacity for overlapping communities to coexist without influencing one-another excessively, because the Holling type II model incorporates a limit on the effect of one OTU on another. This makes intuitive sense when the interaction consists of one OTU providing some metabolite to another OTU. We expect the

Table 2.5: Mean misclassification test errors (the value before /) and the associated standard errors (the value after /) for simulation with outliers.

outliers proportion \ NMF rank	method	2&3 types	5&6 types	10&9 types
5 percent	NMF	0.0150/0.0029	0.0396/0.0054	0.0582/0.0063
	RF	0.0042/0.0012	0.0036/0.0009	0.0028/0.0009
	SVM	0.1874/0.0102	0.1870/0.0117	0.2114/0.0145
10 percent	NMF	0.0266/0.0040	0.0564/0.0050	0.0890/0.0061
	RF	0.0098/0.0024	0.0094/0.0020	0.0080/0.0018
	SVM	0.2404/0.0101	0.2298/0.0120	0.2694/0.0126
20 percent	NMF	0.0750/0.0088	0.1424/0.0083	0.1610/0.0084
	RF	0.0338/0.0037	0.0264/0.0042	0.0308/0.0036
	SVM	0.3066/0.0110	0.3020/0.0147	0.3118/0.0158

Table 2.6: NMF mean misclassification test errors (the value before /) and the associated standard errors (the value after /) for simulation with zero-inflated weight matrix

SNR	class 1 / class 2	2 types	5 types	10 types
$+\infty$	3 types	0.0000/0.0000	0.0048/0.0016	0.0158/0.0023
4		0.0000/0.0000	0.0050/0.0016	0.0154/0.0023
2		0.0000/0.0000	0.0050/0.0016	0.0150/0.0024
1		0.0000/0.0000	0.0050/0.0016	0.0148/0.0021
$+\infty$	6 types	0.0016/0.0009	0.0062/0.0017	0.0148/0.0025
4		0.0016/0.0009	0.0058/0.0076	0.0144/0.0024
2		0.0014/0.0008	0.0066/0.0018	0.0148/0.0024
1		0.0018/0.0010	0.0062/0.0016	0.0144/0.0025
$+\infty$	9 types	0.0033/0.0013	0.0068/0.0017	0.0148/0.0024
4		0.0110/0.0018	0.0058/0.0015	0.0134/0.0022
2		0.0036/0.0013	0.0066/0.0015	0.0142/0.0026
1		0.0032/0.0011	0.0078/0.0024	0.0156/0.0023

Table 2.7: RF mean misclassification test errors (the value before /) and the associated standard errors (the value after /) for simulation with zero-inflated weight matrix

SNR	class 1 / class 2	2 types	5 types	10 types
$+\infty$	3 types	0.0072/0.0015	0.0062/0.0015	0.0046/0.0012
4		0.0054/0.0013	0.0026/0.0010	0.0042/0.0015
2		0.0048/0.0013	0.0058/0.0012	0.0044/0.0012
1		0.0068/0.0021	0.0056/0.0013	0.0050/0.0012
$+\infty$	6 types	0.0074/0.0013	0.0122/0.0021	0.0082/0.0018
4		0.0090/0.0017	0.0118/0.0014	0.0066/0.0013
2		0.0112/0.0021	0.0148/0.0024	0.0076/0.0018
1		0.0080/0.0015	0.0122/0.0022	0.0070/0.0013
$+\infty$	9 types	0.0102/0.0019	0.0150/0.0024	0.0118/0.0023
4		0.0110/0.0018	0.0168/0.0024	0.0104/0.0021
2		0.0104/0.0017	0.0150/0.0030	0.0122/0.0028
1		0.0100/0.0019	0.0126/0.0022	0.0140/0.0026

Table 2.8: SVM mean misclassification test errors (the value before /) and the associate standard errors (the value after /) for simulation with zero-inflated weight matrix

SNR	class 1 / class 2	2 types	5 types	10 types
$+\infty$	3 types	0.1224/0.0166	0.2022/0.0193	0.1622/0.0171
4		0.1130/0.0150	0.1776/0.0188	0.1884/0.0162
2		0.1324/0.0163	0.1690/0.0205	0.1948/0.0144
1		0.1222/0.0134	0.2040/0.0158	0.1896/0.0134
$+\infty$	6 types	0.1246/0.0146	0.1818/0.0179	0.1596/0.0144
4		0.1254/0.0141	0.2048/0.0112	0.2200/0.0125
2		0.1262/0.0149	0.2088/0.0129	0.2366/0.0093
1		0.1206/0.0147	0.2084/0.0120	0.2058/0.0128
$+\infty$	9 types	0.1744/0.0127	0.2274/0.0157	0.2230/0.0123
4		0.1750/0.0096	0.1472/0.0173	0.1794/0.0125
2		0.1858/0.0083	0.2130/0.0144	0.1882/0.0096
1		0.2070/0.0100	0.1664/0.0144	0.1742/0.0137

growth of an OTU to be limited by multiple metabolites, and when one metabolite is used up, increasing the supply of another metabolite would not be expected to have a significant increase on the growth rate. This limit on the effect allows overlapping subcommunities to mix in an approximately linear way. We anticipate that a detailed model based on flux balance equations could be developed which would both model community dynamics more accurately and follow the assumptions behind NMF more closely. However, developing new models for the dynamics of microbial ecology is beyond the scope of this thesis.

We use the fixed network structure shown in Figure 2.1 for the simulations. We can see that the network used is made up from three overlapping clusters ($M1$–$M10$, $M9$–$M18$ and $M17$–$M26$). The intuition is that for each cluster there is a metabolic subcommunity, representing the stable state of the system when restricted to that cluster, and that the overall community is made up as a mixture of these subcommunities. For each black link in the network in Figure 2.1, we simulate the species interaction coefficient b_{ij} as following a uniform distribution between 0 and 0.008. For the blue links in the network, we simulate b_{ij} from a uniform distribution between -0.002 and 0.008 and for the red ones we simulate b_{ij} from a uniform between -0.08 and 0. We set T_{Hij} around 10^{-5} by generating

$\frac{1}{T_{Hij}}$ from $10^5 \times \text{beta}(5, 1)$. This scale of T_{Hij} allows the Holling type II dynamics to take effect — if T_{Hij} is much larger, the effect of one OTU on another is limited, so the OTUs become almost independent, losing the subcommunity structure. If T_{Hij} is much smaller, then the interspecific interaction term is approximately linear, so we get gLV dynamics, which are less suited for overlapping clusters. We allow r_i and c_i to vary between samples in each dataset, with r_i simulated from a uniform distribution between 0 and 1, and $\frac{1}{c_i} - 1$ simulated from $99 \times \text{beta}(1, 2)$. The idea is that these parameters are related to the suitability of the environment for OTU i, so different samples would have different values. The other parameters are kept fixed for all samples, since these represent the inherent ability of these OTUs to interact, so should not be expected to vary greatly between environments. We simulate 10 values of the parameters b_{ij} for the given network. For each of these simulated values, we simulate one data set with 50 samples, one with 100 samples and one with 200 samples. To construct each sample, we simulate values of r_i and c_i for each OTU, and simulate the dynamics from Equation 2.1, using 1000000 iterations with a stepsize of 0.001.

For each dataset, we apply NMF with four types. We compare the fitted types with the known subcommunities, both visually and using a formal loss function.

Results

We also calculate the co-occurrence networks [6] of the simulated data and compare the results with NMF. The co-occurrence network is produced by calculating the correlation of each pair of nodes in the simulated data. A null distribution for each pair is generated by permuting the abundance of one of the pair and re-calculating the correlation. The resampling is performed 1000 times and the distribution is used to calculate p-values. The p-values are then corrected using Benjamini-Hochberg [3] to control the false discovery rate and p-values less than $p = 0.05$ were considered to be statistically significant edges in the network. [6] We use the same procedure described in [6] for comparison in this chapter. But considering the tests are dependent, a more appropriate way to control the false discovery rate is through Benjamini–Yekutieli [4].

Neither NMF nor co-occurrence networks are designed exactly to identify the

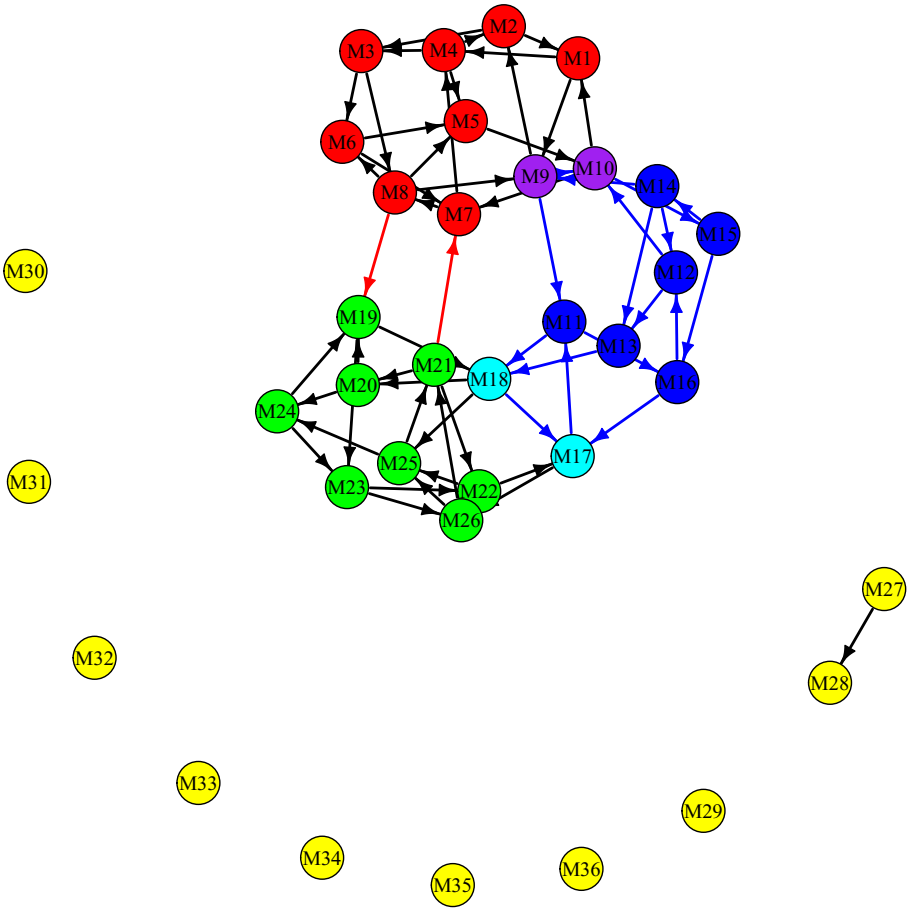


Figure 2.1: Network used for community dynamics simulations. The red nodes represent OTUs from cluster 1, blue nodes are OTUs from cluster 2, green nodes are from cluster 3 and yellow nodes are isolated OTUs not in any subcommunity. The purple and cyan nodes are overlapping OTUs of cluster 1 and cluster 2, or cluster 2 and cluster 3, respectively.

network structures or parameters of the Holling model. However, from the network structure in Figure 2.1, we see that the network can be reasonably decomposed as containing three large subcommunities (shown in red, blue and green in that figure, with nodes in multiple subcommunities coloured in mixed colours, purple and cyan). Both NMF and co-occurrence networks have some capacity to recover these subcommunities. For NMF, these subcommunities would be recovered as the most abundant OTUs in a type, while for co-occurrence networks, they would arise as connected components in the networks. We can attempt to compare the extent to which the two methods succeed at recovering these subcommunities. This extent is somewhat subjective. For an NMF type, we form clusters of OTUs as the OTUs with abundance above some threshold in that type. For co-occurrence networks, we form clusters as the connected components of the network at a certain significance level. We then choose unions of these clusters to recover the subcommunities used for simulation. To allow comparison, we have defined the following loss function for each true subcommunity to measure how far each such union is from the true subcommunity.

- For each OTU in the subcommunity, but not in the union of clusters, the loss is 1.
- For each OTU in the union of clusters, but not in the subcommunity, the loss is 1.
- For each additional cluster after the first in the union, the loss is 1.

For example, the loss for the red, green and blue subcommunities in Figure 2.2 are respectively 1, 1 and 5, and the loss for the red, green and blue subcommunities in Figure 2.3 are respectively 6, 7 and 9 (the blue community being best approximated by a singleton connected component). The abundance thresholds in the NMF type and the significance levels in the co-occurrence networks are chosen to minimize the total loss for each subcommunity. We allow different significance levels for different connected components here. Note that the example calculation above was meant to demonstrate how the loss function is calculated for a given set of clusters, based on the single figure, not on clusters with different p-values, so the values calculated may not be the actual loss function for that dataset.

Table 2.9: Mean/standard deviation of the optimum loss scores for each true subcommunity under two different methods, with 10 simulations for each sample size

sample size	cluster	$M1-M10$	$M9-M18$	$M17-M26$
n=50	NMF	3/1.8	7.2/2	2.4/1.9
	Co-occurrence network	5.4/2.4	7.5/0.8	3.5/1.6
n=100	NMF	2.7/1.3	6.8/2.2	2.1/1.4
	Co-occurrence network	6.7/2.2	6/1.6	4.5/1.7
n=200	NMF	2.7/0.9	6.1/2	1.5/1.2
	Co-occurrence network	6.9/1.7	6.6/2.1	6.3/1.3

Table 2.9 shows that NMF most often is able to recover the subcommunities used to simulate the data, especially when sample size is large. Note that the blue subcommunity (M9–M18) has weaker interactions between OTUs, so is less clearly a subcommunity, and is therefore not identified as well as the others. Since the loss function is somewhat ad-hoc, Figure 2.2 and Figure 2.3 show typical examples of the recovered types from one simulation with 200 data points, to allow more direct comparisons visually. As we can see, NMF has done a better job in recovering the subcommunities. It is also worth noting that co-occurrence networks tend to create many small clusters, which gives the method an advantage over NMF for the above defined loss function, particularly for subcommunities which are not identified well.

The fact that NMF is able to recover the true subcommunities in the simulated data does not necessarily mean that the subcommunities found by NMF on real data are genuine subcommunities, because the dynamics of the real microbial community could be different from those in this simulation. We have however shown that vaguely realistic models of microbial community dynamics can produce subcommunity structures similar to those modeled by NMF. Given that NMF is able to uncover these structures, we have better justification to support that the true community dynamics might also be well represented in terms of the subcommunities identified by NMF, and that these subcommunities have meaningful biological structure.

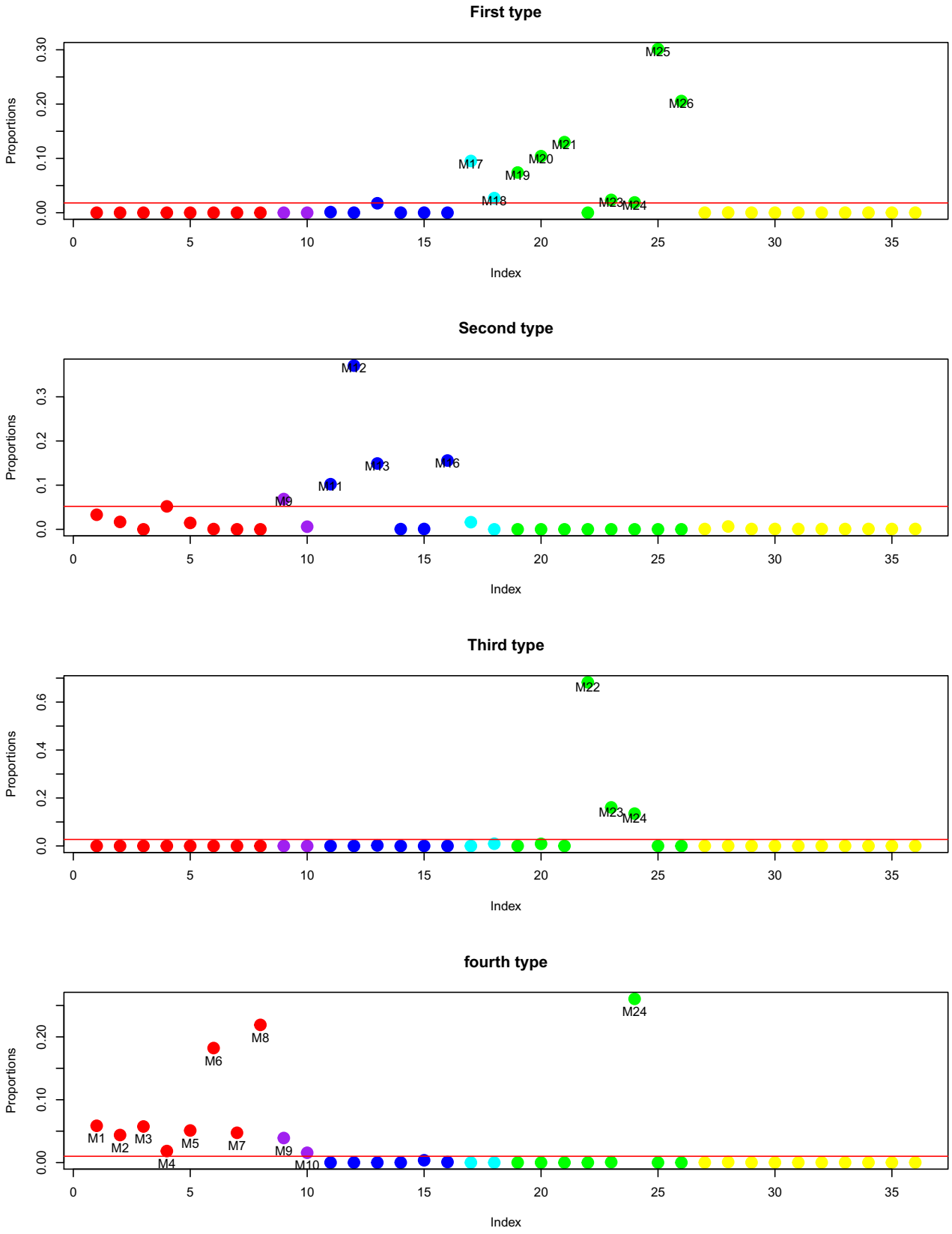


Figure 2.2: NMF features extracted from data simulated under a Holling type II model.

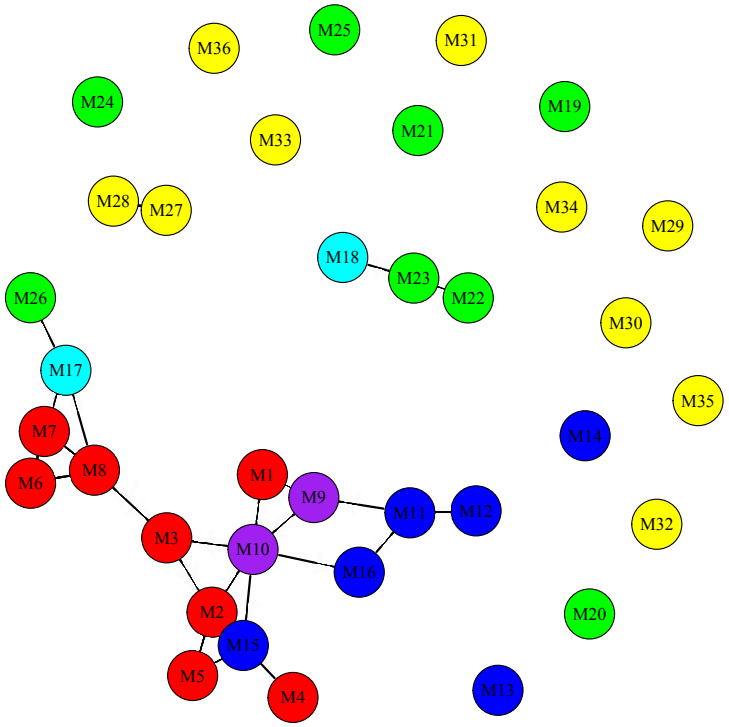


Figure 2.3: Co-occurrence network calculated from data simulated under a Holling type II model. OTUs are colored the same as in Figure 2.1. Two OTUs are linked by an edge if they are significantly correlated. The sign of the correlation is not considered in this plot.

2.2.5 Simulation for the performance of NMF as a clustering method

We construct the simulation following the method in McMurdie and Holmes *simulation A* [76]. The variables from real microbial data *ocean* and *feces* of the *GlobalPatterns* dataset are used to get two basic sets of multinomial probabilites. We then produce new multinomial probabilites for two classes as linear mixtures of the original two sets. The ratio of these mixtures is determined by the parameter *effect size*, $s_e \geqslant 1$. One class mixes the basic sets in the ratio $1 : s_e$, the other mixes them in the ratio $s_e : 1$. When $s_e = 1$, the classes are identical, so we expect no separation. As s_e increases, the difference between the classes becomes larger, so the clustering problem becomes easier. In the simulation, the sequencing depth for each sample is fixed to be 10000 and we simulate 200 samples for each class with effect size set to 1.01, 1.05, 1.1, 1.5 and 2. For each value of the effect size, we simulate 30 replicates.

For each simulated data set, we calculate 2 types NMF weight matrix on the original count data and then calculate the Euclidean distance between samples based on NMF weight matrix. For comparison, we also calculate Bray-Curtis dissimilarity, Euclidean distance, weighted UniFrac and rarefied Unweighted UniFrac on proportional data. We perform clustering analysis method Partitioning Around Medoids (PAM) with number of clusters fixed as two to measure the performance of these methods. The mis-clustering errors are shown in Table 2.10.

Table 2.10: Mean/standard error of the mis-clustering errors from the 30 replicates.

method \ effect size	1.01	1.05	1.1	1.3	1.5
NMF	0.4732/0.0042	0.134/0.0104	0.0003/0.0002	0/0	0/0
weighted UniFrac	0.469/0.0035	0.0195/0.0012	0/0	0/0	0/0
unweighted UniFrac	0.4827/0.0034	0.4798/0.0025	0.4887/0.0077	0.4916/0.0082	0.2823/0.0203
Bray-Curtis	0.4683/0.0033	0.2411/0.0128	0.0089/0.0020	0/0	0/0
Euclidean	0.4596/0.0053	0.1607/0.0084	0.0053/0.0010	0/0	0/0

From the table, we see that NMF performs generally better than other methods except the weighted UniFrac. But the NMF use the OTU table only and the rank for NMF is set to be 2 without tuning. This shows that the dimension reduction by NMF could help to filter out the noise and retain the major dissimilarity signals of the data.

2.3 Real data results and discussion

2.3.1 The mammal data

The unsupervised and supervised NMF results are calculated in my master thesis. The comparisons and the interpretations are completed during my PhD program.

The mammal dataset [81] contains gut metagenomes extracted from $n = 39$ mammals. The metagenomes include 1239 different types of genes (categorized by EC number). The mammals can be classified into four types: Carnivore, Foregut fermenting Herbivore, Hindgut fermenting Herbivore and Omnivore. There are 21 herbivores, 11 omnivores and 7 carnivores.

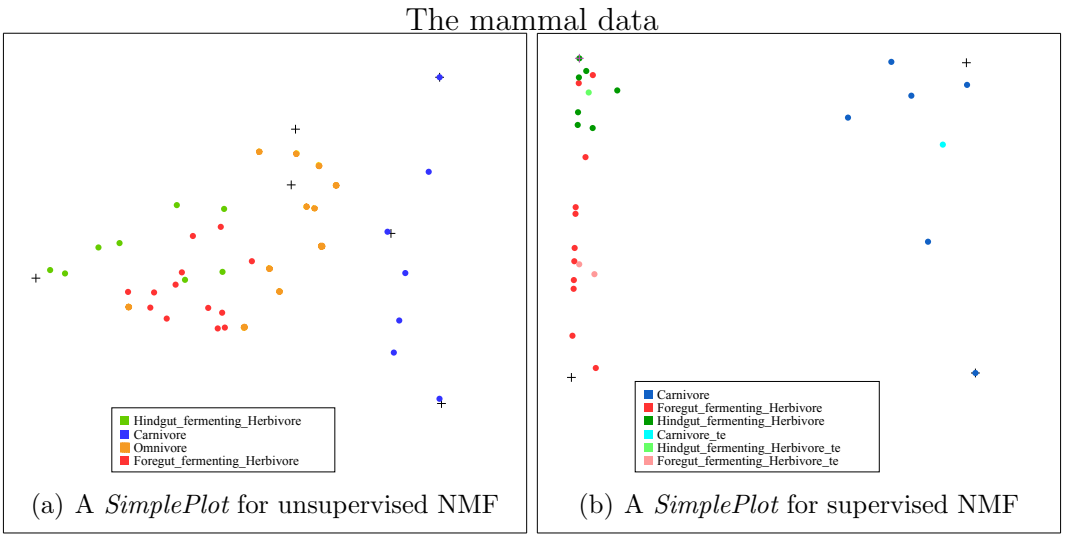
Unsupervised NMF results for mammal data

We calculate the log-likelihood for a range of k values and then observe how the log-likelihood changes with the k values. We choose the NMF rank for the mammal data as nine and apply unsupervised NMF on the data. A snapshot of the projected data on a plane is shown on the left panel of Figure 2.4. From the plot, we can see that the carnivores can be totally separated from others and the other three types are mostly separated with a few overlapping points. The dimension of the data is reduced from 1239 to 9 in this analysis.

Supervised NMF results for mammal data

In order to find the most important discriminant features between herbivores and carnivores, we apply supervised NMF only on the carnivores and herbivores from the mammal dataset [81]. So the data we use here contain metagenomic sequencing of fecal samples from 28 mammals: 7 carnivores and 21 herbivores. As the number of observations is small, we perform a 7-fold cross-validation on the whole data. Each time we use 6-folds as training data and the remaining observations as test data.

We find 2 types suitably describe both classes. Then we calculate 2 types on each class using the training data and combine them to get the type matrix T. Fixing the type matrix, we obtain the weight matrices for the training cases and test cases by the non-negative Poisson regression method. We fit a logistic



(a) A *SimplePlot* for unsupervised NMF

(b) A *SimplePlot* for supervised NMF

Figure 2.4: Left: Unsupervised NMF can totally separate the carnivores (blue) from the other three types of animals. The Foregut-fermenting Herbivores (red), the Hindgut-fermenting Herbivores (green) and Omnivores (orange) are largely separated with a few mixed. Right: Supervised NMF for separation of the carnivores from the herbivores. Both training carnivores (dark blue) and testing carnivores (light blue) are easily separated from the herbivores. The model was not trained to separate two types of herbivores, but a good degree of separation is shown for the Foregut-fermenting Herbivores (dark red for training and light red for testing cases) and the Hindgut-fermenting Herbivores (dark green for training and light green for testing cases).

regression using the training data weight matrices and perform a prediction on the test data.

The projections of both training and test data in one fold of the 7-fold cross-validation, relative to the positions of 4 types calculated from the training data, are plotted in the right panel of Figure 2.4. It shows that both training carnivores and test carnivores could be well separated from herbivores. Also from the plot, we can see that although we did not supervise the distinction between the two types of herbivores, there is some reasonable degree of separation between these two classes.

Both the training and test errors are 0 in each fold of the 7-fold cross-validation data split. The misclassification errors are all 0 meaning our algorithm could separate the two classes of mammals perfectly. The huge number of variables in the original data could be reduced to 4 features (2 for each class), which means the classes of mammals can be easily determined by four features.

To compare the supervised NMF with support vector machine and random forest, we choose the best tuning parameters for SVM by the same 7-fold cross-validation as in supervised NMF. The best cost value for all kernels is 1. The best gamma value for polynomial kernel is 0.01, for sigmoid kernel 0.001 and for radial basis kernel 0.1. We also compare with Random Forest with the sparse variables removed. (We remove 50% variables with lower abundance in all samples.) The mean and standard deviation of misclassification errors for models with these best tuning parameters on different folds are summarized in Table 2.11. The table shows that supervised NMF is among the methods which perform perfectly on the mammal data.

Interpretation of the features in the Mammal Data

We map the features extracted separately from herbivores and carnivores to the Metabolic pathways in KEGG by allocating abundant reactions in each feature to their corresponding pathways. We find that most of the features from herbivores and carnivores involve the same metabolic pathways except that herbivores have more reactions in the biosynthesis of Macrolides pathway, shown in Figure 2.5. The most significant difference is found in one of the features of Herbivores, which

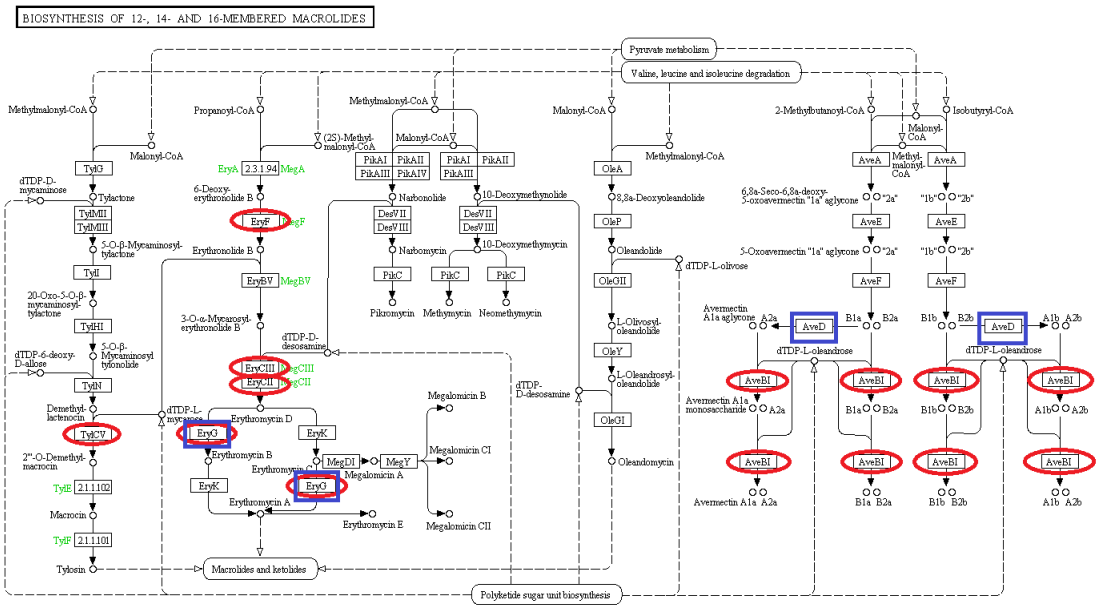


Figure 2.5: Biosynthesis of 12-,14- and 16-membered Macrolides. Reactions in red ellipses are those appearing in Herbivores' features. Reactions in blue rectangles are those appearing in Carnivores' features.

corresponds to the feature (cross) in the upper left corner of the right panel of Figure 2.4. (This feature has been highlighted in purple on this plot). Macrolides are a group of drugs belonging to the polyketide class of natural products. Macrolides are not to be used on non-ruminant herbivores, they rapidly produce a reaction causing fatal digestive disturbance [34]. That explains the results that 8 out of 10 herbivores which have heighest weight on this feature are non-ruminants. These correspond to the 8 hindgut-fermenting herbivores (green) in Figure 2.4(b). This shows that the inferred differences in the microbial communities of mammals relate well to the known different phenotypes for different mammals.

2.3.2 The moving picture data

The unsupervised and supervised NMF results are calculated in my master thesis. The comparisons and the interpretations are completed during my PhD program.

The moving picture data [14] is the most detailed investigation of temporal microbiome variation to date. It consists of a long-term sampling series from two human individuals at four body sites: gut, tongue, right and left palm. Person 2

was measured for a longer time than Person 1 (336–373 samples from each body site for Person 2 over a period of 443 days, compared to 131–135 samples from each site for Person 1 over a period of 186 days). The total number of variables (different OTUs) across all samples was more than 15000. After removing all 0's, the total number of different OTUs for the gut data is around 3000, for the tongue data is around 2000, for the left palm data and right palm data are around 13000. In spite of this extensive sampling, no temporal core microbiome was detected, with only a small subset of OTUs reoccurring at the same body site across all time points [14].

Unsupervised NMF results for gut data in the moving picture data

First we apply NMF to the gut data. The gut data consists of 131 observations from person 1 and 336 observations from person 2. We find the NMF rank is 6 based on the plot of log-likelihood values versus NMF rank values. And we see that the data from two individuals can be well separated — see the left panel of Figure 2.6. It can be seen 4 types seemed to be used to mainly describe individual 2 and 2 types are mainly related to individual 1. It also shows that the observations for individual 2 are separated into 2 groups, the reason for which will become clear later in this chapter.

Supervised NMF results for gut data in the moving picture data

As the gut data is time based, we choose the first 70 time points' observations out of 131 observations of person 1 and the first 170 time points out of 336 observations of person 2 as training data. If the system changes slowly, we might expect samples from the same individual separated by only a short time might be more closely related. By choosing this separation into training and test data, we minimize the correlation between training and test data, ensuring that we only test the method's ability to pick up long-term microbial signatures of each individual. A 10-fold cross validation with the training data split into 10 folds sequentially over time is applied for choosing the NMF rank and we find 2 types for each person is the best according to our method. This is an easy classification problem: based on two types for person 1, all deviance values for person 2 are much larger than

NMF and Unifrac results on the moving picture data

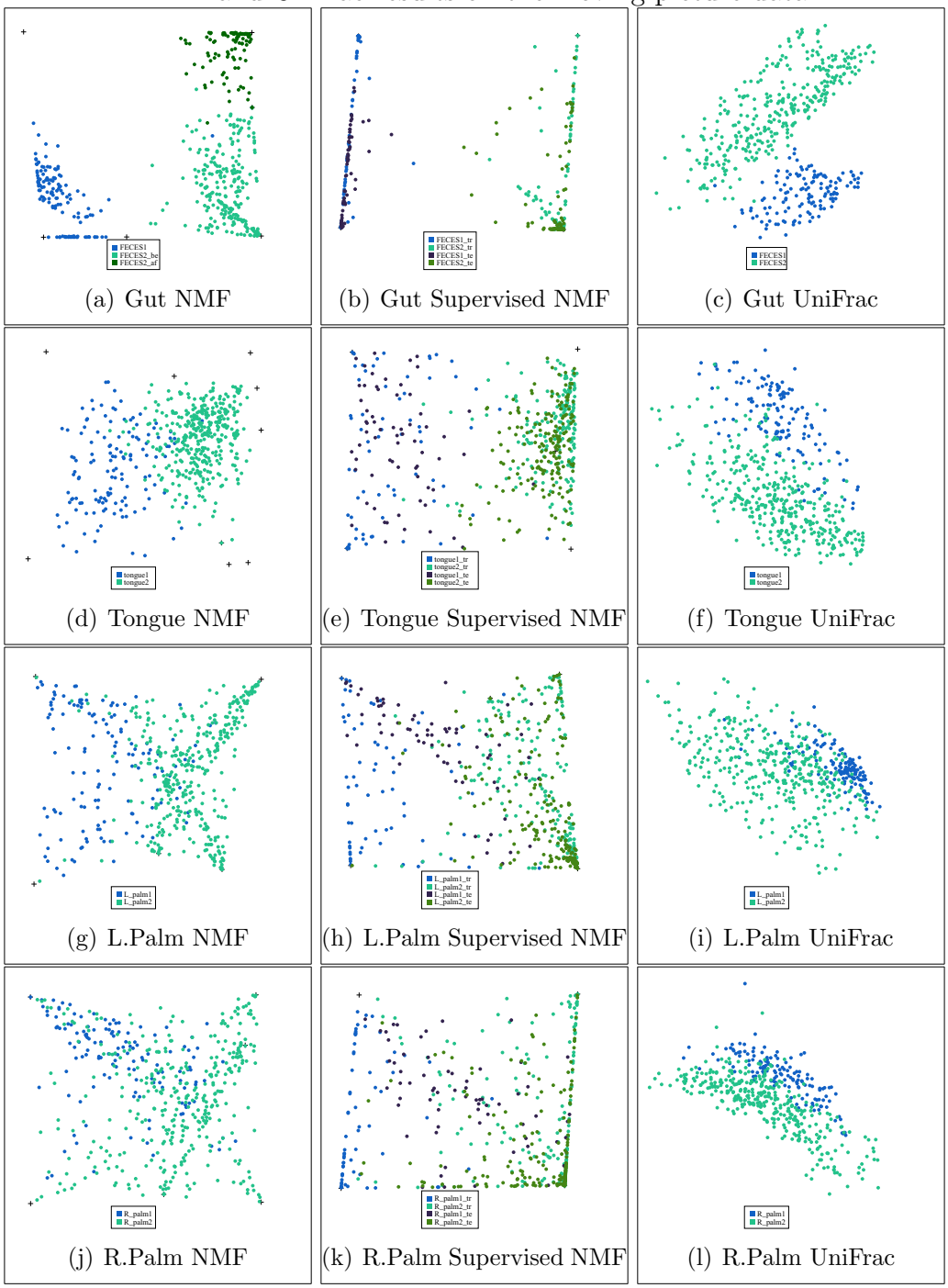


Figure 2.6: Top row shows results from the gut dataset (with 6 types/coordinates used for the unsupervised methods); second row shows results from the tongue dataset (with 9 types used for the unsupervised methods); third row shows results from the left palm (with 6 types/coordinates); fourth row shows results from the right palm (with 6 types/coordinates). Blue points are from person 1, green points are from person 2. **Left**: unsupervised NMF; **Middle**: supervised NMF on both training and testing data — darker blue and green points are testing data; **Right**: UniFrac.

the deviance values from person 1. A total separation is almost achieved for each fold of the cross-validation for any value of k ($k \geq 2$). This is the same based on 2 types for person 2. Thus we choose 2 types for each person. Then we fit a logistic regression model on the training W matrix, and perform a prediction on the test data. The results are shown in the right panel of Figure 2.6 and the misclassification error is 0 for the test data.

We see that both training and test data are almost perfectly separated between the two individuals which means the distinguishing features of the gut data are included in a matrix consisting of 4 features. These 4 features contain sufficient information for classification and will be examined in detail in the interpretation section together with the features computed from the tongue data from these two individuals, because some interesting connections between the tongue data features and gut data features can be detected within individual 2.

Unsupervised NMF results for tongue data in the moving picture data

Next we apply NMF to the tongue data. For the tongue data, there are 135 observations from person 1 and 373 observations from person 2.

It's not obvious what the appropriate value for the NMF rank should be by looking the plot of log-likelihood versus number of types. We try NMF on 9 types and 10 types. Neither achieves good separation between samples from the two individuals. A *SimplePlot* for unsupervised NMF for 9 types is shown Figure 2.6(d) as an example. Here we see that samples from the two individuals are somewhat separated, but there is a lot of mixing: we cannot achieve a great classification from these features.

Although standard NMF works for the animal dataset and the gut dataset, it does not perform as well on the tongue dataset. The reason is that with unsupervised methods, the signal that is identified is not always the signal we are interested in. Using supervised NMF, we will be able to identify the different features for different classes. This allows us to more easily distinguish samples from different classes.

Supervised NMF results for tongue data in the moving picture data

For the tongue data, as above, we choose the first 70 time points' observations out of 135 observations of person 1 and the first 190 time points out of 373 observations of person 2 as training data. The remaining data are test data. We split the training data over time and perform a 10-fold cross-validation on the training data to find the NMF rank for both individuals. Our method shows that 2 types are appropriate for Person 1, but is not so clear for Person 2 (possibly suggesting 9 types). Fixing 2 types for Person 1, and comparing cross-validated error, we choose 3 types for Person 2. This results in a test error of 0.04.

For illustration purposes, we show the *SimplePlot* of both training and test data based on 2 types for person 1 and 3 types for person 2 in Figure 2.6(f). Through these 5 features, most of the observations in tongue data could be correctly classified according to which individual they come from.

Unsupervised NMF results for left palm data in the moving picture data

For the left palm data, there are 134 observations from person 1 and 365 observations from person 2. We try NMF for several different types on the left palm data. None of them achieve good separations between samples from the two individuals. A *SimplePlot* for unsupervised NMF for 6 types is shown in Figure 2.6(g) as an example. Here we see that samples from the two individuals are somewhat separated with a considerable amount of mixing.

Standard NMF does not perform well on the left palm dataset. Using supervised NMF allows us to more easily distinguish samples from different classes.

Supervised NMF results for left palm data in the moving picture data

For the left palm data, we choose the first 67 time points' observations out of 134 observations of person 1 and the first 183 time points out of 365 observations of person 2 as training data. The remaining data are test data. Using the same procedure as for the tongue data, we find 2 types for person 1 and 3 types for person 2 can best separate the two individuals.

We show the *SimplePlot* of both training and test data based on 2 types for person 1 and 3 types for person 2 in Figure 2.6(h). Most of the observations in left palm data could be correctly classified with a test error of 0.092.

Unsupervised NMF results for right palm data in the moving picture data

For the right palm data, there are 134 observations from person 1 and 359 observations from person 2. Similar to the results for the left palm, there is not a good seperation between samples from the two individuals. A *SimplePlot* for unsupervised NMF for 6 types is shown in Figure 2.6(j) as an example. Here we see that most samples from the two individuals cannot be seperated using these features.

Supervised NMF results for right palm data in the moving picture data

For supervised NMF on right palm data, we choose the first 67 time points from person 1 and the first 180 time points from person 2 as training data. The remaining data are test data. We find 2 types for each person can best separate the two individuals.

We show the *SimplePlot* of both training and test data based on 2 types for each person in Figure 2.6(k). Most of the observations in the right palm data could be correctly classified according to which individual they come from with a test error of 0.179.

Comparisons with other methods

These datasets have also been extensively analysed by BioMiCo [95]. To enable comparison with their results, we reran our analysis with individual months as training data. (Rerunning BioMiCo with our splits into training and test data is infeasible due to its excessive running time).

We train supervised NMF on different months and predict the identity of the two individuals for all other months. The NMF rank used for each dataset is the same as we mentioned above. Even though a smaller number of samples are used to train the model, we still get very high-classification accuracy. The accuracy is

between 98.1% and 99.8% when using the gut dataset and 85.4% and 92.9% when using the tongue dataset. This is almost the same as BioMiCo's accuracy, between 98.6% and 99.3% for the gut dataset and between 85% and 93% for the tongue dataset. However, we also get very high accuracy when using the palm data, between 88.9% and 93% when using left palm dataset and between 77.8% and 83% when using right palm dataset. This is significantly higher than BioMiCo's results (40% to 75%). Palm data are more challenging because human palms are exposed to the external environment. The comparison with BioMiCo concludes that the supervised NMF is not only efficient in terms of computation, but also better at finding discriminant features of individuals even with very noisy data.

We also compared supervised NMF with support vector machine, random forest and random forest with sparse variables removed on this moving picture data. We split each body part's data in the same way as that in supervised NMF. A 10-fold cross-validation is applied to the training part to calculate the best tuning parameters. Models with the best tuning parameters then are trained on the whole training data and used to predict the test data. The results are summarized in Table 2.11. The comparison for moving picture data shows that supervised NMF gives comparable or better classification results than other methods except for the left and right palm dataset. For the left palm dataset, the random forest with sparse variables removed performs a little better than NMF and for the right palm dataset, random forest has smaller misclassification error.

UniFrac is a widely used unsupervised method. To compare the separation of two individuals, we project the samples on principle coordinates of the unweighted UniFrac distance matrix (based on rarefied samples) in the right-hand column of Figure 2.6 with the numbers of the principle coordinates equal to the numbers of types we have used for each case, presented using *SimplePlot*. We can see a clear separation of the two individuals from the gut dataset. Plots of tongue data and palm data show separations to some degree, but not as clear as in our unsupervised NMF plots (left panels in Figure 2.6). This shows NMF is an alternative and possibly more useful data exploratory method for such data. In addition, NMF has a natural interpretation in terms of mixtures of communities, but the results from UniFrac are hard to interpret, as they cannot show what causes the grouping

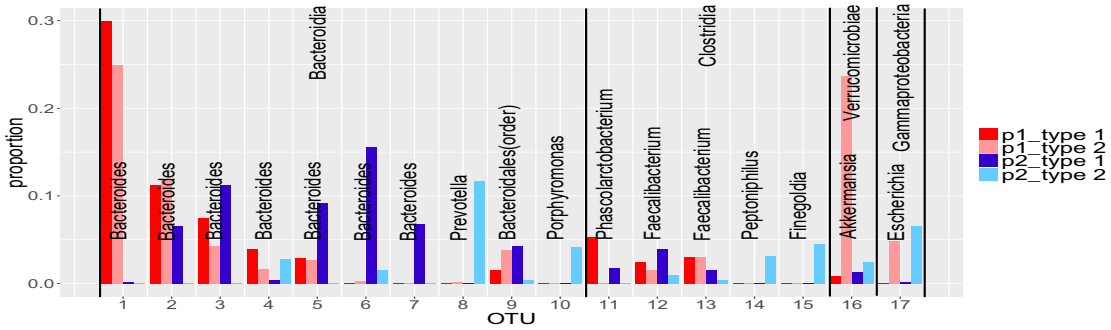
effects or where the differences in microbime composition lie.

Interpretation of the results

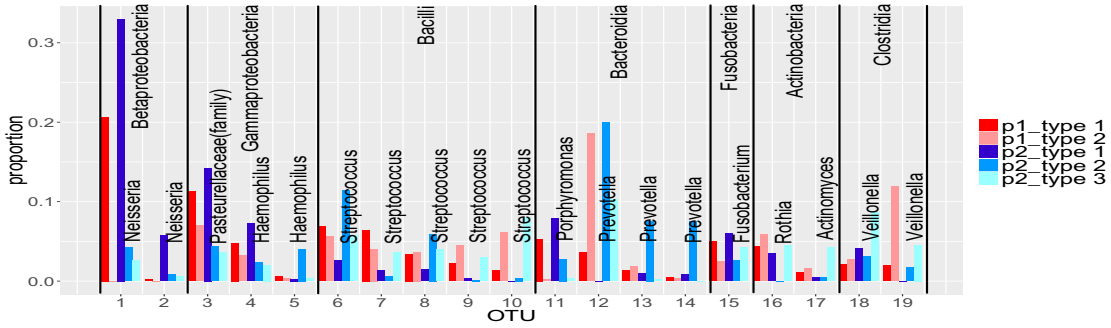
To examine the main aspects of the features identified, we plot the relative abundance of OTUs for different features in Figure 2.7. The feature vectors are of the same dimension as the original observations. A natural side effect of NMF is that the resulting feature vectors are usually sparse. The feature vectors consist of non-negative elements with each vector sum equal to 1. The non-zero values can be interpreted as the percentages of the OTU composition in a particular feature. To get a better illustration, we use a cut-off of 3% for each feature vector in Figure 2.7. That is, only those OTUs with above 3% composition in at least one feature are included in the plot.

Figure 2.7(a) shows the main OTUs for the gut data. We find only 17 out of more than three thousand OTUs are larger than the cut-off of 3%. Among these major OTUs, the two features within each individual bear some similarities. But the features between two different individuals are quite different. This is reflected by the fact that several of the most common OTUs in individual 1's features are not present in individual 2's features and vice versa. Since each individual's data can be best represented by his/her own two features and their two features are largely different, this partially explains why the classification of two individuals based on the gut data is an easy problem.

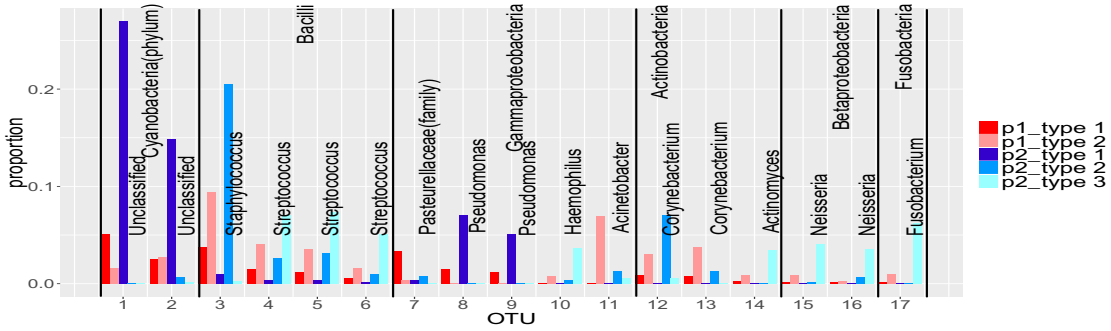
Figure 2.7(b) shows the main OTUs for the tongue data. There are only around 20 OTUs in tongue features above the cut-off of 3%. Again the type matrix of the tongue data is highly sparse. Unlike the features of the gut data, the features of the tongue data for these two individuals are more similar. By looking at the compositions of the most dominant OTUs in each feature, we can easily see similarities between person 1's type 1 and person 2's type 1. Also person 1's type 2 is similar to person 2's type 2 for OTUs in the classes Fusobacteria and Gammaproteobacteria and similar with person 2's type 3 for OTUs in the class Bacilli. This suggests that there are similar variation patterns between the two individuals, with the same groups of OTUs increasing or decreasing together. Naturally the classification for the tongue data is a harder problem.



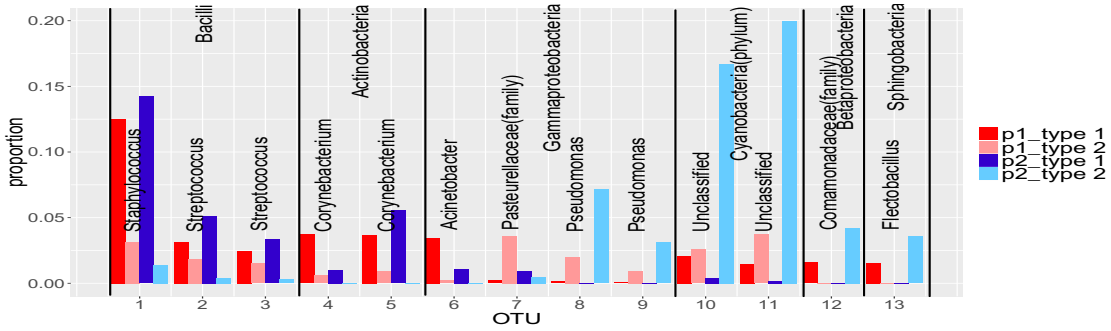
(a) Outstanding OTUs in features of gut data



(b) Outstanding OTUs in features of tongue data



(c) Outstanding OTUs in features of left palm data



(d) Outstanding OTUs in features of right palm data

Figure 2.7: Outstanding OTUs in features of moving picture data: The light and dark red bars are two features from Person 1 and the blue bars are features from Person 2. The OTUs from the same class are in the same block which is labeled by their class name and the bars are labeled by the genus of the OTUs. The two unlabeled bars in left palm data are the same OTUs with these unlabeled bars in the right palm plots. They are two different unclassified classes in Cyanobacteria phylum.

Figure 2.7(c) shows the main OTUs for the left palm data. 17 out of more than twelve thousand OTUs in the left palm features are larger than the cut-off of 3%. Among these major OTUs, the two features within individual 1 have OTUs present and absent together with some variations in their values. The features within individual 2 show a different pattern with each OTU mainly represented by one of the three features. Left palm features within each individual are quite different because the palm's microbial environment is more variable. Features between individuals are also quite different for most of these major OTUs. Several OTUs in individual 2's features are not present in individual 1's features. This may explain why the left palm data can achieve high classification accuracy but lower than the gut data.

Figure 2.7(d) shows the main OTUs for the right palm data. There are only 13 OTUs in right palm features above the cut-off of 3%. The patterns of features within each individual are similar to their left palm data. But features between individuals are more similar except for differences in the two unlabeled OTUs. This explains the difficulty in separating two individuals from the right palm data. We also find major OTUs in the right palm features are nearly all present in the left palm features. We do not find the same situation in gut and tongue features. This may be because an individual's left and right hands are usually exposed to the same environment. It could also be caused by contact between the two hands.

In many of the examples NMF can act like a variable selection method — identifying individual reactions or OTUs which show different abundances in the two groups of samples. However, in the moving picture tongue dataset, we do not obtain such good classification by looking at individual OTUs. Instead, we look more deeply at the community structure identified by NMF. By examining community-level differences we were able to classify the individuals with a very high degree of accuracy. We now look in more detail at the communities involved, in an attempt to understand why unsupervised NMF was less effective in this case, and why supervised NMF was able to resolve this problem. This also demonstrates more of the range of interpretability offered by NMF. In addition to highlighting individual OTUs or reactions that differ between the two classes, it is able to isolate bacterial subcommunities from which the microbiome is built up, and offer

insights into the different structures of these communities.

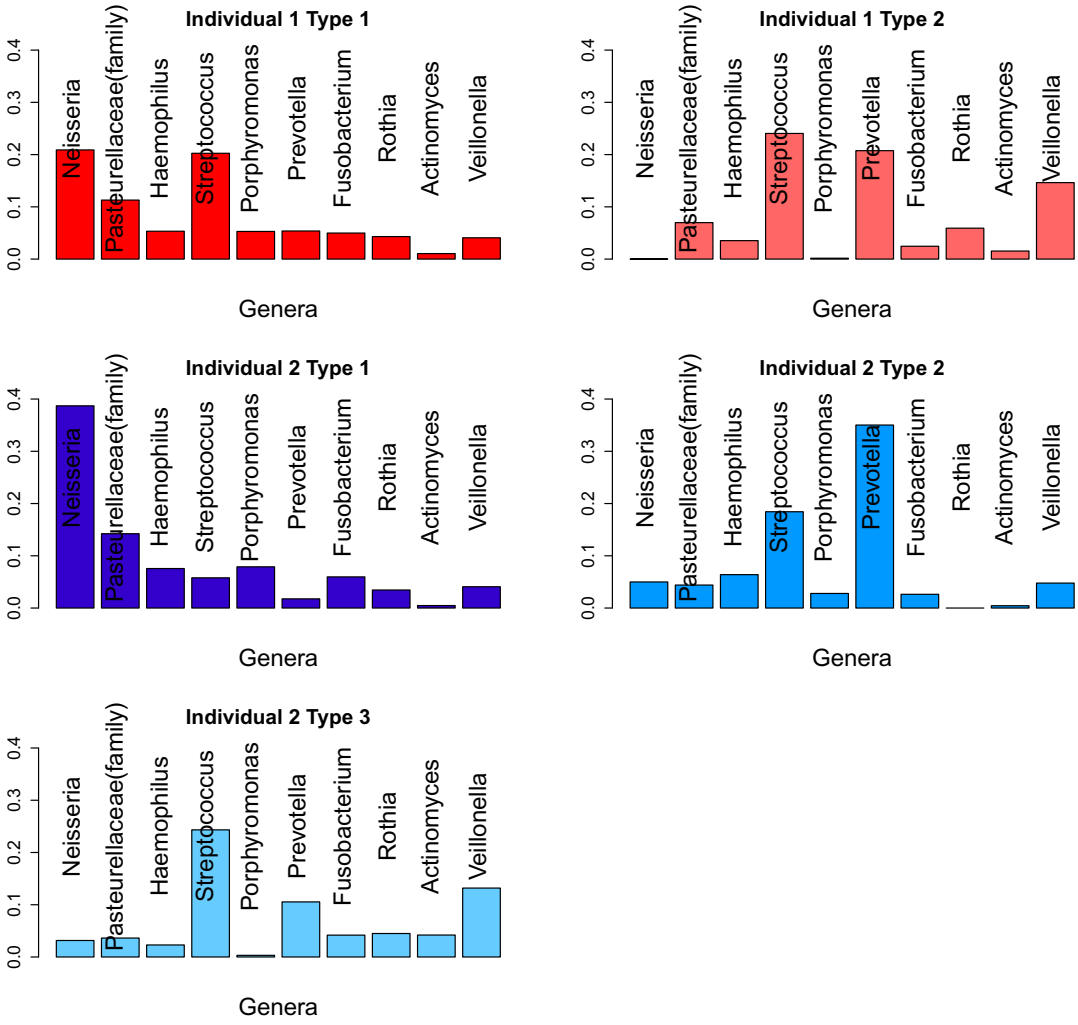


Figure 2.8: Major genera for Tongue feature matrix. The light and dark red bars are two features from Individual 1 and the blue bars are features from Individual 2. Each bar is labeled by the name of the genus or family.

Figure 2.8 shows the profiles of the types extracted from the two individuals, with graphs of abundance of each genus in that type. For individual 1, we see that Type 1 contains higher abundances of *Neisseria*, *Haemophilus*, *Porphyromonas*, *Fusobacterium* and the unclassified genus from the Pasteurellaceae family, while Type 2 includes higher abundance of *Streptococcus*, *Prevotella*, *Rothia*, *Actinomyces* and *Veillonella*. This may well be associated with the action of *Porphyromonas*. One species, *Porphyromonas gingivalis*, has been shown in [22] to

manipulate the host immune system, allowing pathogens to colonise the community. While the OTU from the genus *Porphyromonas* in this dataset is unclassified at species level, it could have a similar effect to the studied species *P. gingivalis*. This would seem consistent with type 1 having higher levels of various Proteobacteria and Fusobacteria closely related to known pathogens. When we look at the features for individual 2, we see a similar picture, with types representing varying levels of *Porphyromonas*. Again, we see with increased *Porphyromonas*, we have an increase in *Neisseria*, *Haemophilus*, *Fusobacterium*, and the unclassified genus of the Pasteurellaceae family, and a corresponding decrease in *Streptococcus*, *Prevotella*, *Actinobacteria* and *Veillonella*. Type 2 may show that the effect of *Porphyromonas* is non-linear with *Prevotella* actually increasing in abundance with low levels of *Porphyromonas*.

We also examine the types in the absence of *Porphyromonas* (type 2 for individual 1 and type 3 for individual 2). For both individuals, we see that these types are dominated by *Streptococcus*, *Prevotella* and *Veillonella*. However, Figure 2.7(b) shows the differences between these types. We see that individual 2 has more *Actinomyces*, and a different distribution between OTUs within the genus *Veillonella*. Similarly, there are subtle differences between the types with high abundance of *Porphyromonas* (type 1 for both individuals). Individual 1's type 1 has higher levels of *Streptococcus* than individual 2's. This might be partially explained by the use of 3 types to model individual 2, allowing separate types to model both high and low levels of *Streptococcus* in cases with high levels of *Porphyromonas*. However, in Figure 2.7(b), we see the presence of higher abundance of a second OTU from the genus *Neisseria* in individual 2's type 1. This cannot be explained by the different numbers of types used to analyse the two individuals. Supervised NMF is able to identify these subtle differences and use them to identify the individuals, even in situations where the large-scale community structure varies a lot between samples within each individual.

We also consider the idea that the types correspond to communities of microbes. When we look at the type without *Porphyromonas*, we can see the makings of a community structure, with a number of microbes (such as *Prevotella*, *Streptococcus* and *Actinobacteria*) that metabolise glucose into pyruvate, which is later

metabolised into lactate, and other microbes such as *Veillonella* which metabolise lactate.

Temporal Dynamics

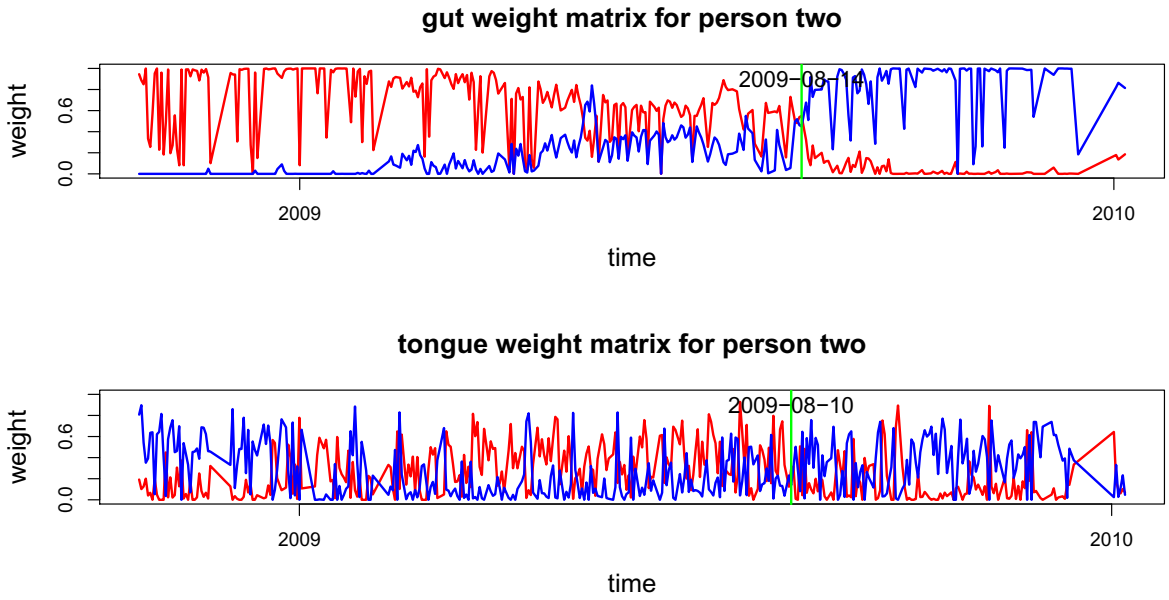


Figure 2.9: Gut and Tongue weight matrix time series plot for person two. The top plot shows the gut weight matrix on the second type (red line) and third type (blue line) from NMF with 3 types. The bottom plot shows the tongue weights on the first type (red line) and the third type (blue line) from NMF for tongue data with 4 types.

To investigate the temporal dynamics of the four body sites' microbiomes, the weight matrices for the gut data and the tongue data are plotted in Figure 2.9. When we apply NMF to person 2 with three types for the gut data (see the upper panel of Figure 2.9), there is a clear shift at around 2009-08-14. This timepoint is highlighted in Figure 2.9. For the gut weight matrix, the dominant weight is initially type 2 and changes to type 3 after this time. For person 2's tongue data, this shift is not very clear when we use only three types. However with four types we can identify a more apparent shift in their weight matrix time series plots. For this data one more feature can bring out more details in the variation of the data. In the lower panel of Figure 2.9, the weight matrix time series plots for

the tongue data relative to these two features show that type 1 is consistently more represented than type 3 in the early part of the study although not always dominant due to the effects of types 2 and 4, type 3 is more represented than type 1 after the changing point (highlighted on the plot). The shift occurs first in the tongue weight matrix and then can be detected about four days later in the gut weight matrix. This suggests that some significant change has taken place in person 2's system at around this time, and that the change has influenced both the gut and the tongue microbiomes.

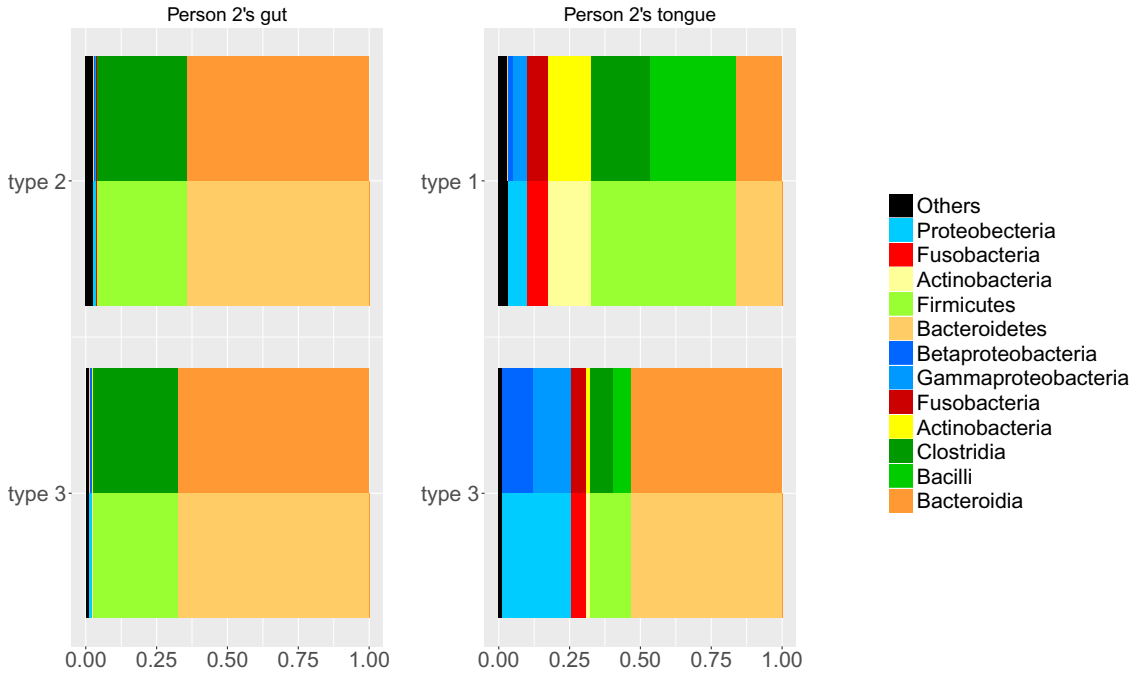


Figure 2.10: Class and phylum proportions in gut and tongue type matrices. The left panels contain two types from person 2's gut data and the right panels are for his tongue data. The top plots present the dominant types at the beginning in the time series plot. The bottom plots present the dominant types after the shift in the time series plot. Similar colours in classes are from the same phylum.

In order to compare the changes which we have identified as taking place in these microbiomes, the distributions of different phyla and classes of OTUs in each feature are presented in Figure 2.10. The top features in this plot are the ones that are more represented in the earlier part of the data (i.e. type 2 for the gut data, and type 1 for the tongue data). The bottom features in this plot are those that are more represented in the later part of the data.

Figure 2.10 shows a similar shift of composition between the two features for both gut and tongue. In both cases, the type which was more represented in the earlier part of the study has a lower proportion of Bacteroidia and a higher proportion of Clostridia. The proportion of Bacteroidia increases and the proportion of Clostridia decreases for both representative features of gut and tongue data in the later part of the study. The consistency of these changes between the two datasets gives further support to our conjecture that this represents a systematic change at this time. The differences between the types are more pronounced in the tongue data. This could be because the tongue is more exposed to external influences, so its microbiome may be more variable. It might also be because we were using four types to model the tongue data and only three for the gut data. Fitting more types gives the types more room to spread out, allowing for more extreme types, and amplifying the differences between the fitted types.

We see that the changes shown in Figure 2.10 are consistent with the earlier interpretation of the types in Figure 2.8. We used four types here to model the microbiome, but we can see in Figure 2.10 that the dominant type after the transition includes much higher abundances of Bacteroidetes (including *Porphyromonas* and *Prevotella*, which has been associated with Periodontal disease [21]) and Proteobacteria (including *Neisseria* and *Haemophilus*) and lower levels of Firmicutes (including *Streptococcus* and *Veillonella*) and Actinobacteria (including *Rothia* and *Actinomyces*). Note that the types in Figure 2.8 are fitted from the training data, which is entirely before the state change in person 2.

Having identified the state change using NMF, we ask whether NMF was a necessary tool for identifying the change. First we compare a naive examination of the composition of the microbiome by class. Figure 2.11 shows the smoothed proportion of each class over time in person 2's gut and tongue microbiomes. We see that there are no clear changes in composition at this level, indicating that this is not an obvious change to identify.

For comparison, we also use UniFrac and PCoA for person 2's gut and tongue data. We see from Figure 2.12 that the first 3 principle coordinates for the gut data and the first 4 coordinates for the tongue data do not reveal this change. It is only when we examine the 4th principle coordinate for the gut data and the

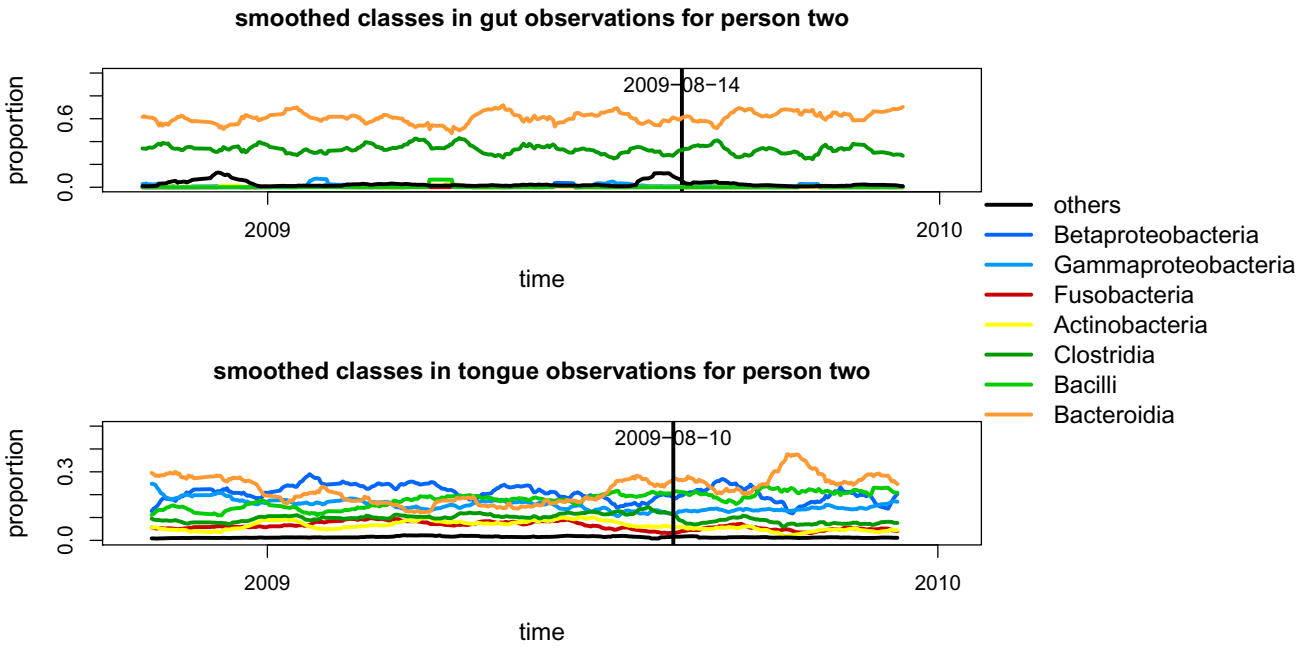


Figure 2.11: Moving average of class proportions in gut and tongue observations.

5th principle coordinate for the tongue that we are able to detect the changes. The difficulty of finding this explains why this pattern was not found in the many previously published analyses of these data. This is made more difficult by the common practice of examining only the first three principal coordinates. It is possible to find the pattern using UniFrac, if one knows what to look for, but NMF certainly makes the pattern much easier to find.

2.3.3 The Qin data

The unsupervised and supervised NMF results are calculated in my master thesis. The comparisons and the interpretations are completed during my PhD program.

The Qin dataset [88] contains human gut metagenome samples extracted from 99 healthy people and 25 IBD patients. The data include 2804 different reactions.

Unsupervised NMF results for Qin data

We choose the NMF rank for the Qin data as six and apply unsupervised NMF on the data. A projection of the data onto a plane is shown on the left panel

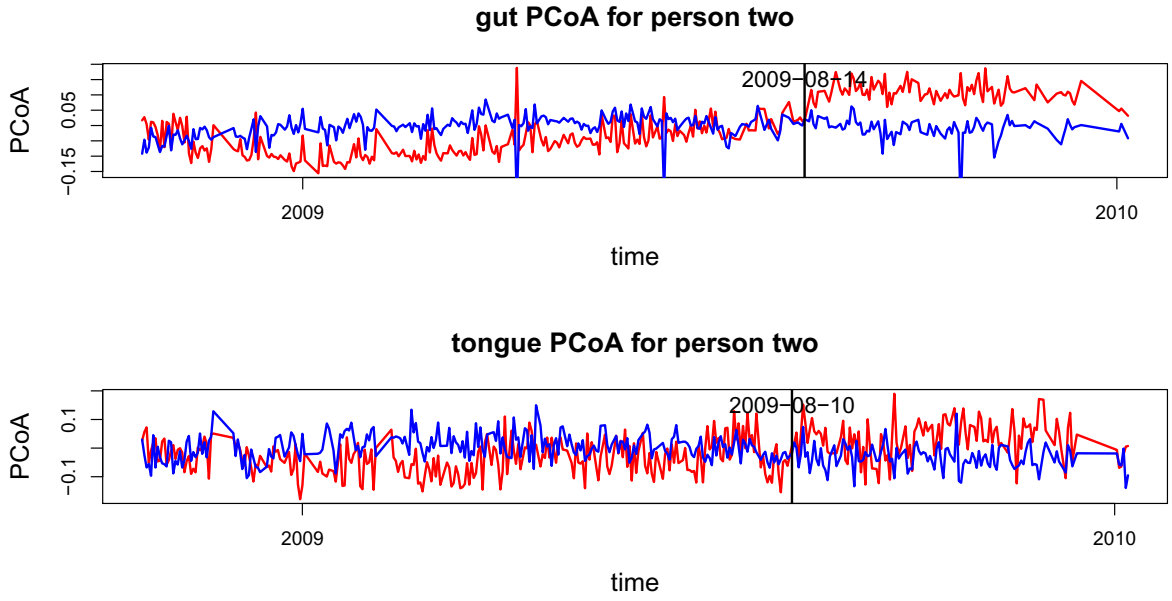


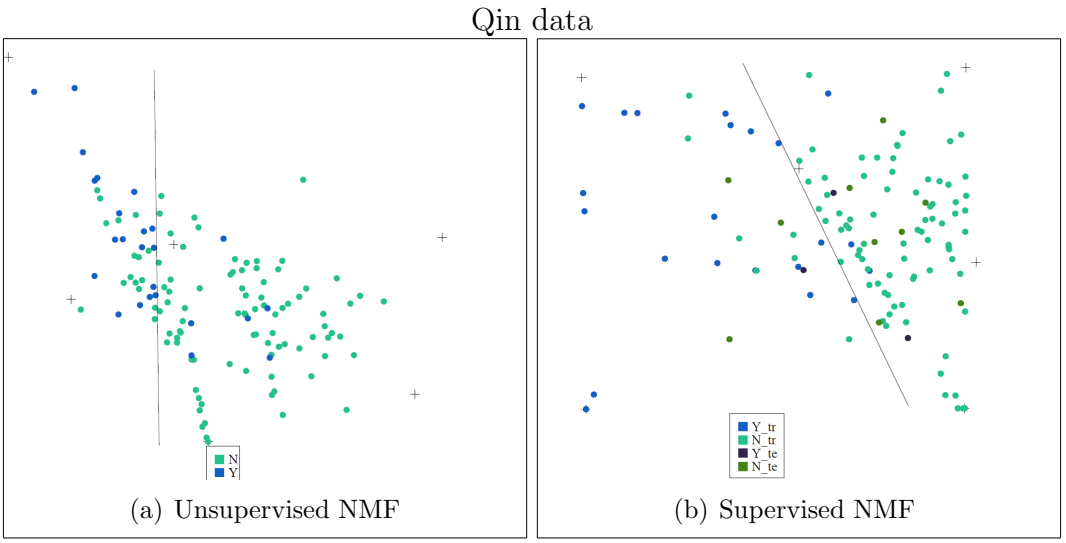
Figure 2.12: Gut and Tongue principle coordinates based on UniFrac time series plot for person two. The top plot shows the gut UniFrac matrix on the third principle coordinate (red line) and fourth principle coordinate (blue line). The bottom plot shows the tongue UniFrac matrix on the second principle coordinate (red line) and the fifth principle coordinate (blue line).

of Figure 2.13. From the plot, we can see that about 19 of the IBD patients can be separated from healthy people. The seperation is similar to the results of BiomeNet [96]. The plot shows that two of these features are more related to IBD patients and the other four more related to healthy people. This is consistent with what we find using supervised NMF.

Supervised NMF results for Qin data

The sample size of patients is much smaller than the sample size of healthy people. So we perform a classification giving the patients a weight of 4 to balance the class sizes. (For supervised NMF, these weights don't affect the fitted matrices T and W, only the classifier applied to the weight matrix W.) This means that the classifier that assigns all samples to one class will have an accuracy of about 50%. We perform a 10-fold cross-validation on the whole data. Each time we use 9-folds as training data and the remaining observations as test data.

We find 2 types are enough for patients and 4 types for healthy people. We



(a) Unsupervised NMF (b) Supervised NMF

Figure 2.13: **Left**: Unsupervised NMF based on 6 types. The blue points are from IBD patients and the green ones are from healthy people. **Right**: Supervised NMF on both training and test data. The blue points are training data from patients and green points are training data from healthy people, the dark blue points are test data from patients and the dark green points are test data from healthy people.

perform supervised NMF and fit a logistic regression using the training data weight matrices (with patients given weights of 4) and perform a prediction on the test data. The average of the weighted misclassification error over the 10 folds is 0.233 with a standard error of 0.0487.

The projections of both training and test data in one fold of the 10-fold cross-validation are plotted in the right panel of Figure 2.13. It shows a quite good separation between these two groups. The classification is not perfect, but is an improvement upon previous methods, such as BiomeNet [96].

The comparisons with support vector machine and random forest methods are summarized in Table 2.11. The dataset is split to the same 10 folds as supervised NMF. The best parameters are tuned by a 10-fold cross-validation on the whole dataset. The best cost parameter in SVM is 3 for radial basis kernel and 1 for the other three kernels. The best gamma parameter is 10^{-4} for radial basis kernel, 0.1 for polynomial kernel and 0.001 for sigmoid kernel. No method performs significantly better than supervised NMF.

Interpretation of the results

The six type vectors are highly sparse with each vector sum equal to 1. We use a cut-off of 0.5% for each type to find the distribution of each type over the major reaction groups. Here each reaction group includes the different reactions that correspond to the same enzyme-coding gene, thus each category can also be understood as corresponding to one enzyme-coding gene. The type distribution over 17 enzyme-coding genes or reaction groups is shown in Figure 2.14. We can observe that the *IBD Type 2* is quite different from other types, with large abundance on the fourth and fifth enzyme-coding genes and that both IBD types have weight zero on the second enzyme-coding gene. Each individual's metagenome profile is expressed as a linear combination of these six types, the weight distribution over each type is shown in Figure 2.15, where the top part of each bar presents the distribution of the weights for healthy individuals for the corresponding type, and the bottom part of each bar is for the weight distribution of IBD patients with each patient counted as 4 times to make the results comparable to the healthy individuals. From Figure 2.15, we can see the IBD patients mainly have non-zero weights on *IBD Type1*, *IBD Type2*, *Healthy Type 1* and *Healthy Type 2*, and healthy individuals mainly have non-zero weights on *Healthy Type 1*, *Healthy Type 2* and *Healthy Type 4*. It seems that the *IBD Type 2* typically represents a group of IBD patients and *Healthy Type 2* represents a group of healthy individuals with these two types distributed very differently over the enzyme-coding genes shown in Figure 2.14.

According to Figure 2.14, the first three reaction groups contribute more to healthy types and the fourth and fifth reaction groups contribute more to IBD patients (mainly to *IBD Type 2*). Reactions in the first group are all in macrolide biosynthesis. Macrolides are protein synthesis inhibitors and can be used as an antibiotics treatment of inflammatory diseases including inflammatory bowel disease [98] [78] [57]. The second reaction group is involved in polycyclic aromatic hydrocarbon degradation and the third group is in carotenoid biosynthesis. Polycyclic aromatic hydrocarbons (PAHs) are one family of ubiquitous environmental toxicants. This family has contribution significantly to development of Colorectal cancer (CRC), a disease highly linked to IBD [27]. Carotenoids can enhance

the human immune system's effectiveness [47]. As IBD is a kind of autoimmune disease, this could explain why these two compounds are lower in IBD patients' features. The fourth group in *IBD Type 2* is involved in ascorbate and aldarate metabolism and the fifth group in amino sugar and nucleotide sugar metabolism, fructose and mannose metabolism, glycolysis and gluconeogenesis and additional pathways. These are concordant with BiomeNet's findings in subnetworks 38, 64 and 73. Comparing our reaction groups with the three subnetworks, we notice that reaction group 4 can be found in subnetwork 64 and group 5 has some overlaps with subnetwork 38 and subnetwork 73. These three subnetworks were discovered to have a larger contribution to IBD samples than healthy ones.

Table 2.11: Comparison of test errors for support vector machine with linear kernel (SVM l), with polynomial kernel (SVM p), with sigmoid kernel (SVM s), with radial basis kernel (SVM r), RandomForest (RF), RandomForest with sparse variables removed (RFrm) and Supervised NMF. The first four rows are the misclassification errors on the test data. The Mammal and Qin data include the mean misclassification errors and standard deviations(below the means in brackets) for cross-validation. Best prediction for each dataset is highlighted by red color.

dataset	SVM l	SVM p	SVM s	SVM r	RF	RFrm	Supervised NMF
Gut	0	0.2335	0	0.0661	0	0	0
Tongue	0.0202	0.2694	0.0202	0.0484	0.0081	0.0173	0.0040
Left Palm	0.1245	0.2691	0.1446	0.2691	0.1285	0.0916	0.0924
Right Palm	0.3455	0.2724	0.3455	0.1667	0.0732	0.0560	0.1789
Mammal	0.0714	0.1428	0.0714	0.1071	0.1429	0.1071	0
	[0.0461]	[0.0505]	[0.0461]	[0.0504]	[0.0504]	[0.0504]	[0]
Qin	0.3178	0.3359	0.2592	0.2853	0.2299	0.2299	0.2333
	[0.0567]	[0.0530]	[0.0516]	[0.0494]	[0.0573]	[0.0467]	[0.0515]

2.4 Conclusion

The NMF analysis can provide a range of interpretable conclusions about the data sets. For metagenomic data, the features extracted can be mapped to metabolic pathways. For OTU data, the features correspond to communities of OTUs, and can be studied in terms of the proportion of each phylum, class or genus. In any case, looking at the results of the NMF can reveal important patterns or differences between individuals that are not apparent from the original data. We were able to

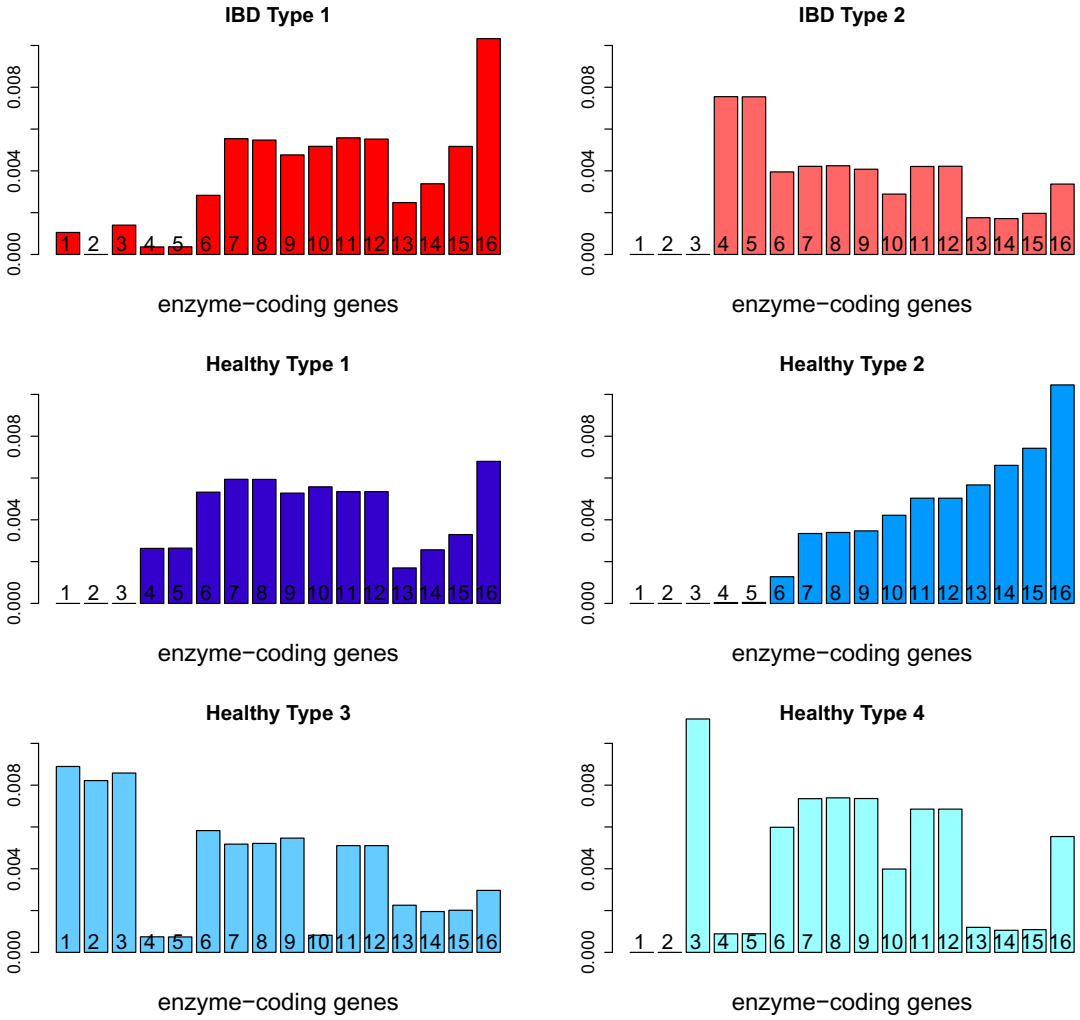


Figure 2.14: Qin data: The distribution of each type over major enzyme-coding genes: *IBD Type 2* typically represents a group of IBD patients and *Healthy Type 2* represents a group of healthy individuals with these two types distributed very differently over the enzyme-coding genes.

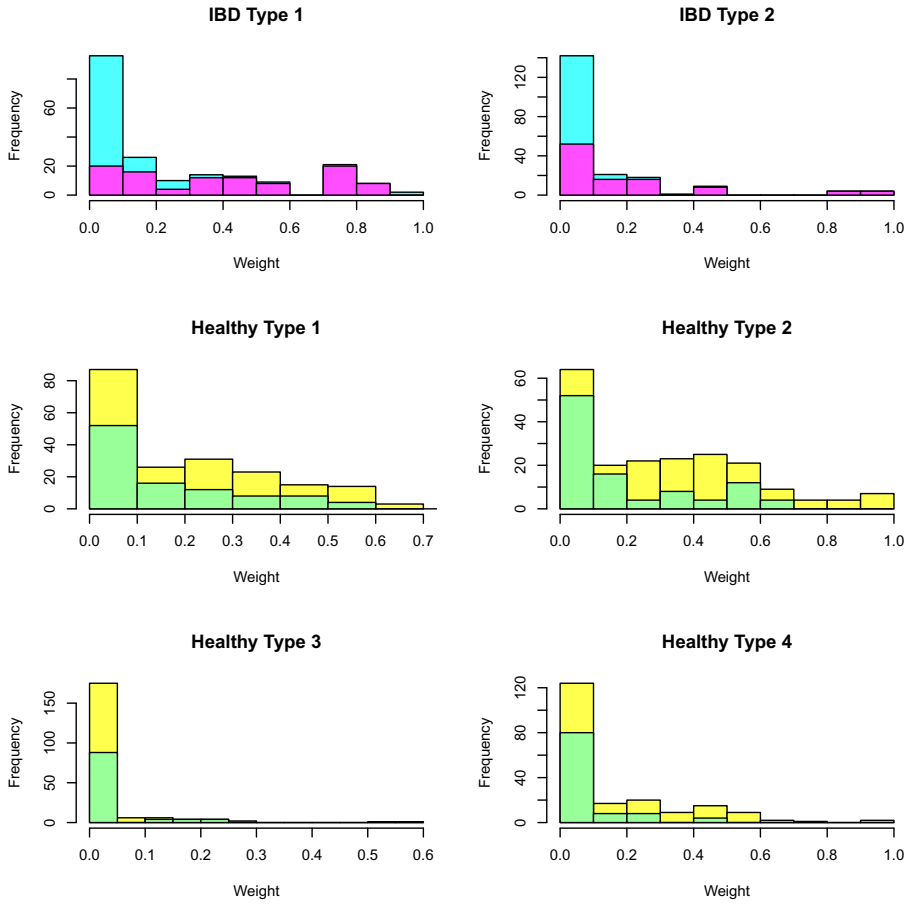


Figure 2.15: The weights distribution over each type for heathy individuals (top for each bar) and IBD patients (bottom for each bar): the IBD patients mainly have non-zero weights on *IBD Type1*, *IBD Type2*, *Healthy Type 1* and *Healthy Type 2*, and healthy individuals mainly have non-zero weights on *Healthy Type 1*, *Healthy Type 2* and *Healthy Type 4*.

identify this type of pattern in all three real data sets — the difference in macrolide synthesis pathways for the non-ruminant herbivores; the change in composition of the gut and tongue microbiomes for person 2 in the moving picture data; and the differences in various pathways for the Qin data.

The simulation results show that supervised NMF can recover the right NMF rank values based on which a good classification result can be achieved. Supervised NMF can effectively reduce the dimensionality of the data to a non-negative and most often sparse data matrix, which contains sufficient discriminative information for classification purposes. In addition to the accuracy for classification, these typical features are the community signatures for each class of objects and their interpretation can often uncover important information about the differences between different classes of objects. Simulations of community dynamics under a Holling type II model show that plausible models of community dynamics can lead to the type of additive subcommunity structure assumed by NMF, and that in such a case, NMF is able to identify biologically meaningful types representing the subcommunities.

Chapter 3

Deconvolution density estimation with penalized MLE

3.1 Introduction

Measurement error is a common problem with data. It occurs when the apparatus measuring a variable is not perfect, and the value it returns is a random variable, based on the true value. A simple example is additive measurement error, where the recorded value is the true value (which is itself a random variable) plus a random error. More formally, the observed value is given by $Y = X + E$, where X is the true value and E is a random measurement error. This can happen with a lot of measurement apparatus. It can also happen when Y is an estimated quantity (such as an MLE estimate from a particular model or a test statistic on some data) for which there is no analytic solution. In this case, the estimates Y are subject to convergence error, which behaves like measurement error. This is the motivating situation that we discuss in Chapter 4.

In this chapter, we look at the problem of estimating the density of the underlying variable X from a sample of observations with measurement error. This is referred to as deconvolution. Figure 3.1 shows an example of this problem. The green curve is the density function of interest: it follows a scaled chi-squared distribution with 4 degrees of freedom. The black curve is the density of variable Y with measurement error following a scaled beta distribution. We see that the distribution with measurement error is very different from the original distribution, so some method is needed to correct for this difference.

Formally, we have a sample of observations $Y_1, \ldots, Y_n$ given by the additive error model $Y_j = X_j + \epsilon_j$, where ϵ_j are i.i.d. The latent variables X_j are i.i.d. with density f_x, but are not observed. We are interested in estimating the density f_x from this data, and either a known error distribution for ϵ_j, or a separate sample from the error distribution (for example obtained by repeated measurements of one observation).

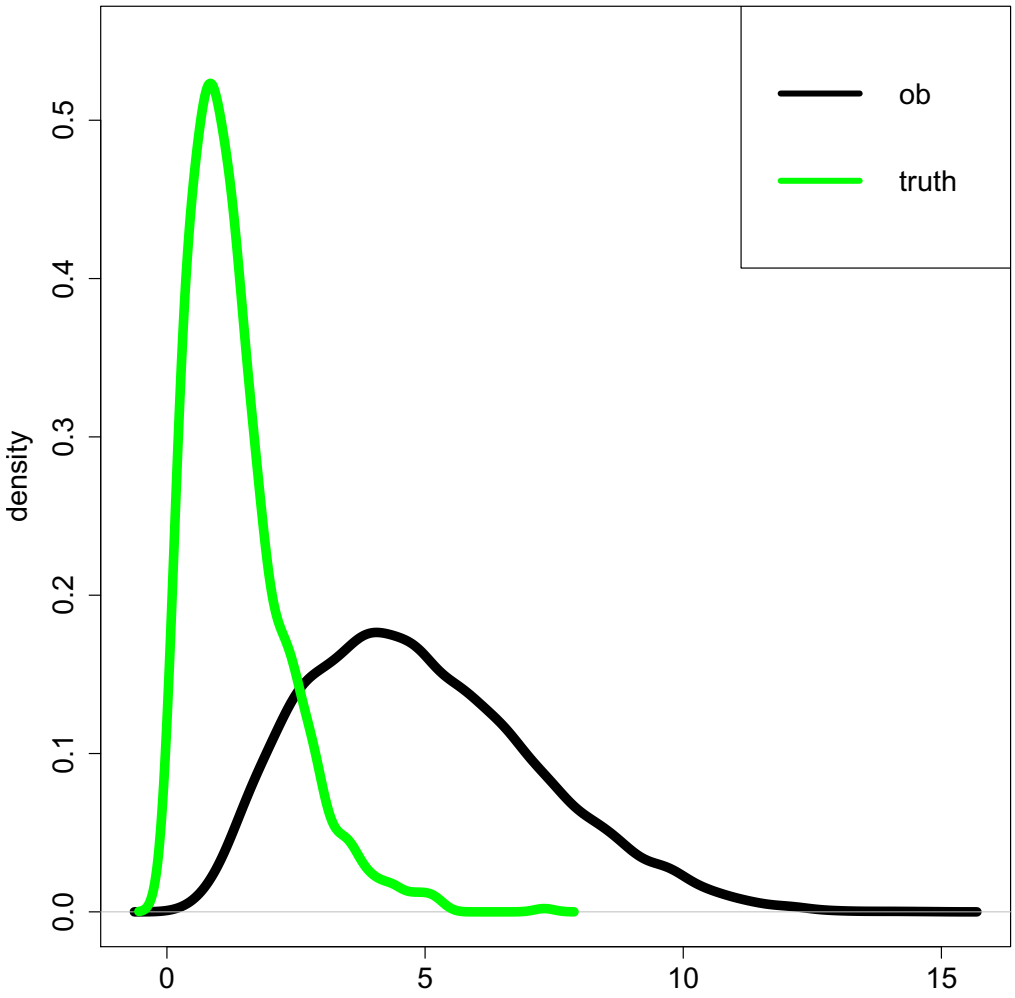


Figure 3.1: The black curve is the density of the contaminated data. The green curve is the density of the underlying truth.

There have been a number of methods developed for this problem, mostly based on the characteristic function $\chi_X(t) = \mathbb{E}e^{iX}$. The key to these methods is that the convolution that gives the distribution of Y becomes pointwise multiplication of characteristic functions. That is, if Y has distribution function given by the convolution $F_Y(y) = \int_{-\infty}^{\infty} f_x(x)F_\epsilon(y-x)\,dx$, then the characteristic function is given by elementwise multiplication $\chi_Y(t) = \chi_X(t)\chi_\epsilon(t)$. Elementwise multiplication is easily inverted by elementwise division, i.e. $\chi_X(t) = \frac{\chi_Y(t)}{\chi_\epsilon(t)}$, so if we know the characteristic functions of Y and ϵ, then we can calculate the characteristic function of X. Different deconvolution methods in the literature are based on different estimators for the characteristic function of Y (and sometimes ϵ), and different regularisation and correction (the estimated χ_X is not guaranteed to be the characteristic function of a distribution). This formulation in terms of the characteristic function also highlights the difficult cases — when $\chi_\epsilon(t)$ is very small, the quotient becomes much larger, so estimation errors in the characteristic function are magnified. In cases where $\chi_\epsilon(t)$ converges quickly to zero as $t \to \infty$, the error distribution is called supersmooth, and the deconvolution problem is particularly challenging.

One widely used method is developed by Liu and Taylor [71]. They use a kernel density estimator for the characteristic function χ_Y, a known error distribution, and a method based on minimising mean squared error (MSE) to select the boundary and bandwidth. They prove the method is consistent in cases where the error distribution and kernel function are symmetric.

Assuming the error distribution as known is unrealistic. A more reasonable approach is to model the error distribution parametrically or nonparametrically. Delaigle and Gijbels [23] used the moment estimators for the parametric error distribution parameters in their example, with the main contribution of the paper being a bootstrap bandwidth selection method for the deconvolution kernel density estimation of f_x. The R package `decon` [105] implements their method for Gaussian or Laplace error, using either direct computation or a fast Fourier transform (FFT).

The error distribution can be nonparametrically estimated from repeated observations of the contaminated variable Y, see for example, Delaigle et al. [24], Comte et al. [20] and Susko et al. [102]. Alternatively it can be estimated from a

pure error sample (which can be obtained through repeated measurements of the same quantity which is independent of the observed sample of Y). For example, Kerkyacharian et al. [51] used the empirical characteristic function from a pure error sample for estimating χ_ϵ. There are a few other approaches to deconvolution with a pure error sample, mainly differing in details such as bandwidth selection.

The R package `deamer` [49] implements several deconvolution methods based on the FFT algorithm, including situations for known error density; for unknown error density with an auxiliary sample of i.i.d. pure errors (method by Comte and Lacour [18]) which is only proven consistent under the assumption that f_x is ordinary smooth or supersmooth; and for unknown error density with replicate observations for variable Y with the assumption that the error distribution is symmetric around zero (methods by Delaigle et al. [24] and Comte et al. [20]).

While the Fourier-based methods are mathematically elegant, estimation of the characteristic function is much more challenging than estimation of other distributional quantities, and because of the division by $\chi_\epsilon(t)$, values where $\chi_\epsilon(t)$ is small can cause instability in the estimates. The methods in the literature get around this by limiting the range of t. This hard limitation can result in poor estimation. Another source of significant errors comes in the correction stage where the estimate $\hat{\chi_X}(t) = \frac{\hat{\chi_Y}(t)}{\hat{\chi_\epsilon}(t)}$ is adjusted to become the characteristic function of a distribution. Because of these difficulties, existing methods perform very poorly unless the sample size and the signal-noise ratio (SNR) are both large.

Another way for estimating the deconvoluted density function is by maximizing the log-likelihood of the data with a smoothness penalty for the estimated density. (It was shown by Laird [58] that without the penalty, the MLE estimate is a discrete distribution.) Most methods use the EM algorithm to optimize this likelihood. The "M" step is computationally challenging for this problem, and the methods of Silverman [99] and Green [38] do not fully overcome these challenges for deconvolution problems. Liu et al. [70] developed a functional EM algorithm (FEM) with the M-step being equivalent to solving a fourth order ordinary differential equation (ODE). Instead of the commonly used penalty given by the smoothness of the estimated density, FEM used the penalty given by the smoothness of the logarithm of the density function. This removes the non-negativity

constraint on the density function. The drawback is that there is no analytic solution to the ODE, so they use numerical solutions to the case where the function is restricted to lie in a lower dimensional space generated by B-splines. Other penalized likelihood approaches also restrict to a lower dimensional space to simplify computation, for example Kuusela and Panaretos [56] and Wood [107].

Here, we develop a completely new method to maximize the penalized log-likelihood with the common smoothness penalty on f_x. We also provide a proof of on the convergence and consistency of the deconvolution density estimation based on the penalized likelihood methods. To the best of our knowledge, this is the first general proof of consistency of penalized maximum likelihood for deconvolution. The closest existing proof appears to be provided by Madrid-Padilla et al. [73], which proves consistency but not convergence rate of the maximum likelihood estimate subject to a constraint on the penalty function, when the true density is restricted to a space satisfying a set of hard-to-parse conditions, and the error distribution is continuous, has finite entropy, and its density is Lipschitz.

Our method can overcome both the instability problems associated with the Fourrier-based methods, and the restriction of the estimated function lying in a lower dimensional space spanned by a particular set of basis functions in the FEM method. Our method (termed P-MLE) can produce better estimates, particularly when the sample size is small, or the signal-noise ratio is low.

The outline of this chapter is as follows. In Section 3.2, we introduce our method and define the estimator. In Section 3.3, we explain the procedures used to solve the challenges occur in implementation of the method. In Section 3.4, we discuss the theoretical convergence properties of our method with proofs. In Section 3.5, we compare P-MLE with `decon`, `deamer` and `FEM` on simulated datasets. In Section 3.6, we apply our method to real data and compare the performance with `decon`, `deamer` and `FEM`. The chapter finishes with the conclusions in Section 3.7.

3.2 Deconvolution based on penalized log-likelihood

We want to estimate the density function $f_x(x)$ of a continuous random variable X from a sample $\{y_1, y_2, \ldots, y_n\}$ of the random variable $Y = X + \epsilon$, where ϵ is a

random variable independent from X. For simplicity, we will start with the case where the distribution of ϵ is known. If the distribution of ϵ is unknown, but we have a pure error sample, $\{e_1, e_2, \cdots, e_M\}$, then we may apply our method with the empirical distribution of ϵ from this sample. We will also assume that X has finite support $[l, u]$. In theory, we could set the support to be $(-\infty, \infty)$, but this causes practical challenges with the optimization.

The density function f_y is obtained via the convolution

$$f_y(y) = \mathbb{E}_\epsilon(f_x(y - \epsilon))$$

so the log-likelihood of our data for a particular density function f_x is

$$\sum_{i=1}^{n} \log f_y(y_i) = \sum_{i=1}^{n} \log \left(\int_l^u f_x(x) d\mu_\epsilon(y_i - x) \right)$$

where μ_ϵ is the probability measure of the error distribution. (We need to assume that X has a continuous distribution, but our method has no problems with ϵ having a discrete or mixed distribution.)

Our method is to minimize the negative log-likelihood function of $\{y_1, y_2, \cdots, y_n\}$ plus a penalty term on the smoothness of function f_x. The smoothness penalty $\psi(f_x)$ is given by

$$\psi(f_x) = \langle f_x'', f_x'' \rangle = \int_l^u f_x''(x)^2 dx$$

where $\langle ., . \rangle$ denotes the inner product on the Hilbert space $L^2[l, u]$. This penalty is widely used in the smoothing splines method [39]. This gives us the penalized negative log-likelihood function:

$$J = -\sum_{i=1}^{n} \log \left(\int_l^u f_x(x) d\mu_\epsilon(y_i - x) \right) + \lambda_n \langle f_x'', f_x'' \rangle \tag{3.1}$$

where λ_n is the smoothness penalty tuning parameter used to control the smoothness of f_x. To minimize J, instead of using the EM algorithm, we use integration by parts twice to express the log-likelihood as a function of f_x''. We can then solve for the optimal function f_x'' and obtain f_x by integrating twice.

Using integration by parts twice to compute the integral defining $f_y(y_i)$, we get

$$\begin{aligned} f_y(y_i) &= \int_l^u f_x(x) d\mu_\epsilon(y_i - x) \\ &= [-f_x(x)F_\epsilon(y_i - x)]_l^u + \int_l^u F_\epsilon(y_i - x) f'_x(x) dx \\ &= [-f_x(x)F_\epsilon(y_i - x)]_l^u - [f'_x(x)H(y_i - x)]_l^u + \int_l^u f''_x(x)H(y_i - x)dx \end{aligned} \tag{3.2}$$

where F_ϵ denotes the cumulative distribution function of f_ϵ and $H(v) = \int_{-\infty}^{v} F_\epsilon(u)du$. (If ϵ has a particularly heavy-tailed distribution, we can change the lower limit of the integral, so that H is well-defined.)

We assume that f_x is a density function which is twice differentiable on $\mathbb{R}$ and $[l, u]$ contains the positive support of f_x, so without loss of generality we let $f_x(l) = f'_x(l) = 0$ and $f_x(u) = f'_x(u) = 0$. The first two terms in (3.2) vanish, giving

$$f_y(y_i) = \int_l^u f''_x(x)H(y_i - x)dx = \langle f''_x, h_i(x)\rangle$$

where $h_i(x) = H(y_i - x)$.

Using this inner product, our objective function (3.1) becomes

$$J = -\sum_{i=1}^{n} \log(\langle f''_x, h_i\rangle) + \lambda_n \langle f''_x, f''_x\rangle \tag{3.3}$$

We want to minimize this J subject to the constraints:

$$f_x(l) = f'_x(l) = 0, \tag{3.4}$$

$$f_x(u) = f'_x(u) = 0, \tag{3.5}$$

$$\forall r \in (l, u), f_x(r) \geqslant 0, \tag{3.6}$$

$$\int_l^u f_x(r)dr = 1. \tag{3.7}$$

Constraint (3.4) gives a unique solution for f_x given f''_x. Constraint (3.4) and (3.5) are boundary conditions needed to integral back to f_x from f''_x. The conditions (3.5)-(3.7) can be rewritten as conditions on f''_x.

For Condition (3.5), note that

$$\langle f_x''(r), 1\rangle = \int_l^u f_x''(r)dr = f_x'(u) - f_x'(l)$$

and

$$\langle f_x''(r), r\rangle = \int_l^u f_x''(r)rdr = [f_x'(r)r]_l^u - \int_l^u f_x'(r)dr = f_x'(u)u - f_x'(l)l - f_x(u) + f_x(l)$$

With Condition (3.4), we have

$$\langle f_x'', 1\rangle = 0 \Longleftrightarrow f_x'(u) = 0$$

and

$$\langle f_x'', r\rangle = 0 \Longleftrightarrow f_x(u) = 0$$

From Constraints (3.4), (3.5) and (3.7), we get

$$\langle f_x''(r), r^2\rangle = [f_x'(r)r^2]_l^u - 2\int_l^u f_x'(r)rdr = -2[f_x(r)r]_l^u + 2\int_l^u f_x(r)dr = 2$$

Under Constraints (3.4) and (3.5), the above condition is equivalent to (3.7).

For the non-negativity constraint, (3.6), we integrate $f_x'(r)$ by parts to get the following inner products.

$$f_x(r) = \int_l^r f_x'(s)ds = [f_x'(s)(s-r)]_l^r - \int_l^r f_x''(s)(s-r)ds = \langle f_x''(s), (r-s)_+\rangle \tag{3.8}$$

$$f_x(r) = -\int_r^u f_x'(s)ds = -[f_x'(s)(s-r)]_r^u + \int_r^u f_x''(s)(s-r)ds = \langle f_x''(s), (s-r)_+\rangle \tag{3.9}$$

We can also take any affine combination of Equations (3.8) and (3.9) to evaluate $f_x(x)$. That is $f_x(x) = \lambda\langle f_x''(r), (x-r)_+\rangle + (1-\lambda)\langle f_x''(r), (r-x)_+\rangle$ for any λ. In particular, it is convenient to set

$$b_x(r) = \frac{(u-x)^2((u-x)+3(x-l))}{(u-l)^3}(x-r)_+ + \frac{(x-l)^2((x-l)+3(u-x))}{(u-l)^3}(r-x)_+ - 2\frac{(u-x)^2(x-l)^2}{(u-l)^3}$$

so that

$$\begin{aligned}
\langle b_x(r), 1\rangle &= \frac{(u-x)^2((u-x)+3(x-l))}{(u-l)^3}\int_l^x (x-r)\,dr + \frac{(x-l)^2((x-l)+3(u-x))}{(u-l)^3}\int_x^u (r-x)\,dr \\
&\qquad - 2\frac{(u-x)^2(x-l)^2}{(u-l)^3}\int_l^u 1\,dr \\
&= \frac{1}{2}\left(\frac{(u-x)^2((u-x)+3(x-l))}{(u-l)^3}(x-l)^2 + \frac{(x-l)^2((x-l)+3(u-x))}{(u-l)^3}(u-x)^2\right) - 2\frac{(u-x)^2(x-l)^2}{(u-l)^2} \\
&= 0
\end{aligned}$$

$$\begin{aligned}
\langle b_x(r), r\rangle &= \langle b_x, r-x\rangle \\
&= \frac{(u-x)^2((u-x)+3(x-l))}{(u-l)^3}\int_l^x (r-x)(x-r)\,dr + \frac{(x-l)^2((x-l)+3(u-x))}{(u-l)^3}\int_x^u (r-x)(r-x)\,dr \\
&\qquad - 2\frac{(u-x)^2(x-l)^2}{(u-l)^3}\int_l^u (r-x)\,dr \\
&= \frac{1}{3}\left(-\frac{(u-x)^2((u-x)+3(x-l))}{(u-l)^3}(x-l)^3 + \frac{(x-l)^2((x-l)+3(u-x))}{(u-l)^3}(u-x)^3\right) \\
&\qquad - \frac{(u-x)^2(x-l)^2}{(u-l)^3}((u-x)^2-(x-l)^2) \\
&= \frac{(u-x)^2(x-l)^2}{(u-l)^3}\left(\frac{1}{3}\left(3(u-x)^2 - 3(x-l)^2\right) - ((u-x)^2-(x-l)^2)\right) \\
&= 0
\end{aligned}$$

And the non-negativity constraint is $\langle f_x'', b_{x^*}(r)\rangle \geqslant 0$ for all $x^* \in (l,u)$.

Thus we have converted the constraints in the minimization problem from (3.3) to Hilbert Space inner product conditions on f_x'':

$$J = -\sum_{i=1}^{n} \log(\langle f_x'', h_i \rangle) + \lambda_n \langle f_x'', f_x'' \rangle \tag{3.10}$$

$$\langle f_x'', 1 \rangle = 0 \tag{3.11}$$

$$\langle f_x'', r \rangle = 0 \tag{3.12}$$

$$\langle f_x'', r^2 \rangle = 2 \tag{3.13}$$

$$\langle f_x'', b_{x^*}(r) \rangle \geqslant 0, \forall x^* \in (l, u) \tag{3.14}$$

Recall that h_i is calculated from F_ϵ. When the distribution function of ϵ is unknown, h_i can be calculated from the empirical distribution of the random sample $\{e_1; e_2; \cdots; e_M\}$. Without loss of generality, we assume $e_1 \leqslant e_2 \leqslant \cdots \leqslant e_M$, the empirical distribution of ϵ is a step function

$$\hat{F}_\epsilon(\epsilon) = \frac{1}{M} \sum_{m=1}^{M} 1_{\{e_m \leqslant \epsilon\}}(\epsilon)$$

Here $1_{\{e_m \leqslant \epsilon\}}(\epsilon)$ is the indicator function, which is 1 when $\{e_m \leqslant \epsilon\}$ and 0 otherwise. Thus $\hat{H}(v) = \int_{-\infty}^{v} \hat{F}_\epsilon(u)du = \frac{1}{M} \sum_{v > e_m} (v - e_m)$ is a piecewise linear function. Then

$$\hat{h}_i(x) = \hat{H}(y_i - x) = \frac{1}{M} \sum_{x < y_i - e_m} (-x + y_i - e_m)$$

or

$$\hat{h}_i(x) = \begin{cases} y_i - x - \frac{1}{M} \sum_{m=1}^{M} e_m & \text{if } x < y_i - e_M \\ \frac{k(y_i - x)}{M} - \frac{1}{M} \sum_{m=1}^{k} e_m & \text{if } y_i - e_{k+1} \leqslant x < y_i - e_k (k = M-1, \cdots, 1) \\ 0 & \text{if } x \geqslant y_i - e_1 \end{cases} \tag{3.15}$$

So our penalized MLE method can also be applied to deconvolution with unknown error distribution but a pure error sample, with $h_i(x)$ replaced by $\hat{h}_i(x)$.

When only an auxiliary sample of replicate observations $Y^* = X + \epsilon^*$ independent from Y^* is available, where ϵ^* is independent from ϵ, we can estimate the true density of X from the average of the two contaminated observed samples $\frac{Y+Y^*}{2}$ with error as $\frac{\epsilon+\epsilon^*}{2}$. Under the assumption that the error distribution is

symmetric, which is true for most measurement errors, $\frac{\epsilon+\epsilon^*}{2}$ and $\frac{\epsilon-\epsilon^*}{2}$ follow the same distribution. So we can obtain the pure error sample as the difference of the two observations divided by two $\frac{Y-Y^*}{2} = \frac{\epsilon-\epsilon^*}{2}$. An application of this procedure is shown in the real data analysis section.

The advantage of writing the optimization problem in this way is that in the Hilbert space L^2 $([l,u])$, we can decompose f_x'' as a linear combination of $\{h_i|i=1,\ldots,n\}, 1, r, r^2$, and $\{b_x|x\in(l,u)\}$ plus a function orthogonal to all these elements. If we let f_0 be the linear combination of these elements, and let $f_\perp$ be the orthogonal part, we see that $f_\perp$ only affects the penalty term $\langle f_x'', f_x''\rangle$, and because of the orthogonality, $\langle f_x'', f_x''\rangle = \langle f_0, f_0\rangle + \langle f_\perp, f_\perp\rangle$ is clearly minimized when $f_\perp = 0$, so we have shown that the optimal solution is a linear combination of $\{h_i|i=1,\ldots,n\}, 1, x, x^2$, and $\{b_x|x\in(l,u)\}$. Thus, we have reduced the constrained optimization over the whole Hilbert space $L^2([l,u])$ to a constrained optimization of the coefficients in this basis. This is still infinite dimensional. However, we can get a good approximate solution to the problem by only requiring a finite subset of the non-negativity constraints. After we restrict to this condition, the problem has become a standard finite-dimensional constrained optimization problem.

More precisely, suppose we choose $k-3$ basis functions of $\{b_x|x\in(l,u)\}$, corresponding to $k-3$ values for x, denoted by $x_{n+4},\ldots,x_{n+k}$, as constraint values, and want to minimize the objective function J subject to $f_x(l) = f_x'(l) = 0$, conditions (11)-(13) and condition (14) for $i = n+4,\ldots,n+k$. Let the basis functions be given by

$$k_i(r) = \begin{cases} h_i(r) & \text{if } i = 1,\ldots,n \\ 1 & \text{if } i = n+1 \\ r & \text{if } i = n+2 \\ r^2 & \text{if } i = n+3 \\ b_{x_i}(r) & \text{if } i = n+4,\ldots,n+k \end{cases} \tag{3.16}$$

From the above argument, we know that the solution to the optimization problem

is given by

$$f''_x(r) = \sum_{i=1}^{n+k} \alpha_i k_i(r) \tag{3.17}$$

for some coefficients $\alpha_1, \ldots, \alpha_{n+k}$. If we form a matrix of inner products of these basis terms by

$$A_{ij} = \langle k_i, k_j \rangle \tag{3.18}$$

then the optimization problem from Equations (3.10)–(3.14) can be rewritten as: minimize

$$J = -\sum_{i=1}^{n} \log \left(\sum_{j=1}^{n+k} A_{ij}\alpha_j \right) + \lambda_n \sum_{i=1}^{n+k} \sum_{j=1}^{n+k} A_{ij}\alpha_i\alpha_j \tag{3.19}$$

subject to

$$\sum_{j=1}^{n+3} A_{(n+1)j}\alpha_j = 0 \tag{3.20}$$

$$\sum_{j=1}^{n+3} A_{(n+2)j}\alpha_j = 0 \tag{3.21}$$

$$\sum_{j=1}^{n+k} A_{(n+3)j}\alpha_j = 2 \tag{3.22}$$

$$(\forall i \in \{n+4, \ldots, n+k\}) \sum_{j=1}^{n+k} A_{ij}\alpha_j \geqslant 0 \tag{3.23}$$

A is a positive definite matrix so this is a convex programming problem which guarantees a unique solution. We have implemented this method in the `R` package `pmledecon` [13], which is available from `CRAN`. Our implementation uses the `optim` function from the `stat` package.

3.3 Practical optimization Issues

While the optimization problem in Equations (3.19)–(3.23) looks like a fairly standard multidimensional optimization problem, it is not completely straightforward. There are a number of choices that need to be made for the optimization. Our implementation of the method uses the Nelder-Mead method [83] in the `optim` function from the `stats` package in `R`. This is a simplex-based optimization method.

It has the advantage of being robust. This is particularly useful for our problem because the log-likelihood function is only defined for values of the parameters such that the convolved density is positive, so less robust methods sometimes produce invalid parameter values.

In addition to choice of optimization method, there are a number of particular challenges present in this problem. In this section, we discuss the approach taken to deal with the following challenges: what values to set for l and u; which non-negativity constraints to impose; initial values for parameters; computational singularity; and selection of tuning parameter.

3.3.1 Choosing l and u

In theory, as l and u tend to $-\infty$ and ∞ respectively, we should expect the solution to converge to the solution for support $(-\infty, \infty)$. However, in practice, ensuring the non-negativity constraints all hold becomes more difficult when the support is large and far from the observed data. Also, setting a narrower support reduces the computational difficulty of calculating the necessary inner products A_{ij}.

We initially set the support by first taking the empirical support $Y_{(1)}, Y_{(n)}$ of the observed n data points, and the empirical support $e_{(1)}, e_{(m)}$ of the m pure error sample, where $Y_{(1)}, Y_{(n)}$ and $e_{(1)}, e_{(m)}$ indicate the order statistics. Then, we use $l = Y_{(1)} - e_{(m)}$ and $u = Y_{(n)} - e_{(1)}$ as our initial estimate for the support of f_x. If the error distribution is known, we use the 0.0001 quantile of the distribution as l_e and 0.9999 quantile of the distribution as u_e. Then $e_{(1)}$ and $e_{(m)}$ can be replaced by l_e and u_e for estimating the initial support of f_x.

When we fit the P-MLE for this support, it often happens that $\hat{f}_x$ is negative near the boundaries and positive away from the boundaries. We then use an adaptive method to shrink the boundaries so that $\hat{f}_x$ is positive on (l, u). Suppose our current estimate for the support is (l_k, u_k), and $\hat{f}_x$ is negative on the interval (l_k, w_k) with a minimum at m_k, and on the interval (v_k, u_k) with a minimum at n_k. Then our next estimate for the support is $\left(\frac{l_k+m_k}{2}, \frac{n_k+u_k}{2}\right)$.

3.3.2 Non-negativity constraints

To ensure the non-negativity of density function, a large number of constraints will be involved in the optimization problem. This may cause numerical singularity of matrix A_{ij} and make the computational issues nontrivial. So for simplicity, we choose 30 evenly spaced points to cover the full range on the support of f_x. With the smoothness penalty, the estimated function will often be non-negative on the whole support.

3.3.3 Initial values

Random starting points are commonly used in many optimization problems. However, in our case, random coefficients often do not produce valid density functions to satisfy the non-negative constraints. When the estimated density is negative somewhere, the log-likelihood cannot be computed. As a result, we need a better method to find the appropriate starting points. We take a kernel density estimate for f_Y as our starting point. We use a Gaussian kernel, with bandwidth estimated using Silverman's rule-of-thumb [99], in our implementation. We then project this estimated kernel density of f_Y orthogonally into the space spanned by our basis functions to get the initial coefficients. This can be easily achieved by evaluating the functions of f_Y and all basis functions at a large number of values over the support and solving a regression problem.

Recall that under our adaptive method for setting the boundaries l and u, we sometimes need to repeat the optimization procedure with different values for l and u if the result is not non-negative. In this case, we find the initial solution by projecting the solution of the optimization problem with the previous values l_k and u_k onto the new basis for the reduced support $[l_{k+1}, u_{k+1}]$.

3.3.4 Computational Singularity

A large number of basis functions are translations of the function H. When H is a relatively smooth function, and when there are a lot of observed points, these

translations can become close to be linearly dependent, which can cause numerically unstable solutions, or convergence problems and even sometimes incorrect results. We alleviate this problem using a subsampling approach on the basis. We choose a subsample of size S such that for a sample of S observation points, the translations h_i are not computationally singular. We then solve the optimization problem for this subsample of basis elements (note that we use the whole data set to compute log-likelihood). In our experience, $S = 30$ usually works well. We repeat this for a number of subsamples, and average the results. To improve the reliability, the subsamples are stratified by dividing the support of Y into S intervals and selecting one sample point from each interval. This ensures that the subsample points are well spread-out, reducing the computational instability. We take enough subsamples to ensure that with high probability, the majority of basis elements have been used in at least one subsample. Even with this subsampling, it occasionally still happens that the optimization fails for some subsamples. In these cases, a replacement subsample is drawn.

3.3.5 Selecting λ_n

The smoothness parameter λ_n needs to be selected. This is a tuning parameter, so the usual approach is via cross-validation. We divide the data into a training set and a validation set, use the training set to fit f_x, then evaluate the likelihood on the validation data, where the log-likelihood is $\sum_j \log((\hat{f}_x * \hat{f}_e)(y_j))$. We then choose λ_n to maximize this cross-validated log-likelihood.

In cases where computation time is limited, and we need to fit our method more quickly (for example in the simulations in Section 3.5), we have used the following heuristic approach to quickly select λ_n.

The role of λ_n is to find a reasonable balance between the log-likelihood and the smoothness penalty. For a given sample size, the right balance should be based on controlling the relative size of the two terms. Thus, we take the initial density estimate (before optimization, which uses the initial values calculated from Section 3.3.3, since we need to choose λ_n before we can optimize) and compute the derivatives of the log-likelihood and derivatives of the smoothness penalty terms with respect to all the coefficients. We then set $\lambda_n = R\frac{\sum_{i=1}^{n+k}\left|\frac{\partial l(f_x;y)}{\partial \alpha_i}\right|}{\sum_{i=1}^{n+k}\left|\frac{\partial \langle f_x'', f_x''\rangle}{\partial \alpha_i}\right|}$ for some

constant R. From experience, we found that the following values for R work well.

Sample size	R
30	10^4
100	10^5
300	10^6

3.4 Theory

In this section, we discuss the theoretical large-sample performance of our method. For typical deconvolution problems, theoretical performance is controlled by the identifiability of the deconvolution problem. The Fourier coefficient at each frequency of the convolved density is equal to the product of the Fourier coefficients of the deconvolved density and the error distribution. If one of the Fourier coefficients in the error distribution is zero, the Fourier transformation of the deconvolved density at that frequency is arbitrary. Then the deconvolution problem is not identifiable, since adding the appropriate Fourier term to a "true" distribution does not change its convolution. In practice, distributions rarely have zero Fourier coefficients, so the method is not completely unidentifiable, but when the error distribution is super-smooth, then its Fourier coefficients quickly converge to zero, which makes the problem practically unidentifiable. We resolve this issue by studying the error in the convolved side, rather than the deconvolved side. That is, we can show that $\hat{f}_x * f_e \to f_y$ under general circumstances. This allows us to avoid worrying about ordinary smooth and super-smooth error distributions. We are able to show that P-MLE consistently produces a valid solution to the deconvolution problem. If there are multiple valid solutions, then the choice made by P-MLE may or may not be the truth. We are able to show that

Theorem 1. *Let the true density f_x be twice continuously differentiable, and let the convolved density be $f_y = f_x * f_e$. Let $\hat{f}_x$ be the P-MLE estimate for f_x and $\hat{f}_y = \hat{f}_x * f_e$. If the smoothness penalty parameter for each estimate is given by $\lambda_n = C_1 n^{\frac{7}{8}} \log(n)^{\frac{1}{8}} \sqrt{|u-l|}$, for a certain constant C_1, then almost surely, for all sufficiently large n, $\|\hat{f}_y - f_y\|_\infty < C_2 n^{-\frac{1}{32}} |u-l|^{\frac{1}{8}} \log(n)^{\frac{1}{32}}$ for some constant C_2.*

The constants C_1 and C_2 are given by

$$C_1 = \frac{2^{\frac{31}{20}} 3^{-\frac{1}{10}} 5^{\frac{3}{20}}}{\psi(f_x)^{\frac{4}{5}} \psi(f_y)^{\frac{1}{10}}}$$

$$C_2 = \left(1 + \sqrt{1 + \frac{3^{\frac{2}{5}}}{16}}\right)^{\frac{1}{4}} 2^{\frac{179}{80}} 3^{\frac{9}{40}} 5^{\frac{27}{80}} \psi(f_x)^{\frac{1}{4}} \psi(f_y)^{-\frac{1}{40}}$$

In this section, we change the notation slightly to avoid excessive subscripts.

For a twice differentiable function f with support S, let $\psi(f) = \int_S (f''(x))^2 \, dx$ be the smoothness penalty. Our estimator is the function $\hat{f}_n$ that maximizes the penalized log-likelihood.

$$\sum_{i=1}^{n} \log\left(\left(\hat{f}_n * f_e\right)(x_i)\right) - \lambda_n \psi\left(\hat{f}_n\right)$$

We want to prove consistency of $\hat{f}_n$. That is, more formally, suppose we have a sequence of i.i.d. observations $x_1, x_2, \ldots$ from a distribution with density function $f * f_e$, where f is twice continuously differentiable with finite smoothness. We use the notation f_e for the density of the error distribution, but this is a slight abuse of notation, because the proof works equally well for non-continuous error distribution. Let $(\lambda_n)_{n=1,\ldots}$ be a chosen sequence of penalty terms and $S_1 \subseteq S_2 \subseteq \cdots$ be a chosen increasing sequence of measurable subsets of $\mathbb{R}$ with union $\mathbb{R}$. Let $\hat{f}_n$ be the twice continuously differentiable function with support S_n that maximizes the penalized log-likelihood function

$$\sum_{i=1}^{n} \log((\hat{f}_n * f_e)(x_i)) - \lambda_n \psi(\hat{f}_n)$$

for the first n observations. We want to prove that $\hat{f}_n$ converges to f. Because of the difficulties that can arise when f_e is supersmooth, we will show that $\hat{f}_n * f_e$ converges to $f * f_e$. That is, we will show that for some $\gamma, \xi > 0$, and some ζ, with probability 1, there is some N such that for all $n > N$ we have $\|(\hat{f}_n - f) * f_e\|_\infty < \gamma n^{-\xi} |S_n|^\zeta$.

In addition to the smoothness assumptions needed to state the problem, we will

assume that $g = f * f_e$ has only a finite number M of local maxima. This eliminates a number of pathological distributions without eliminating any distributions of real interest. It simplifies the proof, but we believe that the method is still consistent without this assumption. This assumption is needed for Proposition 1, which allows a relatively simple proof of Proposition 5. However, we believe that a version of Proposition 5 (possibly with the lower bound changed slightly) is true without the need for this assumption.

In our proof, we will change between working on the noiseless scale of the underlying distribution we are trying to estimate, and working on the convolved scale, with the noise distribution added. We will consistently use f to represent densities on the noiseless scale, and g to represent densities on the convolved scale so for example $g = f * f_e$ refers to the convolved density of the true distribution. Similar notation will be used throughout. For example $\hat{g}_n = \hat{f}_n * f_e$ is the convolution of the penalized MLE estimate with the error distribution. For this proof, we will assume that f_e is known.

We will use the notation g for densities that are related to the convolved densities, even if they do not themselves arise as convolutions with f_e. For example, in the proof, we will introduce a density $\tilde{g}_n$, where we use the notation g because it is an estimator for the convolved density $g = f * f_e$, but the density $\tilde{g}_n$ does not necessarily arise as a convolution of any density function with f_e.

To prove consistency, we will first note that the Dvoretzky-Kiefer-Wolfowitz [29] inequality gives that for any i.i.d. sample from a distribution with c.d.f. G, the c.d.f. G_n of the empirical distribution satisfies

$$P\left(\sqrt{n}\|G_n - G\|_\infty > \sqrt{\log(n)}\right) < Cn^{-2}$$

for some positive constant C (later, Massart [74] showed that $C = 1$), so according to the Borel-Cantelli Lemma [30], with probability 1, there is some N such that for all $n > N$ we have $\|G_n - G\|_\infty \leqslant \sqrt{\frac{\log(n)}{n}}$. In our case, we have that G is the c.d.f. for $g = f * f_e$, and G_n is the empirical c.d.f. of the observed sample.

Our approach for proving consistency will follow this outline:

1. Construct a sequence of estimators $\tilde{g}_n(x)$ for g (we use kernel density estimators)

such that the following conditions hold whenever $\|G_n - G\|_\infty \leqslant \sqrt{\frac{\log(n)}{n}}$.

(a) $\|\tilde{g}_n - g\|_\infty < C_3 \frac{\psi(g)^{\frac{1}{5}} \log(n)^{\frac{1}{4}}}{n^{\frac{1}{4}}}$ for some constant C_3.

(b) For any density function g_L^* that is Lipschitz with constant L and has support S_n, and has $g_L^*(x_i) \geqslant \epsilon$ for $i = 1, \ldots, n$,

$$\left| \frac{1}{n} \sum_{i=1}^{n} \log(g_L^*(x_i)) - \int_{S_n} \tilde{g}_n(x) \log(g_L^*(x))\, dx \right| < \frac{\log(n)^{\frac{1}{4}} L}{C_3 \psi(g)^{\frac{1}{5}} n^{\frac{1}{4}} \epsilon}$$

Intuitively, this is saying that the empirical mean of $\log(g_L^*(x))$ is close to the mean over the distribution $\tilde{g}$.

2. Construct a sequence of distributions f_n^* that converge to f, (based partly on the data, and partly on the true function f) with the following properties:

(a) $\psi(f_n^*) \leqslant 2\psi(f)$

(b) Whenever, $\|G_n - G\| < \sqrt{\frac{\log(n)}{n}}$, and n is sufficiently large, $\frac{1}{n} \sum_{i=1}^{n} \log(g_n^*(x_i)) > -H(g) - 4Mn^{-\frac{1}{2}} \log(n)^{\frac{3}{2}}$, where $H(g)$ is the entropy $-\int_{-\infty}^{\infty} g(x) \log(g(x))$.

3. Since the penalized likelihood of $\hat{f}_n$ is bounded below by the penalized likelihood of f^*, using property 1(b), we can bound the likelihood of $\hat{g}_n$, which allows us to prove $\psi(\hat{f}_n) < 3\psi(f)$ for all sufficiently large n.

4. Having found a uniform Lipschitz constant for all $\hat{g}_n$, we will bound $\sum_{i=1}^{n} \log\left(\hat{g}_n(x_i)\right)$ as a function of $\|\hat{g}_n - g\|_\infty$, and deduce that $\|\hat{g}_n - g\|_\infty \to 0$ almost surely.

3.4.1 Preliminary Results

We begin with some general inequalities and results about the smoothness penalty that will be necessary for proving our main results.

Smoothness and L^∞ norms

Lemma 1. *If g is a density function, then $\psi(f * g) \leqslant \psi(f)$.*

Proof.

$$\begin{aligned}
\psi(f*g) &= \int_{-\infty}^{\infty} ((f*g)''(x))^2\,dx \\
&= \int_{-\infty}^{\infty} \left(\frac{d^2}{dx^2}\int_{-\infty}^{\infty} g(t)f(x-t)\,dt\right)^2 dx \\
&= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} g(t)\frac{d^2 f(x-t)}{dx^2}\,dt\right)^2 dx \\
&= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} g(t)f''(x-t)\,dt\right)\left(\int_{-\infty}^{\infty} g(s)f''(x-s)\,ds\right) dx \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(t)g(s)\int_{-\infty}^{\infty} f''(x-t)f''(x-s)\,dx\,ds\,dt \\
&\leqslant \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(t)g(s)\int_{-\infty}^{\infty} f''(x-t)^2\,dx\,ds\,dt \\
&= \psi(f)
\end{aligned}$$

□

Remark 3.4.1. *It is straightforward to modify this proof in the case where the error distribution is not continuous.*

Lemma 2. *For any twice continuously differentiable density function g, $\|g\|_\infty \leqslant \left(\frac{5^4\psi(g)}{3\times 2^{12}}\right)^{\frac{1}{5}}$*

Proof. Suppose g attains its maximum at $x = 0$. (Since $\|g\|_\infty$ and $\psi(g)$ are translation invariant, there is no loss of generality). We first prove the result under the assumption that g is symmetric about 0.

For any constant $c > 0$, the function $h(x) = cg(cx)$ satisfies $\int_{-\infty}^{\infty} h(x)\,dx = \int_{-\infty}^{\infty} g(x)\,dx$, so h is a twice continuously differentiable density function. We have that $\|h\|_\infty = h(0) = cg(0) = c\|g\|_\infty$, and $h''(x) = c^3 g''(cx)$, so $\psi(h) = c^5\psi(g)$. This means that $\|g\|_\infty\psi(g)^{-\frac{1}{5}} = \|h\|_\infty\psi(h)^{-\frac{1}{5}}$. Thus, in trying to maximize $\|g\|_\infty\psi(g)^{-\frac{1}{5}}$, we may without loss of generality assume that $\int_0^1 g(x)\,dx = 0.4g(0)$.

Let $m = \sup g(x) = g(0)$. Let $a = g(1)$, $b = g'(1)$. We have $\int_0^1 g(x)\,dx = 0.4m$, $g(0) = m$ and $g'(0) = 0$. Now on the Hilbert space $L^2([0,1])$, we have

$$\langle g'', x^2 \rangle = \int_0^1 x^2 g''(x)\, dx = [x^2 g'(x)]_0^1 - [2xg(x)]_0^1 + 2\int_0^1 g(x)dx = b - 2a + 0.8m \tag{3.24}$$

$$\langle g'', x \rangle = [xg'(x)]_0^1 - \int_0^1 g'(x)dx = b + m - a \tag{3.25}$$

$$\langle g'', 1 \rangle = b \tag{3.26}$$

$$\langle x^\alpha, x^\beta \rangle = \frac{1}{\alpha + \beta + 1} \tag{3.27}$$

Subject to conditions 3.24-3.26, in the Hilbert space L^2 ($[0, 1]$), we can decompose g'' as a linear combination of $1, x$ and x^2 plus a function orthogonal to all these elements. If we let g_0 be the linear combination of these elements, and let $g_\perp$ be the orthogonal part, because of the orthogonality, $\langle g'', g'' \rangle = \langle g_0, g_0 \rangle + \langle g_\perp, g_\perp \rangle$ is clearly minimized when $g_\perp = 0$, so we have shown that $\|g''\|_2$ is minimized by setting g'' to be a linear combination $\beta_0 + \beta_1 x + \beta_2 x^2$. Let $v = (\beta_0, \beta_1, \beta_2)^T$ and let

$$M = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

be given by $M_{ij} = \langle x^{i-1}, x^{j-1} \rangle$ and 3.27. Then our equations become $Mv = w$, where $w = (b, b + m - a, b - 2a + 0.8m)^T$. We have

$$M^{-1} = \begin{pmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{pmatrix}$$

Thus $\|g''\|_2^2 \geqslant v^T M v = w^T M^{-1} M M^{-1} w = w^T M^{-1} w$. Let $t = (m, a, b)^T$. Then $w = Bt$ where

$$B = \begin{pmatrix} 0 & 0 & 1 \\ 1 & -1 & 1 \\ 0.8 & -2 & 1 \end{pmatrix}$$

This gives

$$\|g''\|_2^2 \geqslant t^T B^T M^{-1} B t = (m, a, b) \begin{pmatrix} 19.2 & 24 & 0 \\ 24 & 192 & -36 \\ 0 & -36 & 9 \end{pmatrix} \begin{pmatrix} m \\ a \\ b \end{pmatrix}$$

For fixed a, $\|g''\|_2$ is minimized when $b = 4a$. In this case $t^T B^T M^{-1} B t$ is an increasing function of a for all non-negative a. Since $a \geqslant 0$, $t^T B^T M^{-1} B t$ is minimized when $a = b = 0$. Thus $\|g''\|_2^2 \geqslant 19.2m^2$.

Now since g is an even function, we have $\int_{-1}^{1} g(x)\,dx = 0.8m \leqslant 1$. It follows that $m \leqslant 1.25$ and $\psi(g) \geqslant 2t^T B^T M^{-1} B t \geqslant 38.4m^2 \geqslant \frac{38.4m^5}{1.25^3} = \frac{3\times 2^{12}}{5^4} m^5$. Therefore, $\|g\|_\infty \psi(g)^{-\frac{1}{5}} \leqslant \left(\frac{5^4}{3\times 2^{12}}\right)^{\frac{1}{5}}$.

We have proved the result when g is symmetric about 0. Suppose g is not symmetric about 0. Let $s_+(x) = \frac{g(|x|)}{c_+}$ and $s_-(x) = \frac{g(-|x|)}{c_-}$ be the symmetric density functions obtained by reflecting the positive and negative parts of g respectively in the y axis, and rescaling. The positive constants c_+ and c_- are chosen so that $\int_{-\infty}^{\infty} s_+(x)\,dx = 1$ and $\int_{-\infty}^{\infty} s_-(x)\,dx = 1$. That is, $c_- = 2\int_{-\infty}^{0} g(x)\,dx$ and $c_+ = 2\int_0^{\infty} g(x)\,dx$. It is easy to see that s_+ and s_- are twice continuously differentiable density functions (because $g'(0) = 0$), and are symmetric, so applying the result for symmetric functions, we get $\psi(s_+) \geqslant \frac{3\times 2^{12}}{5^4} \frac{\|g\|_\infty^5}{{c_+}^5}$ and $\psi(s_-) \geqslant \frac{3\times 2^{12}}{5^4} \frac{\|g\|_\infty^5}{{c_-}^5}$. Also, we have $\psi(g) = \frac{1}{2}\left({c_-}^2\psi(s_-) + {c_+}^2\psi(s_+)\right) \geqslant \frac{3\times 2^{11}}{5^4}\|g\|_\infty^5 \left(\frac{1}{{c_-}^3} + \frac{1}{{c_+}^3}\right)$. Since $\int_{-\infty}^{\infty} g(x)\,dx = 1$, we have $c_- + c_+ = 2$, so ${c_-}^{-3} + {c_+}^{-3}$ is minimized when $c_- = c_+ = 1$. Thus, the result holds for any g. □

Corollary 1. *For any density function f and any distribution, f_e, $\|f * f_e\|_\infty \leqslant \left(\frac{5^4\psi(f)}{3\times 2^{12}}\right)^{\frac{1}{5}}$*

Proof. This is immediate from Lemma 1 and Lemma 2 □

Lemma 3. *If a density function $g : \mathbb{R} \to \mathbb{R}_{\geqslant 0}$ is twice continuously differentiable and has finite support, then $\|g'\|_\infty \leqslant \left(\frac{125\psi(g)^2}{144}\right)^{\frac{1}{5}}$*

Proof. Translating if necessary, we may assume that $|g'(x)|$ is maximized by $x = 0$. By reflecting in the line $x = 0$ if necessary, we may also assume $g'(0) > 0$. We may also rescale (that is, replace g by $h(x) = cg(cx)$ as in the proof of Proposition 2: it

is easy to check that $\|g'\|_\infty \psi(g)^{-\frac{2}{5}} = \|h'\|_\infty \psi(h)^{-\frac{2}{5}}$) so that $g(1) = g'(1) = 0$. We will prove the result by finding a suitable function h that minimizes $\psi(h)$ subject to the constraints

(i) $\|h'\|_\infty = h'(0) = s > 0$

(ii) $\int_{-\infty}^{1} h(x)\,dx \leqslant 1$,

(iii) h is twice continuously differentiable everywhere except for one value $x \leqslant 0$ at which $h(x) = 0$.

Since these conditions on h are slightly weaker than the conditions for g in the proposition, any g satisfying the conditions of the proposition also satisfies the conditions for h. It follows that $\psi(g)^{-\frac{2}{5}}\|g'\|_\infty \leqslant \psi(h)^{-\frac{2}{5}}\|h'\|_\infty$, so if this h satisfies the inequality in the proposition, the result will be proved.

Let

$$k(x) = \begin{cases} 0 & \text{if } x < -\frac{h(0)}{h'(0)} \\ h'(0)x + h(0) & \text{if } -\frac{h(0)}{h'(0)} \leqslant x < 0 \\ h(x) & \text{if } x \geqslant 0 \end{cases}$$

That is, k is a linear extension of the restriction of h to positive real numbers. By inspection, $\|k'\|_\infty \leqslant h'(0) = s$, so k satisfies Condition (i); also by the mean value theorem, $k(x) \leqslant h(x)$ for all x, so $\int_0^1 k(x)\,dx \leqslant \int_0^1 h(x)\,dx \leqslant 1$, so k satisfies Condition (ii). Finally, Condition (iii) is obvious on all intervals, $\left(-\infty, -\frac{h(0)}{h'(0)}\right]$, $\left(-\frac{h(0)}{h'(0)}, 0\right]$ and $(0, \infty)$, so the condition only needs to be tested at 0. At 0, we obviously have $k'(0) = h'(0) = s$, and $k''(0) = h''(0) = 0$ because $h'(x)$ attains its maximum value at $x = 0$.

Since k satisfies the same conditions as h, we have $\psi(k) \geqslant \psi(h)$. On the other hand $|k''(x)| \leqslant |h''(x)|$ for all x, so $\psi(k) \leqslant \psi(h)$. Therefore, $k\left(x + \frac{h(0)}{h'(0)}\right)$ is an alternative minimizer of $\psi(k)$ subject to Conditions (i)–(iii), and it satisfies the additional condition that $k(0) = 0$. Thus, we may assume w.l.o.g. that h' is maximized by a value x such that $h(x) = 0$. By translating, we may assume $x = 0$.

Now since $h(0) = 0$, $h'(0) = s$, $h(1) = 0$, $h'(1) = 0$ and $\int_0^1 h(x)\,dx = r \leqslant 1$, we have

$$\langle h'', 1\rangle = \int_0^1 h''(x)\,dx = h'(1) - h'(0) = -s$$
$$\langle h'', x\rangle = [h'(x)x]_0^1 - \int_0^1 h'(x)\,dx = h'(1) - (h(1) - h(0)) = 0$$
$$\langle h'', x^2\rangle = [h'(x)x^2]_0^1 - 2[h(x)x]_0^1 + 2\int_0^1 h(x)\,dx = 2r \leqslant 2$$

It is easy to see that subject to these conditions, $\psi(h)$ is minimized by setting h'' as a linear combination of 1, x and x^2. Let $h''(x) = a + bx + cx^2$. Then we have

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} -s \\ 0 \\ 2r \end{pmatrix}$$

which gives

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{pmatrix} \begin{pmatrix} -s \\ 0 \\ 2r \end{pmatrix}$$

and $\psi(h) = 9s^2 - 120sr + 720r^2$.

Recall that by rescaling, we are aiming to minimize

$$\psi(h)\|h'\|_\infty^{-\frac{5}{2}} = 9s^{-\frac{1}{2}} - 120rs^{-\frac{3}{2}} + 720r^2 s^{-\frac{5}{2}}$$

It is easy to see that this is minimized when s is a solution to

$$-\frac{1}{2}9s^{-\frac{3}{2}} + 120\frac{3}{2}rs^{-\frac{5}{2}} - 720\frac{5}{2}r^2 s^{-\frac{7}{2}} = 0$$
$$-s^2 + 40rs - 400r^2 = 0$$
$$s = 20r$$

For $s = 20r$, we have

$$\frac{\psi(h)^2}{s^5} = \frac{(3600 - 2400 + 720)^2 r^2}{20^5 r^5} = \frac{1920^2}{20^5 r^3} = \frac{144}{125}r^{-3}$$

Since $r \leqslant 1$, we deduce $\frac{\psi(h)^2}{\|h'\|_\infty^5} \leqslant \frac{144}{125}$.

□

General Inequalities and probability theory

We start by showing that for any random variable, there is at least one point at which it has positive density.

Lemma 4. *For any random variable Z, there is some b, $\delta > 0$ and $\epsilon > 0$ such that for any $t < \delta$, $P(|Z - b| < t) > \epsilon t$.*

Proof. Pick a finite interval $[l_0, u_0]$ such that E has positive probability p on the interval $[l_0, u_0]$, and let $d = \frac{p}{u_0 - l_0}$. Let m_0 be the midpoint of $[l_0, u_0]$, For the intervals $[l_0, m_0]$ and $[m_0, u_0]$, we have $P(Z \in [l_0, m_0]) + P(Z \in [m_0, u_0]) = p$, so one of $P(Z \in [l_0, m_0])$ and $P(Z \in [m_0, u_0])$ must be at least $\frac{p}{2}$. Without loss of generality, suppose $P(Z \in [l_0, m_0]) \geqslant \frac{p}{2}$. We now let $l_1 = l_0$ and $u_1 = m_0$ to create a new interval whose average density is at least $\frac{p}{u_0 - l_0}$. Repeating this interval bisection, we get a decreasing sequence of intervals $[l_k, u_k]$ whose intersection is a single point b. We will show that this b with $\delta = (u_0 - l_0)$ satisfies the result. To do this, for any $t < \delta$, note that there is some k such that $[l_k, u_k] \subseteq [b - t, b + t]$ and for this choise, $u_k - l_k > \frac{t}{2}$. It will follow that $P(|Z - b| < t) \geqslant P(Z \in [l_k, u_k]) \geqslant d(u_k - l_k) \geqslant d\frac{t}{2}$. For $t \geqslant \max(u_0 - b, b - l_0)$, $k = 0$ obviously satisfies the conditions. Otherwise, since $l_k \to b$ and $u_k \to b$, there must be some smallest k such that $[l_k, u_k] \subseteq [b - t, b + t]$. By minimality of this k, we have that either $l_{k-1} < b - t$ or $u_{k-1} > b + t$. In either case, $u_k - l_k = \frac{u_{k-1} - l_{k-1}}{2} > \frac{t}{2}$. □

Proposition 1. *Let g be a twice continuously differentiable function with support $[l, u]$ and with at most M local maxima. Let $r > 0$ and set $b(x) = g(x) \vee r$. Then*

$$\int_l^u \left| \frac{b'(x)}{b(x)} \right| dx \leqslant \frac{2M}{5} \log \left(\frac{5^4 \psi(g)}{3 \times 2^{12} r^5} \right)$$

Proof. Let the local maxima of $g(x)$ be at $a_1 < a_3 < \ldots < a_{2M-1}$, and let the local

minima be at $a_2 < a_4 < \ldots < a_{2M-2}$, and let $a_0 = l, a_{2M} = u$. We have

$$\begin{aligned}\int_l^u \left|\frac{b'(x)}{b(x)}\right| dx &= \sum_{i=0}^{M-1} \left(\int_{a_{2i}}^{a_{2i+1}} \frac{b'(x)}{b(x)}\, dx - \int_{a_{2i+1}}^{a_{2i+2}} \frac{b'(x)}{b(x)}\, dx\right) \\ &= \sum_{i=0}^{M-1} \left(2\log(b(a_{2i+1})) - \log(b(a_{2i})) - \log(b(a_{2i+2}))\right) \\ &\leqslant 2\sum_{i=0}^{M-1} \log\left(\frac{b(a_{2i+1})}{r}\right)\end{aligned}$$

By Lemma 2, we have $\|g\|_\infty \leqslant \left(\frac{5^4\psi(g)}{3\times 2^{12}}\right)^{\frac{1}{5}}$, so $\log(b(a_{2i+1})) \leqslant \log\left(\left(\frac{5^4\psi(g)}{3\times 2^{12}}\right)^{\frac{1}{5}}\right)$ This gives

$$\int_l^u \left|\frac{b'(x)}{b(x)}\right| dx \leqslant \frac{2M}{5} \log\left(\frac{5^4\psi(g)}{3\times 2^{12} r^5}\right)$$

□

Kulback-Leibler Divergence

Lemma 5. *If g and h are density functions and $h(x) > g(x) + \epsilon$ for all $x \in [x_0 - \delta, x_0 + \delta]$, where $\epsilon > 0$ and $\delta > 0$, then the Kulback-Leibler divergence is bounded below by $\int g(x) \log\left(\frac{g(x)}{h(x)}\right) dx > 2\delta^2\epsilon^2$*

Proof. Let $h(x) = k(x) + l(x)$, where

$$l(x) = \begin{cases} \epsilon & \text{if } |x - x_0| < \delta \\ 0 & \text{otherwise} \end{cases}$$

Since $h(x) > g(x) + \epsilon$ for all $x \in [x_0 - \delta, x_0 + \delta]$, it follows that $k(x) > g(x)$ for all $x \in [x_0 - \delta, x_0 + \delta]$. Now $\log(h(x)) = \log(k(x)) + \log\left(1 + \frac{l(x)}{k(x)}\right)$, so

$$\begin{aligned}\int g(x)\log(h(x))\, dx &= \int g(x)\log(k(x))\, dx + \int_{x_0-\delta}^{x_0+\delta} g(x)\log\left(1 + \frac{\epsilon}{k(x)}\right) dx \\ &\leqslant \int g(x)\log(k(x))\, dx + \int_{x_0-\delta}^{x_0+\delta} g(x)\log\left(1 + \frac{\epsilon}{g(x)}\right) dx\end{aligned}$$

To bound the first integral, note that $\frac{k(x)}{1-2\delta\epsilon}$ is a density function, so

$$\begin{aligned}\int g(x)\log(k(x))\,dx &= \int g(x)\left(\log\left(\frac{k(x)}{1-2\delta\epsilon}\right)+\log(1-2\delta\epsilon)\right)\,dx\\ &\leqslant \int g(x)\left(\log\left(g(x)\right)\,dx+\log(1-2\delta\epsilon)\right.\end{aligned}$$

Thus

$$\begin{aligned}\int g(x)\log(h(x))\,dx &\leqslant \int g(x)\log\left(g(x)\right)\,dx+\log\left(1-2\delta\epsilon\right)+\int_{x_0-\delta}^{x_0+\delta} g(x)\log\left(1+\frac{\epsilon}{g(x)}\right)\,dx\\ &\leqslant \int g(x)\log\left(g(x)\right)\,dx+\log\left(1-2\delta\epsilon\right)+2\delta\epsilon\\ &\leqslant \int g(x)\log\left(g(x)\right)\,dx-2\delta^2\epsilon^2\end{aligned}$$

□

Lemma 6. *If g and h are density functions and $h(x) < g(x) - \epsilon$ for all $x \in [x_0 - \delta, x_0+\delta]$, then the Kulback-Leibler divergence is bounded below by $\int g(x)\log\left(\frac{g(x)}{h(x)}\right)\,dx > 2\delta^2\epsilon^2 - \frac{8}{3}\delta^3\epsilon^3$*

Proof. Let $h(x) = k(x) - l(x)$, where

$$l(x) = \begin{cases}\epsilon & \text{if } |x-x_0|<\delta\\ 0 & \text{otherwise}\end{cases}$$

Since $h(x) < g(x) - \epsilon$ for all $x \in [x_0 - \delta, x_0 + \delta]$, it follows that $k(x) < g(x)$ for all $x \in [x_0 - \delta, x_0 + \delta]$. Now $\log(h(x)) = \log(k(x)) + \log\left(1 - \frac{l(x)}{k(x)}\right)$, so

$$\begin{aligned}\int g(x)\log(h(x))\,dx &= \int g(x)\log(k(x))\,dx+\int_{x_0-\delta}^{x_0+\delta} g(x)\log\left(1-\frac{\epsilon}{k(x)}\right)\,dx\\ &\leqslant \int g(x)\log(k(x))\,dx+\int_{x_0-\delta}^{x_0+\delta} g(x)\log\left(1-\frac{\epsilon}{g(x)}\right)\,dx\end{aligned}$$

To bound the first integral, note that $\frac{k(x)}{1+2\delta\epsilon}$ is a density function, so

$$\begin{aligned}\int g(x)\log(k(x))\,dx &= \int g(x)\left(\log\left(\frac{k(x)}{1+2\delta\epsilon}\right)+\log(1+2\delta\epsilon)\right)\,dx\\ &\leqslant \int g(x)\left(\log\left(g(x)\right)\,dx+\log(1+2\delta\epsilon)\right.\end{aligned}$$

Thus

$$\begin{aligned}\int g(x)\log(h(x))\,dx &\leqslant \int g(x)\log\left(g(x)\right)\,dx+\log\left(1+2\delta\epsilon\right)+\int_{x_0-\delta}^{x_0+\delta} g(x)\log\left(1-\frac{\epsilon}{g(x)}\right)\,dx\\ &\leqslant \int g(x)\log\left(g(x)\right)\,dx+\log\left(1+2\delta\epsilon\right)-2\delta\epsilon\\ &\leqslant \int g(x)\log\left(g(x)\right)\,dx-2\delta^2\epsilon^2+\frac{8}{3}\delta^3\epsilon^3\end{aligned}$$

□

Corollary 2. *If density functions $g^!$ and g are both Lipschitz with constant L, and $\|g^!-g\|_\infty>\rho$, where $\rho<\sqrt{2L}$, then the Kulback-Leibler divergence is bounded below by $\int g(x)\log\left(\frac{g(x)}{g^!(x)}\right)\,dx>\frac{\rho^4}{48L^2}$*

Proof. Since $\|g^!-g\|_\infty>\rho$, there is some x_0 such that $|g^!(x_0)-g(x_0)|>\rho$. By the Lipschitz condition, if we let $\delta=\frac{\rho}{4L}$, then for $x\in[x_0-\delta,x_0+\delta]$, we have

$$\begin{aligned}\rho<|(g^!(x_0)-g(x_0))| &= |(g^!(x_0)-g^!(x))+g^!(x)-g(x)+(g(x)-g(x_0))|\\ &\leqslant L|x-x_0|+|g^!(x)-g(x)|+L|x-x_0|\end{aligned}$$

so $|g^!(x)-g(x)|>\rho-2L\delta=\frac{\rho}{2}$. By Lemma 5 and Lemma 6, we therefore have

$$\int g(x)\log\left(\frac{g(x)}{g^!(x)}\right)\,dx>2\left(\frac{\rho^2}{8L}\right)^2-\frac{8}{3}\left(\frac{\rho^2}{8L}\right)^3$$

It is easy to see that for $\rho<\sqrt{2L}$, this is bounded below by $\frac{\rho^4}{48L^2}$. □

3.4.2 Constructing $\tilde{g}_n$

The function $\tilde{g}_n$ is constructed as a kernel density estimate of g with uniform kernel and bandwidth $\delta_n = \left(\frac{\log(n)}{n}\right)^{\frac{1}{4}} \left(\frac{3}{\psi(g)}\right)^{\frac{1}{5}} 2^{0.9} 5^{-0.3}$. That is

$$\tilde{g}_n(x) = \frac{1}{2n\delta_n} \left|\left\{i \in \{1, \ldots, n\} \middle| |x_i - x| < \delta_n\right\}\right| = \frac{1}{2\delta_n} \left(G_n(x + \delta_n) - G_n(x - \delta_n)\right)$$

We want to show that $\tilde{g}_n$ has properties (a) and (b), which are

(a) $\|\tilde{g}_n - g\|_\infty < C_3 \frac{\psi(g)^{\frac{1}{5}} \log(n)^{\frac{1}{4}}}{n^{\frac{1}{4}}}$ for some constant C_3.

(b) For any density function g_L^* that is Lipschitz with constant L and has support S_n, and has $g_L^*(x_i) \geqslant \epsilon$ for $i = 1, \ldots, n$,

$$\left| \frac{1}{n} \sum_{i=1}^{n} \log(g_L^*(x_i)) - \int_{S_n} \tilde{g}_n(x) \log(g_L^*(x))\, dx \right| < \frac{\log(n)^{\frac{1}{4}} L}{C_3 \psi(g)^{\frac{1}{5}} n^{\frac{1}{4}} \epsilon}$$

Intuitively, this is saying that the empirical mean of $\log(g_L^*(x))$ is close to the mean over the distribution $\tilde{g}$.

Proposition 2. *If* $\|G_n - G\|_\infty \leqslant \sqrt{\frac{\log(n)}{n}}$, *then* $\|\tilde{g}_n - g\|_\infty < C_3 \psi(g)^{\frac{1}{5}} \left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}$ *where* $C_3 = 2^{0.1} 3^{-0.2} 5^{0.3}$.

Proof. By Lemma 3, g is Lipschitz with constant $L = \frac{5^{\frac{3}{5}} \psi(g)^{\frac{2}{5}}}{3^{\frac{2}{5}} 2^{\frac{4}{5}}}$. Since $\tilde{g}_n(x) =$

$\frac{1}{2\delta_n}\big(G_n(x+\delta_n) - G_n(x-\delta_n)\big)$, For any x,

$$\begin{aligned}
|\tilde{g}_n(x) - g(x)| &= \left|\frac{1}{2\delta_n}\big(G_n(x+\delta_n) - G_n(x-\delta_n)\big) - g(x)\right| \\
&\leqslant \left|\frac{1}{2\delta_n}\big((G_n(x+\delta_n) - G(x+\delta_n)) - (G_n(x-\delta_n) - G(x-\delta_n))\big)\right| \\
&\qquad + \left|\frac{1}{2\delta_n}\big((G(x+\delta_n)) - G(x-\delta_n)\big) - g(x)\right| \\
&\leqslant \frac{\|G_n - G\|_\infty}{\delta_n} + \left|\frac{1}{2\delta_n}\int_{-\delta_n}^{\delta_n} g(x+t)\,dt - g(x)\right| \\
&\leqslant \frac{1}{\delta_n}\sqrt{\frac{\log(n)}{n}} + \left|\frac{1}{2\delta_n}\int_{-\delta_n}^{\delta_n} (g(x+t) - g(x))\,dt\right| \\
&\leqslant \frac{1}{\delta_n}\sqrt{\frac{\log(n)}{n}} + \frac{1}{2\delta_n}\int_{-\delta_n}^{\delta_n} L|t|\,dt \\
&= \frac{1}{\delta_n}\sqrt{\frac{\log(n)}{n}} + \frac{L\delta_n}{2}
\end{aligned}$$

In particular, since $\delta_n = \left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}\left(\frac{3}{\psi(g)}\right)^{\frac{1}{5}} 2^{0.9}5^{-0.3}$, we get

$$|\tilde{g}_n(x) - g(x)| \leqslant 2^{0.1}5^{0.3}3^{-0.2}\psi(g)^{0.2}\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}$$

□

Proposition 3. *If $\|G_n - G\|_\infty \leqslant \sqrt{\frac{\log(n)}{n}}$, then for any function g_L^* that is Lipschitz with constant L, has support S_n, and has $g_L^*(x_i) \geqslant \epsilon > 0$,*

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_L^*(x_i)) - \int_{S_n}\tilde{g}_n(x)\log(g_L^*(x))\,dx < {C_3}^{-1}\psi(g)^{-0.2}\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}\epsilon^{-1}L$$

Proof. We have

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\log(g_L^*(x_i)) - \int_{S_n}\tilde{g}_n(x)\log(g_L^*(x))\,dx \\
&\qquad = \frac{1}{n}\sum_{i=1}^{n}\log(g_L^*(x_i)) - \frac{1}{2\delta_n}\int_{S_n}\left(\frac{1}{n}\sum_{i=1}^{n}1_{\{|x_i-x|<\delta_n\}}\right)\log(g_L^*(x))\,dx \\
&\qquad = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2\delta_n}\int_{x_i-\delta_n}^{x_i+\delta_n}\left(\log(g_L^*(x_i)) - \log(g_L^*(x))\right)\,dx \\
&\qquad \leqslant \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2\delta_n}\int_{x_i-\delta_n}^{x_i+\delta_n}\log\left(1+\frac{L|x-x_i|}{g_L^*(x)}\right)\,dx \\
&\qquad \leqslant \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2\delta_n}\int_{x_i-\delta_n}^{x_i+\delta_n}\frac{L|x-x_i|}{g_L^*(x)}\,dx \\
&\qquad \leqslant \frac{\delta_n L}{2\epsilon}
\end{aligned}$$

□

3.4.3 Constructing f_n^*

Controlling the P-MLE is based on finding a lower bound for the penalized maximum likelihood. We do this by exhibiting a good candidate density function that gives high penalized likelihood. A first attempt is the true density function f. However, f can have low likelihood because of a small number of outliers with very low likelihood. To correct for this, we will take a mixture of the true likelihood f and a data-driven likelihood $f_n^\dagger$ that guarantees $g_n^\dagger(x_i) > \epsilon_n$ for $i = 1, \ldots, n$.

The easiest way to get an estimator with likelihood bounded below on observed data points is with a kernel density estimate. Since we are trying to find a deconvolved density estimator, this approach does not work — there is no guarantee that the kernel density estimator arises as a convolution with the error distribution. However, by Lemma 4, there is a basepoint b for the error distribution, and an $\epsilon > 0$, such that for all $\delta < c$, $P(|E - b| < \delta) > \delta\epsilon$. We let $f^\dagger$ be a kernel estimate from $x_i - b$. Using the condition from Lemma 4, we get that if there is an interval about x_0 such that $f^\dagger(x) > a$ for all $x \in [x_0 - \delta, x_0 + \delta]$, then $g^\dagger(x_0 + b) > \epsilon\delta a$.

We want to control the smoothness penalty $\psi(f^\dagger)$. To do this, we set the kernel

$$k_\delta(x) = \delta^{-4} \begin{cases} 3\delta(x+\delta)^2 - 2(x+\delta)^3 & \text{if } -\delta < x \leqslant 0 \\ 3\delta(x-\delta)^2 + 2(x-\delta)^3 & \text{if } 0 < x < \delta \\ 0 & \text{otherwise} \end{cases}$$

It is easy to check that this kernel has the following properties:

Lemma 7. *(i)* $\int_{-\infty}^{\infty} k_\delta(x)\,dx = 1$.

(ii) $\psi(k_\delta) = 24\delta^{-5}$

(iii) $k_\delta(0) = \delta^{-1}$

Proof. (i) By symmetry, the positive and negative parts of k_δ have the same integral, so

$$\begin{aligned} \int_{-\infty}^{\infty} k_\delta(x)\,dx &= 2\delta^{-4} \int_{-\delta}^{0} \left(3\delta(x+\delta)^2 - 2(x+\delta)^3\right)\,dx \\ &= 2\delta^{-4} \int_0^{\delta} (3\delta t^2 - 2t^3)\,dt \\ &= 2\delta^{-4} \left[\delta t^3 - \frac{1}{2}t^4\right]_0^{\delta} \\ &= 2\delta^{-4} \left(\delta^4 - \frac{1}{2}\delta^4\right) \\ &= 1 \end{aligned}$$

(ii) We have

$$k_\delta{}''(x) = \delta^{-4} \begin{cases} 6\delta - 12(x+\delta) & \text{if } -\delta < x \leqslant 0 \\ 6\delta + 12(x-\delta) & \text{if } 0 < x < \delta \\ 0 & \text{otherwise} \end{cases}$$

so

$$\begin{aligned}\psi(k_\delta) &= \int_{-\infty}^{\infty} (k_\delta''(x))^2\,dx \\ &= \delta^{-8}\int_{-\delta}^{\delta} (6\delta - 12|x|)^2\,dx \\ &= 36\delta^{-8}\int_{-\delta}^{\delta} \left(\delta^2 - 4\delta|x| + 4x^2\right)\,dx \\ &= 36\delta^{-8}\left(2\delta^3 - 4\delta^3 + \frac{8}{3}\delta^3\right) \\ &= 24\delta^{-5}\end{aligned}$$

(iii) This is immediate by evaluating $k_\delta(0)$.

□

We can now define $f_n^\dagger(x) = \frac{1}{n}\sum_{i=1}^n k_{\delta_n}(x - x_i + b)$ for some suitably chosen δ_n. We obtain the following properties of $f^\dagger$ for sufficiently large n.

Lemma 8. *If $\|G_n - G\|_\infty \leqslant \sqrt{\frac{\log(n)}{n}}$, and n is sufficiently large, then*

(i) $\int_{-\infty}^{\infty} f^\dagger(x)\,dx = 1$.

(ii) $\psi(f^\dagger) \leqslant 48\left(\frac{5^4\psi(g)}{12}\right)^{\frac{1}{5}}\delta^{-4}$

(iii) For all $i = 1, \ldots, n$, $g^\dagger(x_i) \geqslant \frac{\epsilon}{4n}$

Proof. (i) This is obvious, since $f^\dagger$ is a mixture of k_δ.

(ii)

$$\begin{aligned}\psi(f^\dagger) &= \langle f^{\dagger\prime\prime}, f^{\dagger\prime\prime}\rangle \\ &= \left\langle \frac{1}{n}\sum k''_{\delta,x_i-b}, \frac{1}{n}\sum k''_{\delta,x_i-b}\right\rangle \\ &= \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n \left\langle k''_{\delta,x_i-b}, k''_{\delta,x_j-b}\right\rangle\end{aligned}$$

For all i, j, $\left\langle k''_{\delta,x_i-b}, k''_{\delta,x_j-b} \right\rangle \leqslant \psi(k_\delta) = 24\delta^{-5}$. On the other hand, if $|x_i - x_j| \geqslant 2\delta$, then k''_{δ,x_i-b} and k''_{δ,x_j-b} have disjoint supports, so $\langle k''_{\delta,x_i-b}, k''_{\delta,x_j-b} \rangle = 0$. Thus

$$\begin{aligned}
\psi(f^\dagger) &\leqslant \frac{24}{n^2\delta^5}|\{(i,j)||x_i - x_j| < 2\delta\}| \\
&= \frac{24}{n\delta^5}\sum_{i=1}^{n}(G_n(x_i + 2\delta) - G_n(x_i - 2\delta)) \\
&\leqslant \frac{24}{n\delta^5}\sum_{i=1}^{n}(G(x_i + 2\delta) - G(x_i - 2\delta) + 2\|G_n - G\|_\infty) \\
&\leqslant \frac{24}{n\delta^5}\sum_{i=1}^{n}(4\delta\|g\|_\infty + 2\|G_n - G\|_\infty) \\
&\leqslant \frac{24}{n\delta^5}\sum_{i=1}^{n}\left(\delta\left(\frac{5^4\psi(g)}{12}\right)^{\frac{1}{5}} + 2\sqrt{\frac{\log(n)}{n}}\right)
\end{aligned}$$

For large enough n, $2\sqrt{\frac{\log(n)}{n}} \leqslant \left(\frac{5^4\psi(g)}{12}\right)^{\frac{1}{5}}\delta$, so $\psi(f^\dagger) \leqslant 48\left(\frac{5^4\psi(g)}{12}\right)^{\frac{1}{5}}\delta^{-4}$

(iii) For $x \in \left[x_i - b - \frac{\delta}{2}, x_i - b + \frac{\delta}{2}\right]$, we have $f^\dagger(x) \geqslant \frac{1}{n}k_\delta(x - (x_i - b)) \geqslant \frac{1}{n}k_\delta\left(\frac{\delta}{2}\right) = \frac{\delta^{-1}}{2n}$. Therefore $g^\dagger(x_i) \geqslant \epsilon\frac{\delta}{2}\frac{\delta^{-1}}{2n} = \frac{\epsilon}{4n}$.

□

Now we define $f_n^*(x) = (1 - \gamma_n)f(x) + \gamma_n f_n^\dagger(x)$, where

$$\gamma_n = \frac{2M\sqrt{\log(n)}}{\sqrt{n} + 2M\sqrt{\log(n)}} \tag{3.28}$$

and δ_n is the solution to

$$\gamma_n = \left(\sqrt{2} - 1\right)\delta_n^2\sqrt{\frac{\psi(f)}{48\left(\frac{5^4\psi(g)}{12}\right)^{\frac{1}{5}}}} \tag{3.29}$$

Proposition 4. *For the above definition of $f_n^*(x)$, if $\|G_n - G\|_\infty < \sqrt{\frac{\log(n)}{n}}$ and n is sufficiently large, we have*

(i) $\psi(f_n^*) \leqslant 2\psi(f)$.

(ii) For $i = 1, \ldots, n$, $g_n^*(x_i) \geqslant (1 - \gamma_n)\,(g(x_i) \vee r_n)$, *where*

$$r_n = \frac{\epsilon \gamma_n}{4n} \tag{3.30}$$

Proof. (i)

$$\psi(f_n^*) \leqslant (1-\gamma_n)^2\psi(f)+2\gamma_n(1-\gamma_n)\sqrt{\psi(f)\psi(f^\dagger)}+\gamma_n^2\psi(f^\dagger) \leqslant \left(\gamma_n\sqrt{\psi(f^\dagger)} + \sqrt{\psi(f)}\right)^2$$

Substituting $\gamma_n = \left(\sqrt{2}-1\right)\delta_n^2\sqrt{\frac{\psi(f)}{48\left(\frac{5^4\psi(g)}{12}\right)^{\frac{1}{5}}}}$ and $\psi(f^\dagger) \leqslant 48\left(\frac{5^4\psi(g)}{12}\right)^{\frac{1}{5}}\delta^{-4}$ (from Lemma 8) gives

$$\gamma_n\sqrt{\psi(f^\dagger)} \leqslant (\sqrt{2}-1)\sqrt{\psi(f)}$$

so $\psi(f_n^*) < 2\psi(f)$.

(ii)

$$\begin{aligned} g_n^*(x_i) &= \gamma_n g^\dagger(x_i) + (1-\gamma_n)g(x_i) \\ &\geqslant \frac{\gamma_n\epsilon}{4n} \vee (1-\gamma_n)g(x_i) \\ &= (1-\gamma_n)\left(g(x_i) \vee \frac{\gamma_n\epsilon}{4n(1-\gamma_n)}\right) \\ &\geqslant (1-\gamma_n)\left(g(x_i) \vee \frac{\gamma_n\epsilon}{4n}\right) \end{aligned}$$

□

This means that

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_n^*(x_i)) \geqslant \log(1-\gamma_n) + \frac{1}{n}\sum_{i=1}^{n}\log\left(b(x_i)\right)$$

where $b_n(x) = g(x) \vee r_n$ where $r_n = \frac{\gamma_n\epsilon}{4n}$.

Proposition 5. *Whenever,* $\|G_n - G\| < \sqrt{\frac{\log(n)}{n}}$, *and* n *is sufficiently large*

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_n^*(x_i)) > -H(g) - \frac{3}{5}M\log(n)^{\frac{3}{2}}n^{-\frac{1}{2}}$$

where $H(g)$ is the entropy $-\int_{-\infty}^{\infty} g(x)\log(g(x))$

Proof. We set $b_n(x) = g(x) \vee r_n$, so that for all $i = 1, \ldots, n$, $g_n^*(x_i) \geqslant (1-\gamma_n)b_n(x_i)$.

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\log(b_n(x_i)) &= \log(r_n) - \int_{-\infty}^{\infty} G_n(x)\left(\frac{d}{dx}\log(b_n(x))\right)dx \\
&= \log(r_n) - \int_{-\infty}^{\infty} G_n(x)\left(\frac{b_n'(x)}{b_n(x)}\right)dx \\
&= \log(r_n) - \int_{-\infty}^{\infty} G(x)\left(\frac{b_n'(x)}{b_n(x)}\right)dx + \int_{-\infty}^{\infty}(G(x)-G_n(x))\left(\frac{b_n'(x)}{b_n(x)}\right)dx \\
&= \int_{-\infty}^{\infty} g(x)\log(b_n(x))\,dx + \int_{-\infty}^{\infty}(G(x)-G_n(x))\left(\frac{b_n'(x)}{b_n(x)}\right)dx \\
&\geqslant -H(g) - \|G(x)-G_n(x)\|_\infty \int_{-\infty}^{\infty}\left|\frac{b_n'(x)}{b_n(x)}\right|dx \\
\frac{1}{n}\sum_{i=1}^{n}\log(g_n^*(x_i)) &\geqslant -H(g) + \log(1-\gamma_n) - \|G(x)-G_n(x)\|_\infty \frac{2M}{5}\log\left(\frac{5^4\psi(g)}{3\times 2^{12}{r_n}^5}\right) \\
&\geqslant -H(g) + \log(1-\gamma_n) - \frac{2M}{5}\log\left(\frac{5^4\psi(g)}{3\times 2^{12}{r_n}^5}\right)\sqrt{\frac{\log(n)}{n}}
\end{aligned}$$

where the last line is by Proposition 1. Substituting $\gamma_n = \frac{c_n}{1+c_n}$ where $c_n = \frac{2}{5}M\sqrt{\frac{\log(n)}{n}}$, and letting $K_n = \left(\frac{5^4\psi(g)}{3\times 2^{12}}\right)^{\frac{1}{5}} {r_n}^{-1}$, this inequality becomes

$$\frac{1}{n}\sum_{i=1}^{n}\log(g)_n^*(x_i)) \geqslant -H(g) - c_n\log(K_n) - \log(c_n+1)$$

We have $\log(c_n+1) \leqslant c_n$, and for large enough n, we have $n^3 > e^2{K_n}^2$, so $\log(K_n)+1 < \frac{3\log(n)}{2}$ Thus, the inequality becomes

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_n^*(x_i)) \geqslant -H(g) - \frac{3c_n\log(n)}{2}$$

□

3.4.4 Proving Consistency

In this section, we use the $\tilde{g}_n$ and f_n^* defined in the previous sections to show that provided $\frac{|S_n|\log(n)^{\frac{1}{4}}}{n^{\frac{1}{4}}} \to 0$, when we set $\lambda_n = C_1 n^{\frac{7}{8}}\log(n)^{\frac{1}{8}}\sqrt{|S_n|}$, where $C_1 =$

$\frac{2^{\frac{31}{20}}3^{-\frac{1}{10}}5^{\frac{3}{20}}}{\psi(f_x)^{\frac{4}{5}}\psi(g)^{\frac{1}{10}}}$, our method is consistent.

Lemma 9. *If g and h are density functions with support S, and $a > 0$ then $\int_S g(x)\log(h(x)\vee a)\,dx \leqslant -H(g) + a|S|$.*

Proof.

$$\begin{aligned}\int_S g(x)\log(h(x)\vee a)\,dx + H(g) &= \int_S g(x)\log\left(\frac{h(x)\vee a}{g(x)}\right)dx \\ &\leqslant \int_S g(x)\left(\frac{h(x)\vee a - g(x)}{g(x)}\right)dx \\ &\leqslant \int_S (h(x) + a - g(x))\,dx \\ &= 1 + a|S| - 1 = a|S|\end{aligned}$$

□

Proposition 6. *If $\|G_n - G\| \leqslant \sqrt{\frac{\log(n)}{n}}$, and n is sufficiently large, then $\psi(\hat{f}_n) \leqslant 3\psi(f)$.*

Proof. By Propositions 5 and 4(i), the penalized likelihood of f_n^* is at least

$$-n\left(H(g) + 3M\log(n)^{\frac{3}{2}}n^{-\frac{1}{2}}\right) - 2\lambda_n\psi(f)$$

Since $\hat{f}_n$ maximizes the penalized likelihood, we have that

$$\sum_{i=1}^n \log(\hat{g}_n(x_i)) - \lambda_n\psi(\hat{f}_n) \geqslant -n\left(H(g) + 3M\log(n)^{\frac{3}{2}}n^{-\frac{1}{2}}\right) - 2\lambda_n\psi(f) \tag{3.31}$$

By Lemmas 1 and 3, $\hat{g}_n$ is Lipschitz with constant $L = \left(\frac{125}{144}\right)^{\frac{1}{5}}\psi(\hat{f}_n)^{\frac{2}{5}}$. By Proposition 3, we have that for any $0 < a < 1$,

$$\begin{aligned}\sum_{i=1}^n \log(\hat{g}_n(x_i)) &\leqslant \sum_{i=1}^n \log(\hat{g}_n(x_i)\vee a) \\ &\leqslant n\int_{S_n} \tilde{g}_n(x)\log(\hat{g}_n(x)\vee a)\,dx + \frac{n^{\frac{3}{4}}\log(n)^{\frac{1}{4}}}{C_3\psi(g)^{\frac{1}{5}}a}L \qquad (3.32)\end{aligned}$$

Meanwhile, we have

$$\begin{aligned}\int_{S_n} \tilde{g}_n(x)\log(\hat{g}_n(x)\vee a)\,dx &= \int_{S_n}(g(x)+\tilde{g}_n(x)-g(x))\log(\hat{g}_n(x)\vee a)\,dx\\ &\leqslant \int_{S_n} g(x)\log(\hat{g}_n(x)\vee a)\,dx + |S_n|\|\tilde{g}_n-g\|_\infty \sup(|\log(\hat{g}_n(x)\vee a)|)\\ &\leqslant -H(g)+a|S_n|+|S_n|\|\tilde{g}_n-g\|_\infty\left(\log(\sup(\hat{g}_n(x)))\vee(-\log(a))\right)\end{aligned}$$

Since $\hat{g}_n$ is Lipschitz with constant L, the equation $\int_{-\infty}^{\infty}\hat{g}_n(x)\,dx=1$ gives $\sup\hat{g}_n(x)\leqslant\sqrt{L}$, and by Proposition 2, $\|\tilde{g}_n-g\|_\infty\leqslant C_3\psi(g)^{\frac{1}{5}}\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}$ so

$$\log(\sup(\hat{g}_n(x)))\vee(-\log(a))\leqslant\log\left(\sqrt{L}\vee a^{-1}\right)=\frac{1}{2}\log\left(L\vee a^{-2}\right)$$

Therefore,

$$\int_{S_n}\tilde{g}_n(x)\log(\hat{g}_n(x)\vee a)\,dx\leqslant -H(g)+a|S_n|+\frac{C_3}{2}\psi(g)^{\frac{1}{5}}|S_n|\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}\log\left(L\vee a^{-2}\right)$$

Substituting this into (3.32), and substituting the result into (3.31) gives

$$\begin{aligned}-n\left(H(g)+3Mn^{-\frac{1}{2}}\log(n)^{\frac{3}{2}}\right)-2\lambda_n\psi(f)&\leqslant n\left(-H(g)+a|S_n|+\frac{C_3}{2}\psi(g)^{\frac{1}{5}}|S_n|\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}\log\left(L\vee a^{-2}\right)\right)\\ &\quad+\frac{n^{\frac{3}{4}}\log(n)^{\frac{1}{4}}}{C_3\psi(g)^{\frac{1}{5}}a}L-\lambda_n\psi(\hat{f}_n)\\ -3Mn^{\frac{1}{2}}\log(n)^{\frac{3}{2}}-2\lambda_n\psi(f)&\leqslant\frac{n}{2}|S_n|\left(2a+C_3\psi(g)^{\frac{1}{5}}\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}\left(\log\left(L\vee a^{-2}\right)\right)\right)\\ &\quad+\frac{n^{\frac{3}{4}}\log(n)^{\frac{1}{4}}}{C_3\psi(g)^{\frac{1}{5}}a}L-\lambda_n\psi(\hat{f}_n)\end{aligned}\tag{3.33}$$

Since a is arbitrary here, we can set $a=\left(\frac{\log(n)}{n}\right)^{\frac{1}{8}}\frac{\sqrt{L}}{\psi(g)^{\frac{1}{10}}\sqrt{C_3|S_n|}}$ so that $an|S_n|+\frac{n^{\frac{3}{4}}\log(n)^{\frac{1}{4}}}{C_3\psi(g)^{\frac{1}{5}}a}L=C_4n^{\frac{7}{8}}\log(n)^{\frac{1}{8}}\sqrt{L}\sqrt{|S_n|}$, where $C_4=2\psi(g)^{-\frac{1}{10}}{C_3}^{-\frac{1}{2}}$. Also, if $\psi(\hat{f}_n)\leqslant 3\psi(f)$, then the proposition is true. Otherwise, $a^{-2}=\frac{n^{\frac{1}{4}}\psi(g)^{\frac{1}{5}}C_3|S_n|}{L\log(n)^{\frac{1}{4}}}\leqslant\frac{n^{\frac{1}{4}}\psi(g)^{\frac{1}{5}}C_3|S_n|}{\left(\frac{125}{144}\right)^{\frac{1}{5}}(3\psi(f))^{\frac{2}{5}}\log(n)^{\frac{1}{4}}}$. For large enough n, we have $a^{-2}>L$, so we rearrange (3.33) to get

$$\begin{aligned}&\psi(\hat{f}_n) - 2\psi(f)\\ \leqslant &\frac{1}{\lambda_n}\left(\frac{C_3\psi(g)^{\frac{1}{5}}}{2}n^{\frac{3}{4}}|S_n|\log(n)^{\frac{1}{4}}\left(\log\left(\frac{C_3\psi(g)^{\frac{1}{5}}|S_n|n^{\frac{1}{4}}}{\left(\frac{125}{16}\right)^{\frac{1}{5}}\psi(f)^{\frac{2}{5}}\log(n)^{\frac{1}{4}}}\right)\right) + 3Mn^{\frac{1}{2}}\log(n)^{\frac{3}{2}} + C_4 n^{\frac{7}{8}}\log(n)^{\frac{1}{8}}\sqrt{L}\sqrt{|S_n|}\right)\end{aligned}$$

Substituting $L = \left(\frac{125}{144}\right)^{\frac{1}{5}}\psi(\hat{f}_n)^{\frac{2}{5}}$ and rearranging gives

$$\psi(\hat{f}_n) - \alpha_n\psi(\hat{f}_n)^{\frac{1}{5}} \leqslant \beta_n + 2\psi(f)$$

where

$$\alpha_n = \frac{C_4 n^{\frac{7}{8}}\log(n)^{\frac{1}{8}}\sqrt{|S_n|}}{\lambda_n}\left(\frac{125}{144}\right)^{\frac{1}{10}}$$

$$\beta_n = \frac{C_3\psi(g)^{\frac{1}{5}}}{2\lambda_n}n^{\frac{3}{4}}|S_n|\log(n)^{\frac{1}{4}}\log\left(\frac{C_3\psi(g)^{\frac{1}{5}}|S_n|n^{\frac{1}{4}}}{\left(\frac{125}{16}\right)^{\frac{1}{5}}\psi(f)^{\frac{1}{5}}\log(n)^{\frac{1}{4}}}\right) + \frac{3Mn^{\frac{1}{2}}\log(n)^{\frac{3}{2}}}{\lambda_n}$$

We have defined λ_n to satisfy $\lambda_n = \frac{2C_4}{\psi(f)^{\frac{4}{5}}}n^{\frac{7}{8}}\log(n)^{\frac{1}{8}}\sqrt{|S_n|}\left(\frac{125}{144}\right)^{\frac{1}{10}}$, so that $\alpha_n = \frac{\psi(f)^{\frac{4}{5}}}{2}$ and $\beta_n \to 0$, so for large enough n, $\beta_n < \left(1 - \frac{1}{2}\times 3^{\frac{1}{5}}\right)\psi(f)$, so we have $\frac{\psi(\hat{f}_n)}{\psi(f)} - \frac{1}{2}\left(\frac{\psi(\hat{f}_n)}{\psi(f)}\right)^{\frac{1}{5}} < 3 - \frac{1}{2}\times 3^{\frac{1}{5}}$, which gives $\psi(\hat{f}_n) < 3\psi(f)$. □

This proves that $\hat{f}_n$ is Lipschitz. Under this Lipschitz condition, we can show that if $\hat{g}_n$ is too far from g, then its penalized likelihood will be less than the penalized likelihood of g_n^*, contradicting the maximality.

Proposition 7. *If $\|g_L - g\|_\infty \geqslant \rho_n$, where g and g_L are both Lipschitz density functions with constant L, and $\|G_n - G\|_\infty < \sqrt{\frac{\log(n)}{n}}$, then*

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_L(x_i)) \leqslant -H(g) - \frac{{\rho_n}^4}{1536L^2} + \left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}|S_n|\left(\frac{1536L^3}{C_3\psi(g)^{\frac{1}{5}}{\rho_n}^4} + C_3\psi(g)^{\frac{1}{5}}\log\left(\frac{1536|S_n|L^{\frac{5}{2}}}{{\rho_n}^4}\right)\right)$$

Proof. By Proposition 3, we have that for any $a > 0$,

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n}\log(g_L(x_i)) &\leqslant \frac{1}{n}\sum_{i=1}^{n}\log(g_L(x_i)\vee a)\\ &\leqslant \int_{S_n}\tilde{g}_n(x)\log(g_L(x)\vee a)\,dx + \frac{\log(n)^{\frac{1}{4}}L}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}a}\end{aligned}$$

We use Proposition 2 to show

$$\begin{aligned}\int_{S_n}\tilde{g}_n(x)\log(g_L(x)\vee a)\,dx &= \int_{S_n}(g(x)+\tilde{g}_n(x)-g(x))\log(g_L(x)\vee a)\,dx\\ &\leqslant \int_{S_n}g(x)\log(g_L(x)\vee a)\,dx + |S_n|\|\tilde{g}_n-g\|_\infty\log\left(\frac{\sup_{x\in S_n}(g_L(x))\vee 1}{a}\right)\\ &\leqslant \int_{S_n}g(x)\log(g_L(x)\vee a)\,dx\\ &\qquad + C_3\psi(g)^{\frac{1}{5}}\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}|S_n|\log\left(\frac{\sup_{x\in S_n}(g_L(x))\vee 1}{a}\right)\end{aligned}$$

Let $\int_{S_n} g_L(x)\vee a\,dx = c$. Clearly $1\leqslant c\leqslant 1+a|S_n|$. Let $b(x) = \frac{g_L(x)\vee a}{c}$. By definition, $b(x)$ is a density function, and is Lipschitz with constant L, and $\|b-g\|_\infty \geqslant \rho_n - ((c-1)\|g_L\|_\infty\vee a)$, since for any x,

$$|b(x)-g_L(x)| = \left|\frac{g_L(x)\vee a}{c}-g_L(x)\right| = \begin{cases}\left(1-\frac{1}{c}\right)g_L(x) & \text{if } g_L(x)>a\\ \frac{a}{c}-g_L(x) & \text{if } g_L(x)\leqslant a\end{cases} \leqslant (c-1)\,g_L(x)\vee a$$

Since $\|g_L-g\|_\infty\geqslant\rho_n$, there is some x for which $|g_L(x)-g(x)|\geqslant\rho_n$. For this x, $|b(x)-g(x)|\geqslant|g_L(x)-g(x)|-|g_L(x)-b(x)|\geqslant\rho_n-((c-1)\|g_L\|_\infty\vee a)$. Now by Corollary 2,

$$\begin{aligned}\int_{S_n}g(x)\log(g_L(x)\vee a)\,dx &= \int_{S_n}g(x)\log\left(cb(x)\right)\,dx\\ &= \int_{S_n}g(x)(\log(b(x))+\log(c))\,dx\\ &\leqslant -H(g) - \frac{(\rho_n-((c-1)\|g_L\|_\infty)\vee a)^4}{48L^2}+\log(c)\end{aligned}$$

Thus, using the fact that $(c-1)\|g_L\|_\infty\leqslant a|S_n|\|g_L\|_\infty$, and $|S_n|\|g_L\|_\infty\geqslant\int_{S_n}g_L(x)\,dx =$

1, we get

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_L(x_i)) \leqslant -H(g) - \frac{(\rho_n - a(|S_n|\|g_L\|_\infty))^4}{48L^2} + \log(1+a|S_n|) + C_3\psi(g)^{\frac{1}{5}}\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}|S_n|\log\left(\frac{\sup_{x\in S_n}(g_L(x))\vee 1}{a}\right) + \frac{\log(n)^{\frac{1}{4}}L}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}a}$$

for any $a > 0$. In particular, we choose $a = \frac{{\rho_n}^4}{1536L^2|S_n|}$. Since

$$\rho_n \leqslant \|g_L - g\|_\infty \leqslant \|g_L\|_\infty + \|g\|_\infty \leqslant 2\sqrt{L} < 768^{\frac{1}{3}}\sqrt{L} \leqslant \left(\frac{768L^2}{\|g_L\|_\infty}\right)^{\frac{1}{3}}$$

(because $\|g_L\|_\infty \leqslant \sqrt{L}$, by the Lipschitz condition), it follows that $a \leqslant \frac{\rho_n}{2|S_n|\|g_L\|_\infty}$, so we have $\frac{(\rho_n - a|S_n|\|g_L\|_\infty)^4}{48L^2} \geqslant \frac{{\rho_n}^4}{768L^2}$ and $\log(1+a|S_n|) \leqslant \frac{{\rho_n}^4}{1536L^2}$, so that

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_L(x_i)) \leqslant -H(g) - \frac{{\rho_n}^4}{1536L^2} + \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}{\rho_n}^4} + C_3\psi(g)^{\frac{1}{5}}\left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}|S_n|\log\left(\frac{1536|S_n|L^2}{{\rho_n}^4}\left(\sup_{x\in S_n} g_L(x)\vee 1\right)\right)$$

By the Lipschitz condition, $\|g_L\|_\infty \leqslant \sqrt{L}$, so for $L \geqslant 1$,

$$\frac{1}{n}\sum_{i=1}^{n}\log(g_L(x_i)) \leqslant -H(g) - \frac{{\rho_n}^4}{1536L^2} + \left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}|S_n|\left(\frac{1536L^3}{C_3\psi(g)^{\frac{1}{5}}{\rho_n}^4} + C_3\psi(g)^{\frac{1}{5}}\log\left(\frac{1536|S_n|L^{\frac{5}{2}}}{{\rho_n}^4}\right)\right)$$

□

This allows us to show

Theorem 2. *If* $\|G_n - G\|_\infty < \sqrt{\frac{\log(n)}{n}}$ *and* $\lambda_n = C_1 n^{\frac{7}{8}}\log(n)^{\frac{1}{8}}\sqrt{|S_n|}$, *where* S_n *is the support of* $\hat{f}_n$ *and* $C_1 = \frac{2C_4}{\psi(f)^{\frac{4}{5}}}\left(\frac{125}{144}\right)^{\frac{1}{10}} = \frac{2^{\frac{31}{20}}3^{-\frac{1}{10}}5^{\frac{3}{20}}}{\psi(f_x)^{\frac{4}{5}}\psi(g)^{\frac{1}{10}}}$, *then for sufficiently large* n, $\|\hat{g}_n - g\|_\infty < C_2 n^{-\frac{1}{32}}|S_n|^{\frac{1}{8}}\log(n)^{\frac{1}{32}}$ *for some constant* C_2 *(depending on* $\psi(f)$ *and* $\psi(g)$*).*

Proof. By Proposition 6 and Lemma 3, g and $\hat{g}_n$ are both Lipschitz with constant $L = \left(\frac{125}{16}\right)^{\frac{1}{5}} \times \psi(f)^{\frac{2}{5}}$. Let $\rho_n = \|\hat{g}_n - g\|_\infty$. By Proposition 7,

$$\frac{1}{n}\sum_{i=1}^{n}\log(\hat{g}_n(x_i)) \leqslant -H(g) - \frac{{\rho_n}^4}{1536L^2} + \left(\frac{\log(n)}{n}\right)^{\frac{1}{4}}|S_n|\left(\frac{1536L^3}{C_3\psi(g)^{\frac{1}{5}}{\rho_n}^4} + C_3\psi(g)^{\frac{1}{5}}\log\left(\frac{1536|S_n|L^{\frac{5}{2}}}{{\rho_n}^4}\right)\right)$$

If $\rho_n > n^{-1}\left(1536|S_n|L^{\frac{5}{2}}\right)^{\frac{1}{4}}$, we have $\log\left(\frac{1536|S_n|L^{\frac{5}{2}}}{{\rho_n}^4}\right) < 4\log(n)$, so

$$\frac{1}{n}\sum_{i=1}^{n}\log(\hat{g}_n(x_i)) \leqslant -H(g) - \frac{{\rho_n}^4}{1536L^2} + \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}{\rho_n}^4} + 4C_3\psi(g)^{\frac{1}{5}}\frac{\log(n)^{\frac{5}{4}}}{n^{\frac{1}{4}}}|S_n|$$

On the other hand, by Proposition 5,

$$\frac{1}{n}\sum_{i=1}^{n}\log(\hat{g}_n(x_i)) - \frac{\lambda_n}{n}\psi(\hat{f}_n) \geqslant \frac{1}{n}\sum_{i=1}^{n}\log(g_n^*(x_i)) - \frac{\lambda_n}{n}\psi(f_n^*) \geqslant -H(g) - 3M\log(n)^{\frac{3}{2}}n^{-\frac{1}{2}} - \frac{2\lambda_n}{n}\psi(f)$$

Thus we have

$$-\frac{{\rho_n}^4}{1536L^2} + \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}{\rho_n}^4} + 4C_3\psi(g)^{\frac{1}{5}}\frac{\log(n)^{\frac{5}{4}}}{n^{\frac{1}{4}}}|S_n| - \frac{\lambda_n}{n}\psi(\hat{f}_n) \geqslant -3M\log(n)^{\frac{3}{2}}n^{-\frac{1}{2}} - \frac{2\lambda_n}{n}\psi(f)$$

$$\frac{{\rho_n}^4}{1536L^2} - \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}{\rho_n}^4} \leqslant 4C_3\psi(g)^{\frac{1}{5}}\frac{\log(n)^{\frac{5}{4}}}{n^{\frac{1}{4}}}|S_n| + 3M\log(n)^{\frac{3}{2}}n^{-\frac{1}{2}} + \frac{\lambda_n}{n}(2\psi(f) - \psi(\hat{f}_n))$$

For large enough n, $n^{-\frac{1}{4}}\log(n)^{\frac{1}{4}}|S_n|^{-1} < \frac{C_3\psi(g)^{\frac{1}{5}}}{3M}$, so

$$\frac{{\rho_n}^4}{1536L^2} - \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}{\rho_n}^4} \leqslant 5C_3\psi(g)^{\frac{1}{5}}\log(n)^{\frac{5}{4}}n^{-\frac{1}{4}}|S_n| + \frac{\lambda_n}{n}(2\psi(f) - \psi(\hat{f}_n))$$

With

$$\lambda_n = \frac{2C_4}{\psi(f)^{\frac{4}{5}}}n^{\frac{7}{8}}\log(n)^{\frac{1}{8}}\sqrt{|S_n|}\left(\frac{125}{144}\right)^{\frac{1}{10}}$$

for large enough n, and since $\psi(\hat{f}_n) > 0$, we deduce

$$\frac{\lambda_n}{n}(2\psi(f) - \psi(\hat{f}_n)) \leqslant \frac{2\lambda_n\psi(f)}{n} = 2C_4\psi(f)^{\frac{1}{5}}n^{-\frac{1}{8}}\log(n)^{\frac{1}{8}}\sqrt{|S_n|}\left(\frac{125}{144}\right)^{\frac{1}{10}}$$

so

$$\frac{{\rho_n}^4}{1536L^2} - \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}{\rho_n}^4} \leqslant 5C_3\psi(g)^{\frac{1}{5}}\log(n)^{\frac{5}{4}}n^{-\frac{1}{4}}|S_n| + C_5 n^{-\frac{1}{8}}|S_n|^{\frac{1}{2}}\log(n)^{\frac{1}{8}}$$

where $C_5 = 2C_4\psi(f)^{\frac{1}{5}}\left(\frac{125}{144}\right)^{\frac{1}{10}}$. For sufficiently large n, the first term on the right-hand side is much smaller than the second, so we get

$$\frac{{\rho_n}^4}{1536L^2} - \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}{\rho_n}^4} \leqslant 2C_5 n^{-\frac{1}{8}}|S_n|^{\frac{1}{2}}\log(n)^{\frac{1}{8}}$$

Rewriting this as $a_n(\rho_n)^8 - b_n(\rho_n)^4 - c_n \leqslant 0$ where

$$\begin{aligned} a_n &= \frac{1}{1536L^2} \\ b_n &= 2C_5 n^{-\frac{1}{8}}|S_n|^{\frac{1}{2}}\log(n)^{\frac{1}{8}} \\ c_n &= \frac{1536\log(n)^{\frac{1}{4}}|S_n|L^3}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}} \end{aligned}$$

we deduce

$$\begin{aligned} (\rho_n)^4 &\leqslant \frac{b_n + \sqrt{{b_n}^2 + 4a_n c_n}}{2a_n} \\ &= 768L^2\left(2C_5 n^{-\frac{1}{8}}|S_n|^{\frac{1}{2}}\log(n)^{\frac{1}{8}} + \sqrt{\left(2C_5 n^{-\frac{1}{8}}|S_n|^{\frac{1}{2}}\log(n)^{\frac{1}{8}}\right)^2 + \frac{4\log(n)^{\frac{1}{4}}|S_n|L}{C_3\psi(g)^{\frac{1}{5}}n^{\frac{1}{4}}}}\right) \\ &= 768L^2 n^{-\frac{1}{8}}|S_n|^{\frac{1}{2}}\log(n)^{\frac{1}{8}}\left(2C_5 + \sqrt{{4C_5}^2 + \frac{4L}{C_3\psi(g)^{\frac{1}{5}}}}\right) \end{aligned}$$

Thus

$$\rho_n \leqslant C_2\log(n)^{\frac{1}{32}}|S_n|^{\frac{1}{8}}n^{-\frac{1}{32}}$$

where, plugging in all the constants,

$$C_2 = \left(1 + \sqrt{1 + \frac{3^{\frac{2}{5}}}{16}}\right)^{\frac{1}{4}} 2^{\frac{179}{80}} 3^{\frac{9}{40}} 5^{\frac{27}{80}} \psi(f)^{\frac{1}{4}} \psi(g)^{-\frac{1}{40}}$$

□

From this, and the Dvoretzky-Kiefer-Wolfowitz inequality, Theorem 1 follows immediately.

3.5 Simulations

In this section, we compare the performance of our method with three of the most popular methods in the literature: `deamer` [49], `decon` [105] and `FEM` [70]. For `decon`, we compare with both their methods with and without FFT. We compare the performance under a range of true and error distributions, including common examples from Comte and Lacour [18] and Comte, Rozenholc and Taupin [19]. We simulate with a range of different sample sizes and SNRs, including many cases with smaller sample size and SNR, which are often excluded from simulations in the literature, because they highlight a particular weakness of existing methods.

3.5.1 Simulation design

To cover a large range of scenarios of interest, we simulate all combinations from seven true distributions, three error distributions, three SNRs, and three sample sizes. The true distributions used in the simulation are:

Normal distribution.	$X \sim N(0, 1)$
Chi-square distribution.	$X \sim \chi^2(4)/\sqrt{8}$
Beta distribution.	$X \sim \sqrt{39.2}\text{Beta}(2, 5)$
Laplace distribution.	$f_x(x) = \frac{1}{\sqrt{2}} \exp(-\sqrt{2}\lvert x - 1\rvert)$
Mixed normal distribution.	$X \sim \frac{2}{\sqrt{29}}(0.5N(-3, 1) + 0.5N(2, 1))$
Mixed gamma distribution.	$X \sim (0.4\Gamma(5, 1) + 0.6\Gamma(13, 1))/\sqrt{25.16}$
Cauchy distribution.	$f_x(x) = (1/\pi)/(1 + x^2)$

These distributions cover a range of situations including heavy tails, light tails,

symmetric and skew distributions, unimodal and bimodal distributions, and different levels of smoothness. With the exception of the Cauchy distribution (which has infinite variance), these densities have all been normalized to have unit variance. These distributions have previously been used in the literature [19]. However, the standardization used in the literature was incorrect for the mixture distributions, so we have corrected the standardization constants here.

For the pure error sample, we generate independent error samples from the following three distributions, scaled by a factor C. When the true distribution has finite variance (i.e. when it is not Cauchy), C^2 is the inverse of the SNR.

Laplace noise.	$f_\epsilon(\epsilon) = \frac{1}{\sqrt{2}} \exp(-\sqrt{2}\lvert\epsilon\rvert)$
Gaussian noise.	$f_\epsilon(\epsilon) = \frac{1}{\sqrt{2\pi}} \exp(-0.5\epsilon^2)$
Beta noise.	$f_\epsilon(\epsilon) = 30\sqrt{39.2}\epsilon(1-\epsilon)^4$

We choose these three distributions because they have different levels of smoothness. The normal distribution is super smooth and the Laplace distribution is ordinary smooth. Both the normal distribution and the Laplace distribution are often used in the literature on measurement error. The beta distribution often arises as convergence error, so will be particularly important in Chapter 4. The parameters of the three distributions are chosen to ensure the error distribution has unit variance. Because the `decon` package only permits a limited number of chosen distribution families, which does not include the beta distribution, we were unable to compare its performance in the beta noise simulations. We also only compare `FEM`'s performance in normal and laplace noise simulations.

In our simulation we study three sample sizes: 30, 100 and 300. In each case, we use the same sample size for the noisy data and for the pure error sample. Note, however, that because the `decon` package and `FEM` require a known error distribution family, we provide the true error distribution to them, which gives some unfair advantage to these methods.

In each scenario, we simulate 100 replicates.

Because we are running a large number of simulations and computation time is a factor, we have used the heuristic approach from Section 3.3 to quickly select λ_n.

3.5.2 Simulation Results

We use Mean Integrated Squared Error (MISE) to evaluate the performance of the estimators in each scenario. This measure is defined by MISE $= E\int(\hat{f}_x(x) - f_x(x))^2dx$. Here the MISE is computed as the empirical mean of the approximated ISE $\int(\hat{f}_x(x)-f_x(x))^2dx$ over 100 simulation replicates. This is a traditional method for evaluating the performance of deconvolution methods that is widely used in the literature [19]. We calculate the integral over an interval which contains both the support of the underlying true distribution from the 0.01% quantile to the 99.99% quantile and the estimated boundaries. `Deamer` and `FEM` do not give estimated boundaries for the true density, so we calculate their ISE on the same interval used for P-MLE. We numerically calculate the integral using the rectangle rule with the squared error evaluated at evenly-spaced points, where the spacing is chosen for each method so that there are 1000 (or 1024 for `decon` with FFT, 1500 for `FEM`) points within the interval returned by the method. We found that changing the number of points used to estimate the ISE did not noticeably affect the results. For `decon`, we recorded the better of the estimates with and without FFT in each scenario. For `FEM`, we run the algorithm with a sequence of smoothness penalty parameter values provided by Liu, Levine and Zhu [70] and record the best estimates as final results.

MISE for each method in each scenario is given in Tables 3.1–3.7. We see that P-MLE significantly outperforms all other methods in 76 out of 189 scenarios. P-MLE significantly outperforms `deamer` in 159 out of 189 scenarios; `deamer` significantly outperforms P-MLE in 16 out of 189 scenarios, and there are 14 scenarios where there is no significant difference between `deamer` and P-MLE. `Decon` does not significantly outperform P-MLE in any scenario, despite the fact that `decon` uses the true error distribution with estimated parameters, but there are 6 scenarios where P-MLE significantly outperforms `deamer` but not `decon`. P-MLE significantly outperforms `FEM` in 45 out of 126 scenarios (`FEM` could not be compared on the samples with beta error); `FEM` significantly outperforms P-MLE in 36 scenarios and there are 45 scenarios where there is no significant difference between `FEM` and P-MLE. Recall that `FEM` has an unfair advantage in these simulations, because it was given the true error distribution family, and the smoothing

Table 3.1: MISE of the estimates when the true distribution is normal
The best overall performance in each simulation is highlighted in yellow if it is significantly better than other methods and in orange otherwise.

normal-normal		n=30		n=100		n=300	
		mean	se	mean	se	mean	se
SNR=4	P-MLE	0.0193	0.0015	0.0085	0.0007	0.0055	0.0004
	FEM	0.0383	0.0028	0.0114	0.0009	0.0033	0.0003
	deamer	0.0486	0.0021	0.0097	0.0006	0.0029	0.0003
	decon	0.0283	0.0016	0.0147	0.0007	0.0081	0.0004
SNR=1	P-MLE	0.0232	0.0018	0.0121	0.0011	0.0068	0.0007
	FEM	0.0597	0.0052	0.0196	0.0018	0.0065	0.0005
	deamer	0.1104	0.0025	0.0622	0.0014	0.0353	0.0004
	decon	0.0636	0.0018	0.0375	0.0012	0.0245	0.0009
SNR=0.25	P-MLE	0.0726	0.0025	0.0360	0.0019	0.0198	0.0021
	FEM	0.0831	0.0076	0.0315	0.0025	0.0121	0.0010
	deamer	0.1973	0.0001	0.1782	0.0032	0.1247	0.0002
	decon	0.1066	0.0032	0.1030	0.0015	0.0836	0.0011
normal-laplace							
SNR=4	P-MLE	0.0187	0.0015	0.0072	0.0006	0.0046	0.0004
	FEM	0.0372	0.0029	0.0096	0.0008	0.0030	0.0002
	deamer	0.0455	0.0019	0.0075	0.0005	0.0029	0.0003
	decon	0.0526	0.0052	0.0506	0.0065	0.0391	0.0050
SNR=1	P-MLE	0.0241	0.0017	0.0123	0.0009	0.0064	0.0008
	FEM	0.0540	0.0050	0.0179	0.0017	0.0054	0.0004
	deamer	0.0974	0.0029	0.0462	0.0017	0.0156	0.0005
	decon	0.1083	0.0121	0.1142	0.0141	0.0902	0.0091
SNR=0.25	P-MLE	0.0682	0.0025	0.0412	0.0017	0.0217	0.0015
	FEM	0.0781	0.0072	0.0324	0.0026	0.0107	0.0010
	deamer	0.1814	0.0030	0.1265	0.0015	0.0900	0.0026
	decon	0.2186	0.0263	0.2251	0.0236	0.2042	0.0185
normal-beta							
SNR=4	P-MLE	0.0307	0.0038	0.0157	0.0012	0.0145	0.0008
	deamer	0.0518	0.0020	0.0101	0.0057	0.0032	0.0002
SNR=1	P-MLE	0.0572	0.0049	0.0463	0.0022	0.0431	0.0021
	deamer	0.1111	0.0024	0.0641	0.0013	0.0411	0.0006
SNR=0.25	P-MLE	0.0824	0.0055	0.0515	0.0048	0.0414	0.0032
	deamer	0.1966	0.0008	0.1784	0.0317	0.1242	0.0002

Table 3.2: MISE of the estimates when the true distribution is chi-squared

The best overall performance in each simulation is highlighted in yellow if it is significantly better than other methods and in orange otherwise.

chisq-normal		n=30		n=100		n=300	
		mean	se	mean	se	mean	se
SNR=4	P-MLE	0.0453	0.0025	0.0268	0.0013	0.0175	0.0007
	FEM	0.0551	0.0040	0.0276	0.0018	0.0171	0.0008
	deamer	0.0704	0.0025	0.0412	0.0012	0.0268	0.0006
	decon	0.0994	0.0024	0.0672	0.0014	0.0480	0.0009
SNR=1	P-MLE	0.0737	0.0028	0.0631	0.0024	0.0488	0.0015
	FEM	0.0869	0.0067	0.0458	0.0030	0.0290	0.0012
	deamer	0.1424	0.0033	0.0993	0.0020	0.0750	0.0011
	decon	0.1429	0.0019	0.1034	0.0016	0.0826	0.0010
SNR=0.25	P-MLE	0.1412	0.0028	0.1037	0.0028	0.0706	0.0029
	FEM	0.1389	0.0100	0.0768	0.0048	0.0503	0.0020
	deamer	0.2600	0.0024	0.2195	0.0034	0.1962	0.0002
	decon	0.1870	0.0027	0.1779	0.0013	0.1551	0.0011
chisq-laplace							
SNR=4	P-MLE	0.0446	0.0026	0.0260	0.0014	0.0144	0.0005
	FEM	0.0491	0.0031	0.0246	0.0016	0.0151	0.0006
	deamer	0.0684	0.0027	0.0369	0.0009	0.0206	0.0006
	decon	0.0750	0.0060	0.0564	0.0058	0.0404	0.0036
SNR=1	P-MLE	0.0722	0.0025	0.0516	0.0018	0.0394	0.0011
	FEM	0.0794	0.0051	0.0401	0.0025	0.0250	0.0020
	deamer	0.1244	0.0033	0.0773	0.0019	0.0505	0.0010
	decon	0.1322	0.0126	0.1318	0.0130	0.1151	0.0102
SNR=0.25	P-MLE	0.1314	0.0028	0.1035	0.0024	0.0781	0.0026
	FEM	0.1312	0.0094	0.0675	0.0042	0.0453	0.0020
	deamer	0.2350	0.0039	0.1811	0.0026	0.1351	0.0017
	decon	0.2114	0.0170	0.2874	0.0279	0.2968	0.0354
chisq-beta							
SNR=4	P-MLE	0.0659	0.0040	0.0497	0.0044	0.0532	0.0034
	deamer	0.0702	0.0022	0.0423	0.0012	0.0279	0.0007
SNR=1	P-MLE	0.0874	0.0043	0.0727	0.0035	0.0895	0.0038
	deamer	0.1435	0.0027	0.0986	0.0019	0.0724	0.0010
SNR=0.25	P-MLE	0.1412	0.0054	0.0803	0.0047	0.0744	0.0048
	deamer	0.2632	0.0020	0.2137	0.0032	0.1948	0.0006

Table 3.3: MISE of the estimates when the true distribution is beta
The best overall performance in each simulation is highlighted in yellow if it is significantly better than other methods and in orange otherwise.

beta-normal		n=30		n=100		n=300	
		mean	se	mean	se	mean	se
SNR=4	P-MLE	0.0212	0.0015	0.0114	0.0007	0.0074	0.0004
	FEM	0.0382	0.0030	0.0143	0.0010	0.0070	0.0004
	deamer	0.0328	0.0013	0.0168	0.0006	0.0096	0.0002
	decon	0.0646	0.0016	0.0395	0.0009	0.0257	0.0005
SNR=1	P-MLE	0.0341	0.0022	0.0248	0.0020	0.0158	0.0011
	FEM	0.0612	0.0050	0.0281	0.0021	0.0150	0.0009
	deamer	0.0803	0.0023	0.0474	0.0013	0.0311	0.001
	decon	0.0987	0.0018	0.0616	0.0009	0.0439	0.0007
SNR=0.25	P-MLE	0.0821	0.0025	0.0456	0.0020	0.0359	0.0021
	FEM	0.0938	0.0075	0.0469	0.0033	0.0260	0.0015
	deamer	0.1540	0.0027	0.1479	0.0029	0.1334	0.0002
	decon	0.1259	0.0028	0.1186	0.0010	0.0972	0.0009
beta-laplace							
SNR=4	P-MLE	0.0200	0.0014	0.0111	0.0007	0.0061	0.0003
	FEM	0.0364	0.0026	0.0140	0.0008	0.0070	0.0004
	deamer	0.0319	0.0014	0.0154	0.0007	0.0082	0.0003
	decon	0.0560	0.0052	0.0458	0.0048	0.0334	0.0033
SNR=1	P-MLE	0.0327	0.0021	0.0191	0.0011	0.0127	0.0006
	FEM	0.0550	0.0041	0.0245	0.0018	0.0112	0.0007
	deamer	0.0685	0.0022	0.0347	0.0014	0.0246	0.0007
	decon	0.1012	0.0103	0.1117	0.0121	0.1145	0.0011
SNR=0.25	P-MLE	0.0759	0.0024	0.0481	0.0016	0.0336	0.0016
	FEM	0.0902	0.0070	0.0413	0.0033	0.0204	0.0012
	deamer	0.1656	0.0037	0.1147	0.0026	0.0758	0.0013
	decon	0.1973	0.0189	0.2104	0.0240	0.2657	0.0359
beta-beta							
SNR=4	P-MLE	0.0255	0.0021	0.0127	0.0007	0.0082	0.0006
	deamer	0.0336	0.0014	0.0172	0.0006	0.0097	0.0003
SNR=1	P-MLE	0.0501	0.0038	0.0386	0.0030	0.0316	0.0018
	deamer	0.0592	0.0023	0.0480	0.0014	0.0311	0.0010
SNR=0.25	P-MLE	0.0811	0.0042	0.0436	0.0031	0.0400	0.0026
	deamer	0.1944	0.0026	0.1437	0.0027	0.1292	0.0014

Table 3.4: MISE of the estimates when the true distribution is Laplace

The best overall performance in each simulation is highlighted in yellow if it is significantly better than other methods and in orange otherwise.

laplace-normal		n=30		n=100		n=300	
		mean	se	mean	se	mean	se
SNR=4	P-MLE	0.0441	0.0020	0.0282	0.0011	0.0282	0.0011
	FEM	0.0487	0.0034	0.0194	0.0011	0.0108	0.0005
	deamer	0.1012	0.0024	0.0496	0.0008	0.0285	0.0003
	decon	0.0734	0.0016	0.0504	0.0012	0.0351	0.0008
SNR=1	P-MLE	0.0723	0.0024	0.0545	0.0020	0.0414	0.0015
	FEM	0.0684	0.0059	0.0341	0.0021	0.0201	0.0009
	deamer	0.1767	0.0028	0.1267	0.0019	0.0955	0.0010
	decon	0.1247	0.0019	0.0920	0.0014	0.0728	0.0011
SNR=0.25	P-MLE	0.1343	0.0024	0.0969	0.0023	0.0548	0.0032
	FEM	0.0927	0.0086	0.0551	0.0039	0.0345	0.0019
	deamer	0.2664	0.0013	0.2479	0.0033	0.1965	0.0007
	decon	0.1745	0.0035	0.1715	0.0016	0.1523	0.0013
laplace-laplace							
SNR=4	P-MLE	0.0427	0.0021	0.0250	0.0011	0.0157	0.0006
	FEM	0.0405	0.0030	0.0191	0.0013	0.0101	0.0005
	deamer	0.0986	0.0027	0.0437	0.0010	0.0221	0.0004
	decon	0.0704	0.0062	0.0674	0.0084	0.0421	0.0044
SNR=1	P-MLE	0.0700	0.0026	0.0477	0.0018	0.0332	0.0014
	FEM	0.0708	0.0052	0.0335	0.0022	0.0179	0.0009
	deamer	0.1576	0.0032	0.1030	0.0017	0.0658	0.0009
	decon	0.1283	0.0123	0.1256	0.0134	0.1006	0.0097
SNR=0.25	P-MLE	0.1332	0.0027	0.1021	0.0024	0.0673	0.0028
	FEM	0.1152	0.0103	0.0606	0.0040	0.0324	0.0019
	deamer	0.2501	0.0032	0.1971	0.0013	0.1604	0.0027
	decon	0.2231	0.0194	0.2996	0.0446	0.2972	0.0294
laplace-beta							
SNR=4	P-MLE	0.0758	0.0042	0.0604	0.0036	0.0441	0.0036
	deamer	0.1040	0.0027	0.0520	0.0009	0.0293	0.0004
SNR=1	P-MLE	0.1068	0.0069	0.1100	0.0049	0.1361	0.0037
	deamer	0.1804	0.0026	0.1282	0.0017	0.0946	0.0010
SNR=0.25	P-MLE	0.1478	0.0059	0.1096	0.0064	0.1129	0.0063
	deamer	0.2667	0.0012	0.2534	0.0030	0.1952	0.0002

Table 3.5: MISE of the estimates when the true distribution is mixture-normal The best overall performance in each simulation is highlighted in yellow if it is significantly better than other methods and in orange otherwise.

mixnormal-normal		n=30		n=100		n=300	
		mean	se	mean	se	mean	se
SNR=4	P-MLE	0.1145	0.0027	0.0780	0.0025	0.0412	0.0049
	FEM	0.0784	0.0049	0.0669	0.0025	0.0578	0.0010
	deamer	0.1565	0.0019	0.1145	0.0018	0.0800	0.0016
	decon	0.1466	0.0015	0.1292	0.0008	0.1041	0.0009
SNR=1	P-MLE	0.1372	0.0022	0.1501	0.0048	0.1245	0.0040
	FEM	0.1754	0.0010	0.1589	0.0060	0.1414	0.0030
	deamer	0.2108	0.0022	0.1642	0.0012	0.1388	0.0004
	decon	0.1681	0.0018	0.1464	0.0009	0.1365	0.0006
SNR=0.25	P-MLE	0.1724	0.0022	0.1456	0.0014	0.1800	0.0046
	FEM	0.2226	0.0073	0.1923	0.0060	0.1766	0.0038
	deamer	0.2942	0.0010	0.2731	0.0034	0.2240	0.0007
	decon	0.2069	0.0030	0.2023	0.0014	0.1849	0.0010
mixnormal-laplace							
SNR=4	P-MLE	0.1085	0.0033	0.0646	0.0017	0.0462	0.0054
	FEM	0.0818	0.0058	0.0553	0.0018	0.0461	0.0008
	deamer	0.1475	0.0024	0.0848	0.0025	0.0404	0.0014
	decon	0.1010	0.0055	0.0771	0.0058	0.0429	0.0041
SNR=1	P-MLE	0.1360	0.0021	0.1305	0.0032	0.0983	0.0045
	FEM	0.1481	0.0093	0.1222	0.0040	0.1049	0.0020
	deamer	0.1986	0.0028	0.1524	0.0015	0.1252	0.0006
	decon	0.1735	0.0103	0.1555	0.0121	0.1206	0.0091
SNR=0.25	P-MLE	0.1674	0.0020	0.1486	0.0014	0.1531	0.0029
	FEM	0.2020	0.0085	0.1751	0.0050	0.1558	0.0028
	deamer	0.2819	0.0029	0.2253	0.0012	0.1989	0.0026
	decon	0.3090	0.0307	0.3337	0.0341	0.3521	0.0281
mixnormal-beta							
SNR=4	P-MLE	0.1601	0.0041	0.1527	0.0030	0.1312	0.0026
	deamer	0.1561	0.0022	0.1159	0.0014	0.0747	0.0015
SNR=1	P-MLE	0.1747	0.0031	0.2095	0.0063	0.1921	0.0033
	deamer	0.2129	0.0021	0.1679	0.0008	0.1385	0.0004
SNR=0.25	P-MLE	0.1866	0.0033	0.1776	0.0032	0.1978	0.0034
	deamer	0.2951	0.0007	0.2812	0.0029	0.2281	0.0003

Table 3.6: MISE of the estimates when the true distribution is mixture-gamma

The best overall performance in each simulation is highlighted in yellow if it is significantly better than other methods and in orange otherwise.

mixgamma-normal		n=30		n=100		n=300	
		mean	se	mean	se	mean	se
SNR=4	P-MLE	0.0328	0.0016	0.0221	0.0009	0.0161	0.0008
	FEM	0.0498	0.0039	0.0271	0.0014	0.0187	0.0007
	deamer	0.0412	0.0015	0.0258	0.0005	0.0209	0.0002
	decon	0.0756	0.0017	0.0458	0.0006	0.0335	0.0004
SNR=1	P-MLE	0.0381	0.0021	0.0362	0.0028	0.0267	0.0012
	FEM	0.0815	0.0053	0.0491	0.0030	0.0321	0.0012
	deamer	0.0789	0.0024	0.0494	0.0012	0.0358	0.001
	decon	0.1020	0.0018	0.0655	0.0010	0.0497	0.0005
SNR=0.25	P-MLE	0.0780	0.0026	0.0469	0.0014	0.0462	0.0018
	FEM	0.1080	0.0060	0.0635	0.0033	0.0422	0.0016
	deamer	0.1908	0.0029	0.1428	0.0026	0.1314	0.0005
	decon	0.1232	0.0028	0.1191	0.0010	0.0990	0.0094
mixgamma-laplace							
SNR=4	P-MLE	0.0321	0.0015	0.0210	0.0008	0.0133	0.0005
	FEM	0.0472	0.0029	0.0242	0.0011	0.0157	0.0005
	deamer	0.0394	0.0013	0.0252	0.0005	0.0209	0.0002
	decon	0.0650	0.0052	0.0519	0.0059	0.0433	0.0039
SNR=1	P-MLE	0.0356	0.0017	0.0301	0.0011	0.0243	0.0013
	FEM	0.0726	0.0042	0.0395	0.0018	0.0259	0.0010
	deamer	0.0698	0.0022	0.0418	0.001	0.0322	0.0005
	decon	0.1101	0.0119	0.1141	0.0114	0.0988	0.0104
SNR=0.25	P-MLE	0.0766	0.0023	0.0523	0.0016	0.0417	0.0019
	FEM	0.1075	0.0068	0.0591	0.0028	0.0380	0.0013
	deamer	0.1599	0.0039	0.1140	0.0025	0.0873	0.0104
	decon	0.2709	0.0446	0.2323	0.0285	0.2703	0.0254
mixgamma-beta							
SNR=4	P-MLE	0.0423	0.0020	0.0315	0.0011	0.0246	0.0007
	deamer	0.0394	0.0012	0.0259	0.0005	0.0209	0.0002
SNR=1	P-MLE	0.0644	0.0030	0.0614	0.0029	0.0503	0.0016
	deamer	0.0802	0.0021	0.0505	0.0014	0.0420	0.0010
SNR=0.25	P-MLE 2	0.0884	0.0041	0.0623	0.0035	0.0565	0.0022
	deamer	0.1927	0.0026	0.1403	0.0024	0.1344	0.0013

Table 3.7: MISE of the estimates when the true distribution is Cauchy

The best overall performance in each simulation is highlighted in yellow if it is significantly better than other methods and in orange otherwise.

cauchy-normal		n=30		n=100		n=300	
		mean	se	mean	se	mean	se
C=0.5	P-MLE	0.0294	0.0024	0.0142	0.0007	0.0100	0.0004
	FEM	0.0425	0.0033	0.0141	0.0010	0.0056	0.0003
	deamer	0.0513	0.0012	0.0354	0.0014	0.0106	0.0003
	decon	0.0244	0.0010	0.0187	0.0016	0.0196	0.0027
C=1	P-MLE	0.0389	0.0080	0.0153	0.0007	0.0092	0.0003
	FEM	0.0571	0.0053	0.0179	0.0016	0.0063	0.0004
	deamer	0.0649	0.0013	0.0369	0.0008	0.0239	0.0004
	decon	0.0368	0.0010	0.0803	0.0064	0.0315	0.0056
C=2	P-MLE	0.0473	0.0032	0.0285	0.0010	0.0228	0.0007
	FEM	0.0880	0.0081	0.0263	0.0021	0.0147	0.0009
	deamer	0.0944	0.0001	0.0903	0.0004	0.0658	0.0018
	decon	0.0505	0.0016	0.2189	0.0221	0.0494	0.0034
cauchy-laplace							
C=0.5	P-MLE	0.0308	0.0025	0.0135	0.0006	0.0104	0.0004
	FEM	0.0403	0.0027	0.0130	0.0009	0.0061	0.0004
	deamer	0.0503	0.0012	0.0347	0.0016	0.0110	0.0005
	decon	0.0535	0.0050	0.0521	0.0047	0.0472	0.0061
C=1	P-MLE	0.0330	0.0048	0.0146	0.0006	0.0093	0.0004
	FEM	0.0541	0.0046	0.0186	0.0015	0.0078	0.0006
	deamer	0.0631	0.0014	0.0337	0.0006	0.0202	0.0003
	decon	0.0833	0.0073	0.1286	0.0173	0.1016	0.0096
C=2	P-MLE	0.0354	0.0016	0.0236	0.0009	0.0173	0.0009
	FEM	0.0880	0.0065	0.0317	0.0024	0.0126	0.0010
	deamer	0.0934	0.0005	0.0695	0.0019	0.0523	0.0005
	decon	0.1822	0.0180	0.2157	0.0266	0.3105	0.0581
cauchy-beta							
C=0.5	P-MLE	0.0428	0.0041	0.0164	0.0007	0.0108	0.0004
	deamer	0.0783	0.0020	0.0357	0.0004	0.0108	0.0003
C=1	P-MLE	0.0557	0.0049	0.0297	0.0013	0.0217	0.0010
	deamer	0.0899	0.0017	0.0349	0.0007	0.0238	0.0004
C=2	P-MLE	0.0708	0.0062	0.0532	0.0022	0.0516	0.0011
	deamer	0.0914	0.0001	0.0901	0.0005	0.0620	0.0016

Table 3.8: Simulation Results for some Scenarios with different heuristic values for λ_n.

Significantly better results are highlighted in yellow. Results that are not significantly different are highlighted in orange.

Scenario				Best	Old	New	Deamer	Decon	FEM
Truth	Error	n	C	Heuristic λ_n	MISE (SE)	MISE (SE)	MISE	MISE	MISE
cauchy	normal	30	1	1,000	0.0389(0.0080)	0.0314(0.0015)		0.0368(0.0010)	
mixnormal	beta	100	1	10,000,000	0.2095(0.0063)	0.1670(0.0054)	0.1679(0.0054)		
mixgamma	beta	100	1	1,000,000	0.0614(0.0029)	0.0506(0.0032)	0.0505(0.0014)		
normal	laplace	300	0.5	100,000	0.0046(0.0004)	0.0028(0.0002)	0.0029(0.0003)		0.0030
chi-squared	beta	300	1	10,000,000	0.0895(0.0038)	0.0800(0.0048)	0.0724(0.0010)		
mixgamma	beta	300	1	10,000,000	0.0503(0.0016)	0.0410(0.0015)	0.0420(0.0010)		
laplace	beta	300	1	100,000	0.1361(0.0037)	0.1010(0.0034)	0.0946(0.0010)		
normal	beta	300	1	10,000,000	0.0431(0.0021)	0.0345(0.0012)	0.0411(0.0006)		
beta	beta	300	1	10,000,000	0.0316(0.0018)	0.0250(0.0019)	0.0311(0.0010)		

penalty was chosen based on the true distribution.

It is also worth noting that because of the scale of the simulations, we used the heuristic approach to selecting the penalty parameter λ_n. For analyzing a single data set, we would select λ_n more carefully using cross-validation, which would be expected to lead to better results. To see how much potential improvement could be made by choosing λ_n more carefully, we reran a number of scenarios where P-MLE was not the best method with a range of different values of λ_n. We found that better choices of λ_n could improve the results to a modest extent. Details of these simulations are in Table 3.8.

For the simulations, we wanted to know how much the results could be improved with a better choice of the smoothness penalty λ_n. We reran the scenarios where P-MLE did not outperform the other methods using different ratios for the heuristic method of setting λ_n. We ran each scenario using values 1000, 10000, 100000, 1000000, and 10000000 for the ratio in the heuristic for setting λ_n. We then looked at the best results in each scenario. In 9 of the scenarios, there was a difference from the outcome reported in Table 3.9. This is selecting the value of λ_n which performs best on the data, so is not a completely reliable estimate of performance. However, it shows that there is potential to improve performance with a better choice of λ_n. Furthermore, these results are using the same heuristic for λ_n for all simulations in each scenario, rather than tuning the value of λ_n for each simulation. It is therefore possible that more careful tuning could improve the performance of P-MLE still further.

Table 3.9: Summary of outcomes for each scenario
A — P-MLE significantly outperforms all other methods.
B — P-MLE significantly outperforms both `deamer` and `decon`.
C — P-MLE significantly outperforms `deamer` but not `decon`.
D — No significant difference between P-MLE and `deamer`.
E — `Deamer` significantly outperforms P-MLE.
F — P-MLE significantly outperforms `FEM`.
G — `FEM` significantly outperforms P-MLE.
H — No significant difference between P-MLE and `FEM`.

Outcome		A	B	C	D	E	F	G	H
True distribution	Normal	12	21	0	4	2	9	3	6
	Chi-squared	5	23	0	2	2	0	8	10
	Beta	16	26	0	1	0	8	2	8
	Laplace	6	23	0	1	3	0	14	4
	Mixnormal	11	18	3	2	4	9	4	5
	Mixgamma	17	22	0	1	4	13	0	5
	Cauchy	11	21	3	3	0	6	5	7
Error distribution	Normal	19	55	5	2	1	24	18	21
	Laplace	19	58	1	3	1	21	18	24
	Beta	40	NA	NA	9	14	NA	NA	NA
SNR	$C = 0.5$	20	40	2	11	10	17	10	15
	$C = 1$	27	52	2	3	6	18	9	15
	$C = 2$	31	61	2	0	0	10	17	15
Sample size	30	38	56	4	3	0	23	4	15
	100	25	53	1	4	5	14	13	15
	300	13	44	1	7	11	8	19	15

Table 3.9 gives a breakdown of these simulation outcomes. We see that P-MLE outperforms `deamer` and `decon` for all true distributions, but the level of outperformance is less when the true distribution is a mixture of normal distributions. This is not surprising, since the mixture of normal distributions is bimodal, so the smoothness penalty will be larger. However, even in this more challenging case, P-MLE still outperforms `deamer` and `decon` in most scenarios.

P-MLE outperforms `FEM` except when the true distribution is chi-square or Laplace. Recall that the penalty for FEM is the smoothness of the log-density function. For the Laplace distribution, the log-density is piecewise linear, so the smoothness penalty on the log-density is small, meaning that `FEM` has a bias towards a Laplace distribution. P-MLE has a similar bias towards a spline density function with a small number of knots, but we did not include any such distributions in our simulation study. For the chi-square distribution, the log-density smoothness penalty is small for the right tail, and tends to infinity near zero. This means that `FEM` will distort the left-tail of the density close to zero. However, because the density is small here, this distortion has relatively little effect on the density function.

The distribution of the noise has an influence on the comparison with `deamer` and `decon`, with `deamer` performing relatively better when the noise follows a beta distribution. SNR is another important factor, with P-MLE performing much better than `deamer` and `decon` in the low SNR case.

Finally, sample size is important for comparisons with all other methods, with P-MLE performing relatively better in scenarios with low sample size. This might be explained by the hard thresholding of other methods. The Fourier-based methods `deamer` and `decon` use hard-thresholding of the Fourier transform of the density to regularize the estimates. To achieve sufficient regularization, for small sample sizes, the threshold must severely limit the space of possible functions. By contrast, the penalized likelihood approach of P-MLE is better able to find a compromise between smoothness and likelihood.

We also compare the distribution of the integrated squared error (ISE) for different methods in each scenario. Figure 3.2–3.10 compare boxplots of the ISE for the five methods for each combination of true and error distributions(both

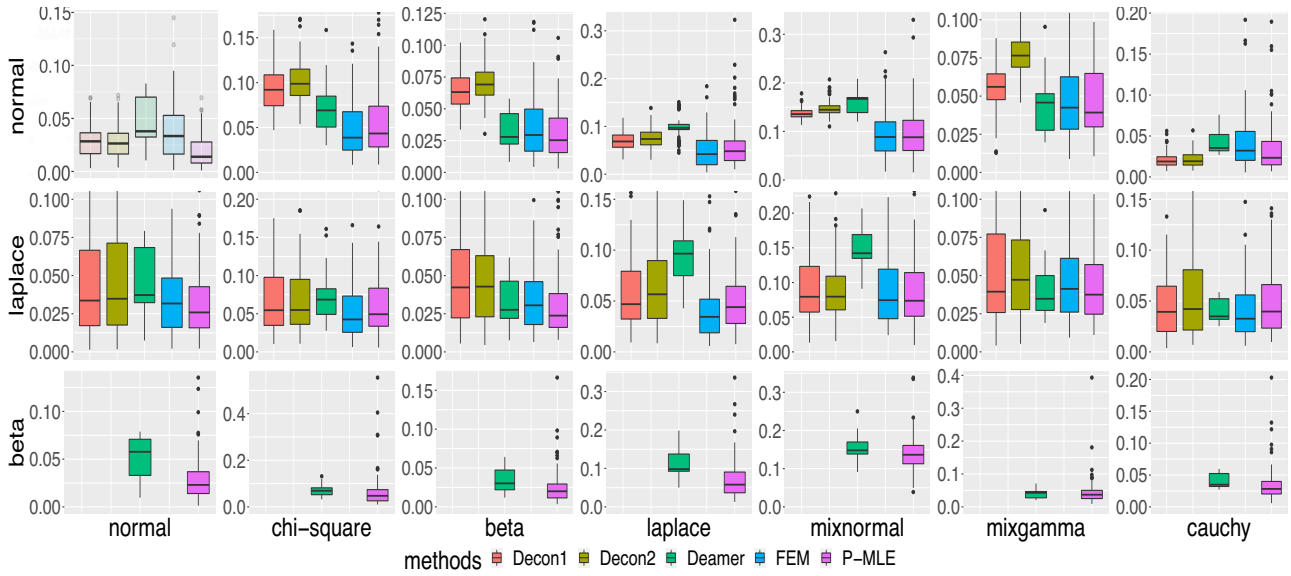


Figure 3.2: Sample distribution of ISE for sample size 30 and SNR 4. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

`decon` methods with or without FFT are displayed). From these plots, we can see that although there are some outliers, and sometimes larger variances, the MISE results given in Tables 3.1–3.7 are mostly consistent with the results shown in Figure 3.2–3.10, rather than being caused by a few outliers.

3.6 Real data analysis

We apply our P-MLE method to a real data set. The Framingham data [15] records the systolic blood pressure measured for 1615 male subjects. Each subject's blood pressure was measured twice at a first visit and twice at a second visit eight years later. We use the measurements at the second visit only. Let SBP_{21} and SBP_{22} denote the two observations at the second visit. Let SBP_2 be the average of SBP_{21} and SBP_{22}. We estimate the density of the underlying true blood pressure X from SBP_2. Let e_{21} and e_{22} be the measurement errors of SBP_{21} and SBP_{22} respectively. Now $SBP_2 = X + e$, where $e = \frac{e_{21}+e_{22}}{2}$. Also,

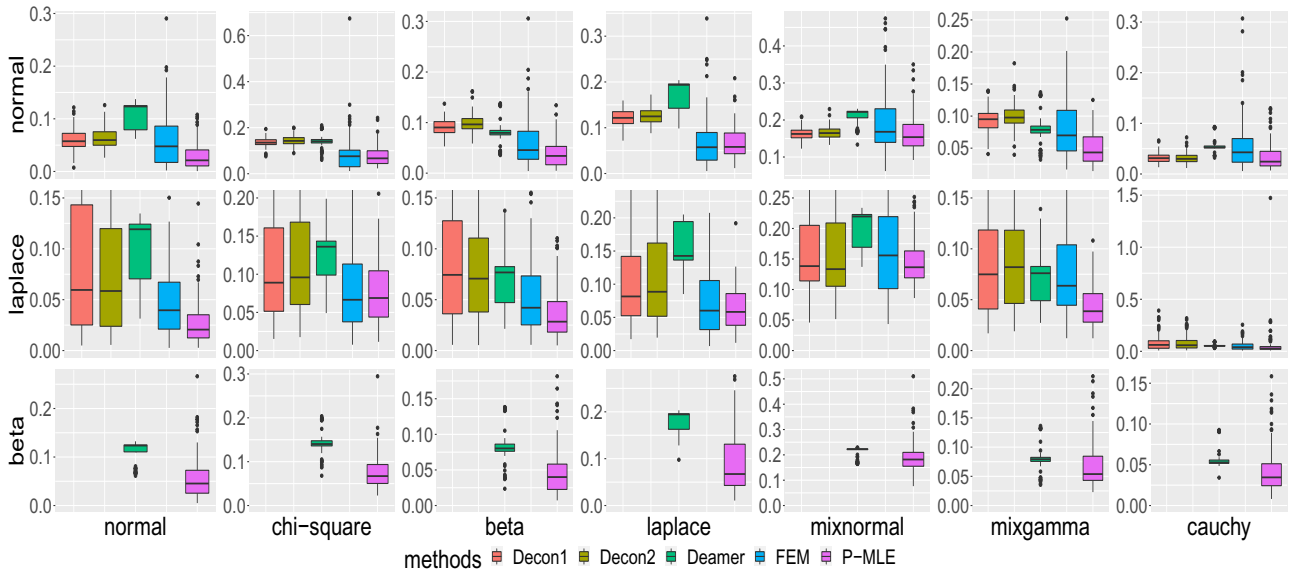


Figure 3.3: Sample distribution of ISE for sample size 30 and SNR 1. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

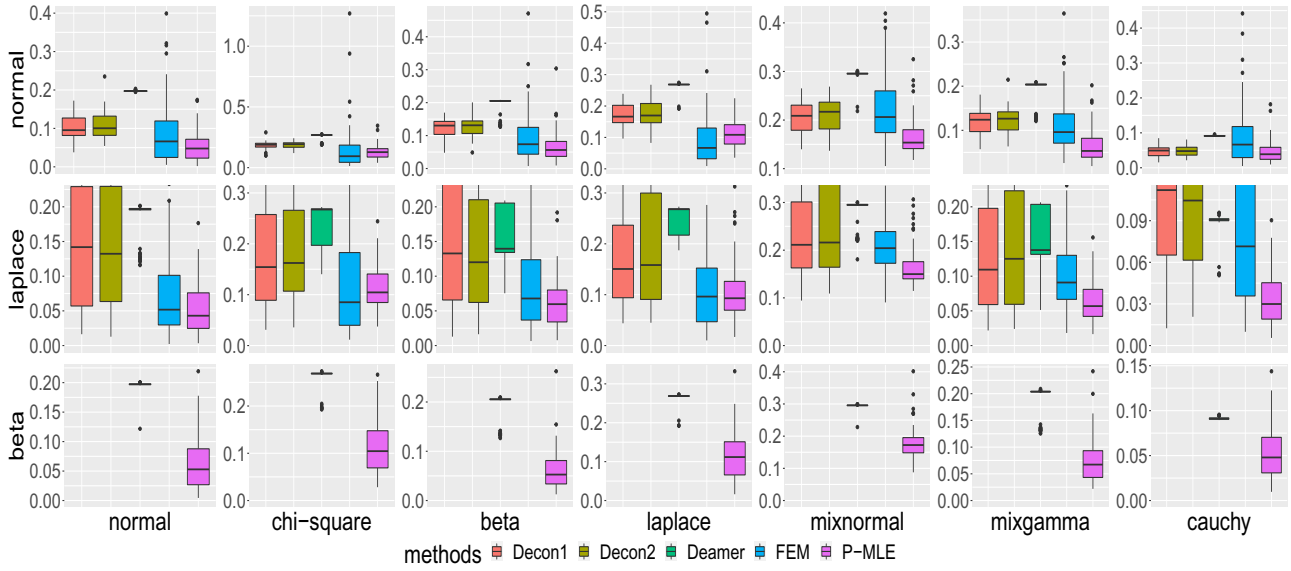


Figure 3.4: Sample distribution of ISE for sample size 30 and SNR 0.25. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

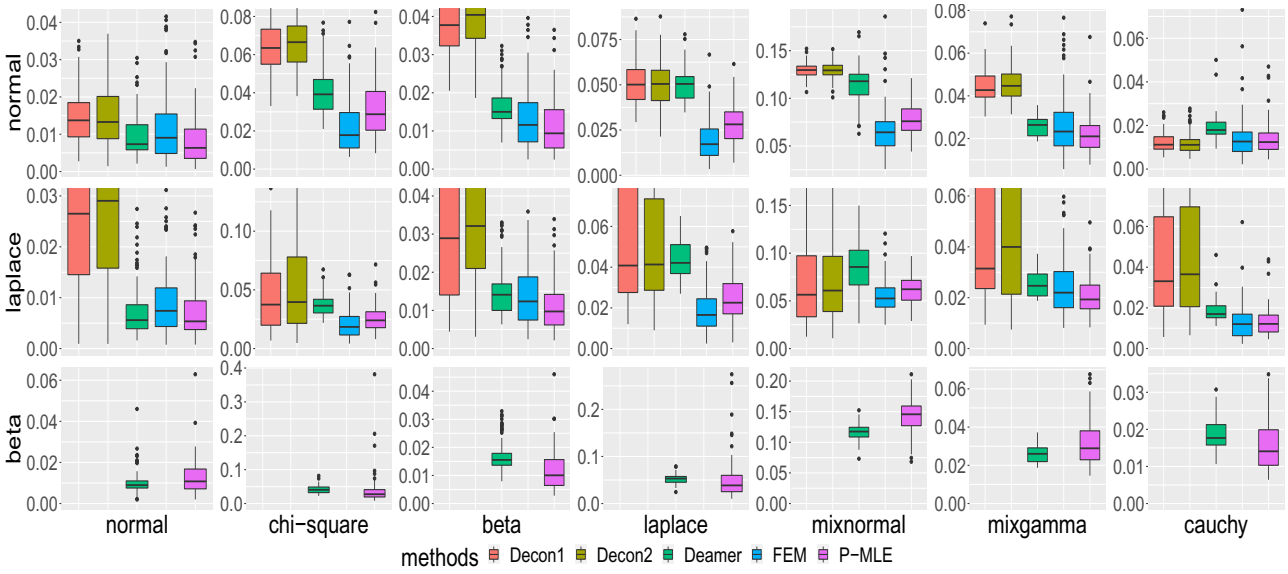


Figure 3.5: Sample distribution of ISE for sample size 100 and SNR 4. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

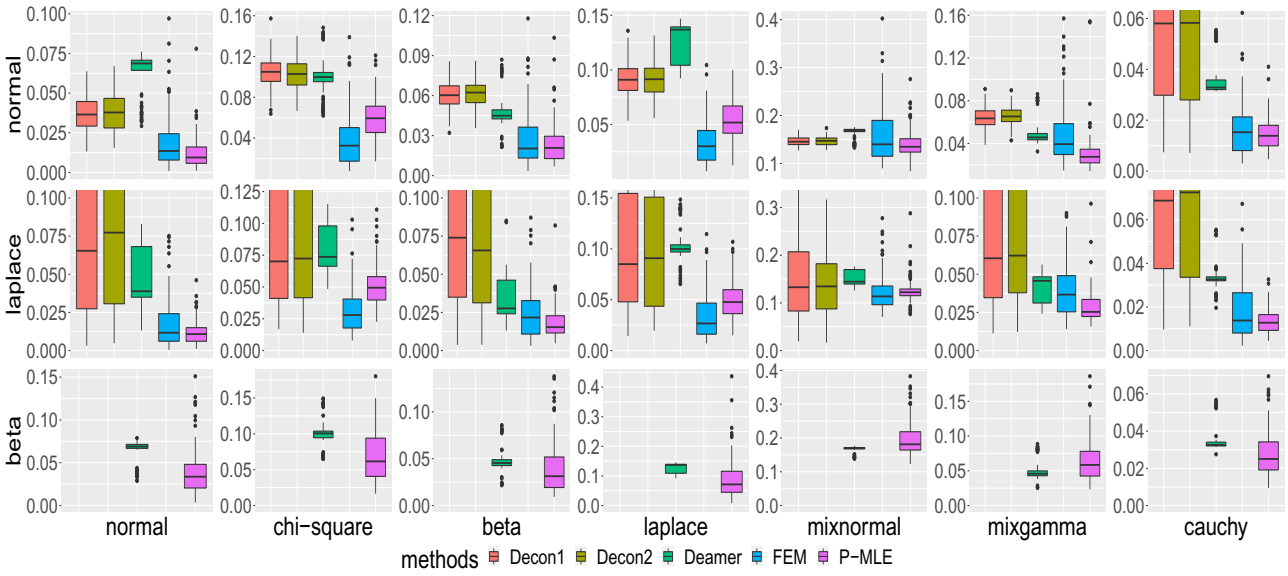


Figure 3.6: Sample distribution of ISE for sample size 100 and SNR 1. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

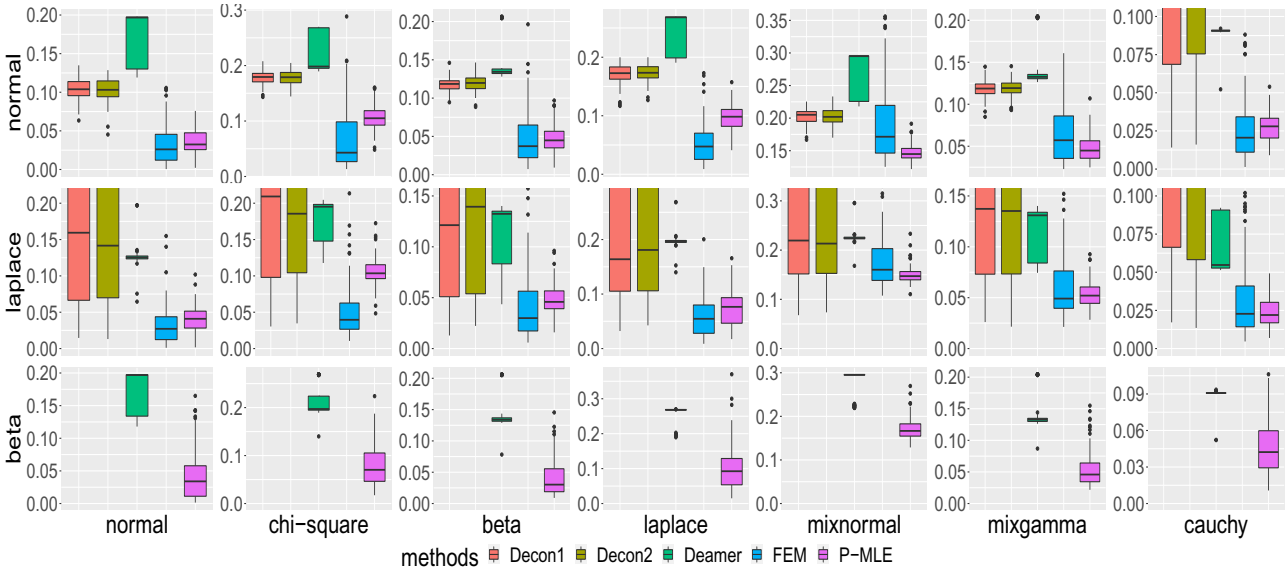


Figure 3.7: Sample distribution of ISE for sample size 100 and SNR 0.25. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

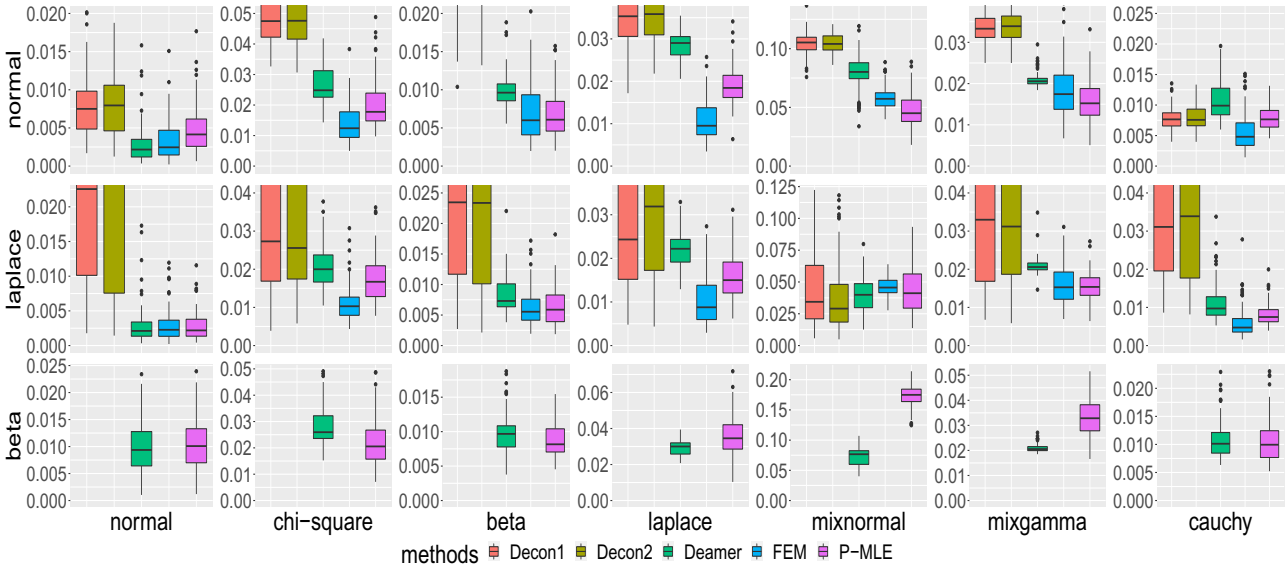


Figure 3.8: Sample distribution of ISE for sample size 300 and SNR 4. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

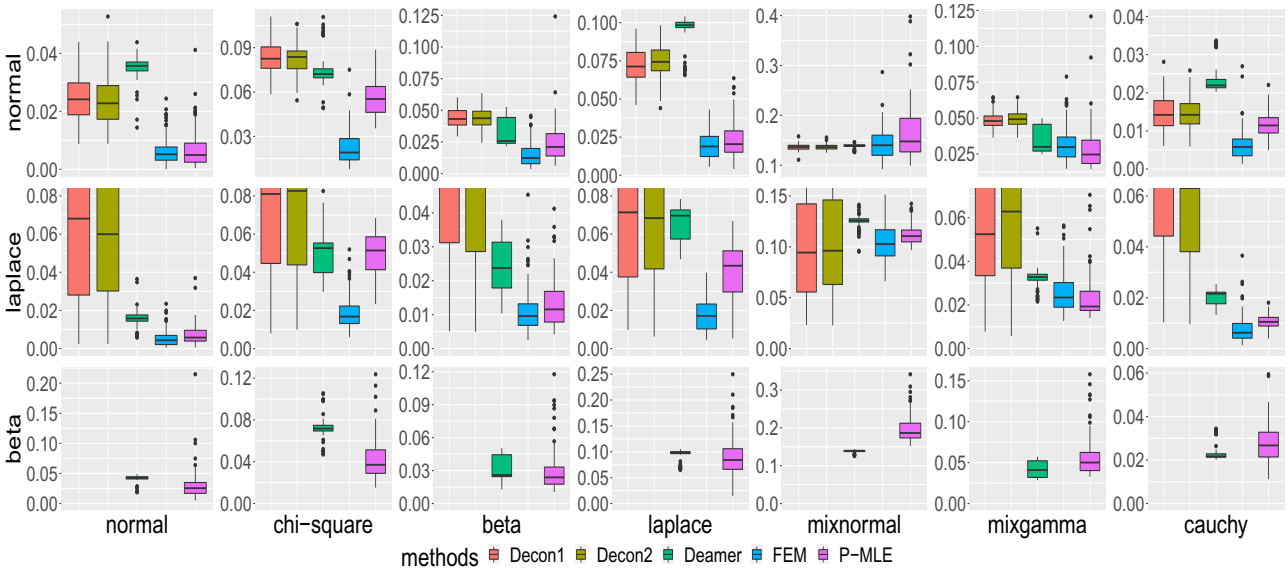


Figure 3.9: Sample distribution of ISE for sample size 300 and SNR 1. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.

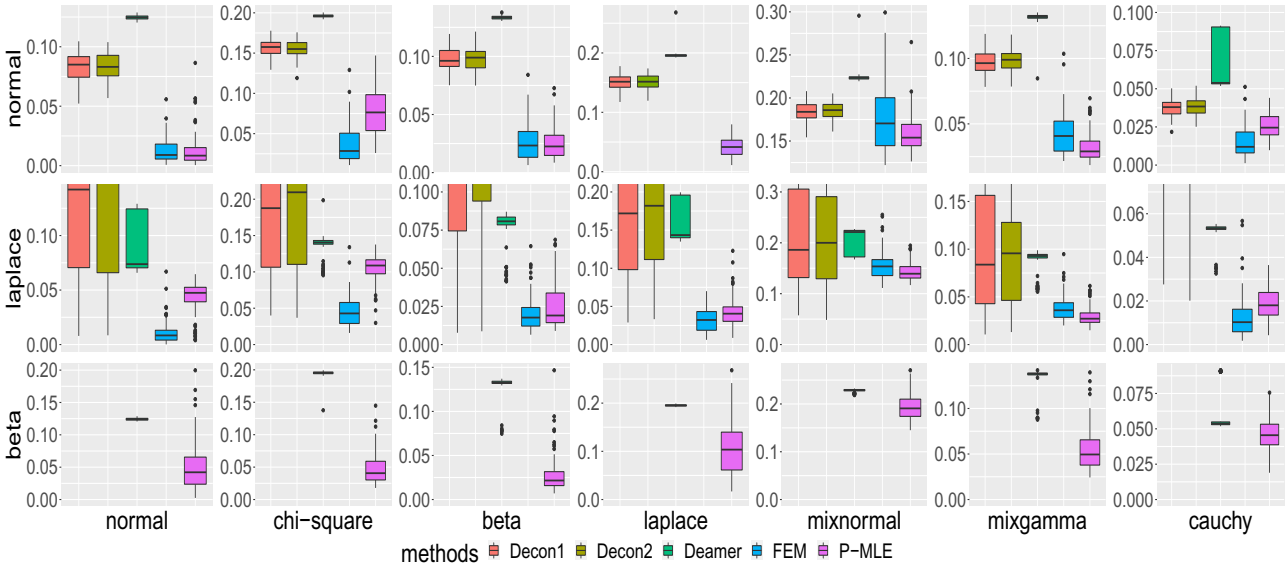
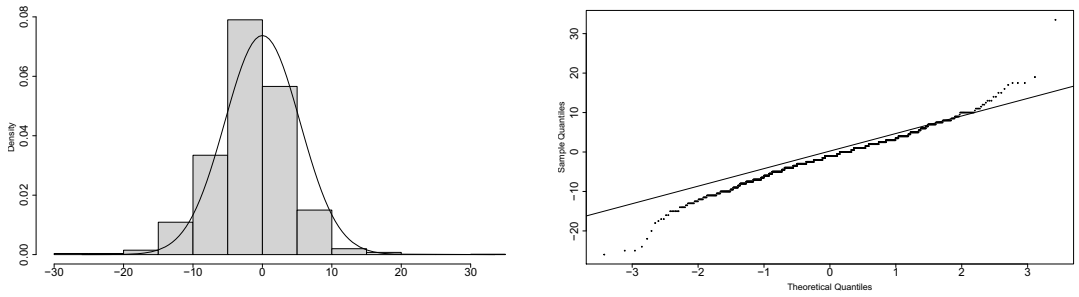


Figure 3.10: Sample distribution of ISE for sample size 300 and SNR 0.25. Each columnn corresponds to one of the seven true distributions in the simulation. Rows correspond to the error distribution. The `decon` package only allows a limited selection of error families, so could not be compared for simulations with a beta error. Some outliers where `decon` produced a large ISE are truncated from these plots. No P-MLE results have been truncated.



(a)
Histogram

(b)
Q-Q plot

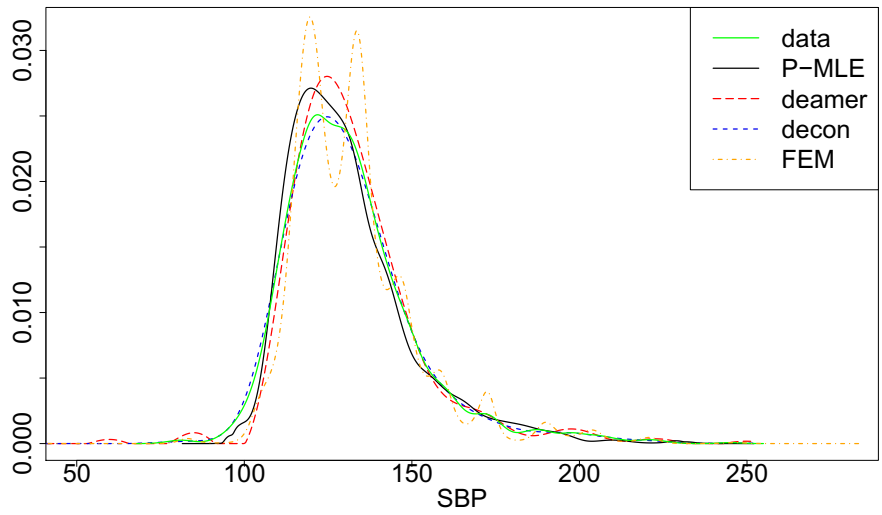
Figure 3.11: Comparison of real data error distribution with a normal distribution

$\frac{SBP_{21}-SBP_{22}}{2} = \frac{e_{21}-e_{22}}{2}$. Assume that the error distribution is symmetric, which is a common assumption for measurement error. Then $\frac{e_{21}+e_{22}}{2}$ and $\frac{e_{21}-e_{22}}{2}$ follow the same distribution. Therefore, we can use $\frac{SBP_{21}-SBP_{22}}{2}$ as the pure error sample for P-MLE and `deamer`. For `decon`, we assume the error distribution is normal with mean zero and estimated variance. Similarly, for `FEM`, we assume the error distribution is normal with estimated mean and variance. Figure 3.11 shows the observed error distribution compared with an estimated normal density. We see that the normal assumption is not unreasonable for this distribution, though the error distribution appears to have heavier tails than the normal distribution.

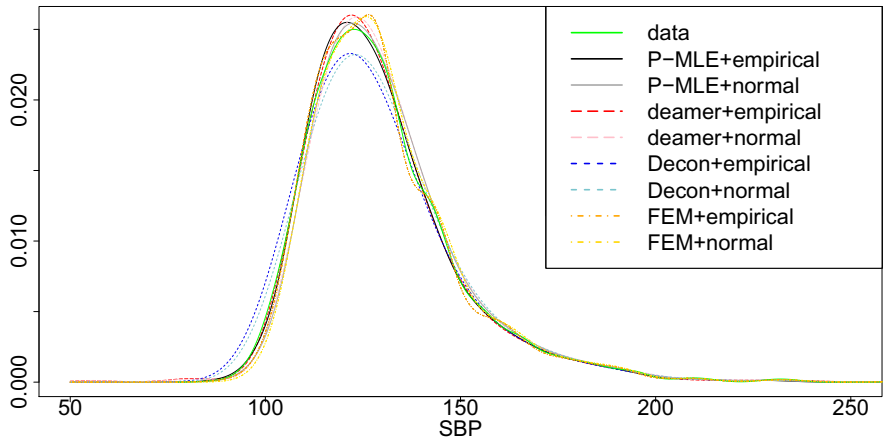
The variance of the error sample is 29.274, and the variance of the observed sample is 395.6506. This suggests an SNR of about 12.5154.

Figure 3.12(a) shows the estimated true distributions by P-MLE and by `deamer`, `decon` and `FEM`. We see that P-MLE and `deamer` both select a much sharper peak and lower variance than the observed data, which is what we should expect to see, since adding noise should increase the variance and produce a less sharp peak. The density estimated by `decon` is extremely close to the observed data, suggesting that `decon` has not removed most of the measurement error. `FEM` is the only method that estimates a bimodal density for the true SBP.

To give a better sense of how well the methods estimate the latent true distribution, we convolve the estimated distributions with the error distribution and



(a)
Deconvolved estimate for density of distribution of SBP. A kernel density estimate of the observed values is shown for comparison, but this is with measurement error, so estimates close to the data estimate have not adequately removed the measurement error.



(b)
Estimated density of SBP distribution convolved with error distribution, compared with a kernel density estimate from the observed data.

Figure 3.12: Real data results

compare the results with a kernel density estimate from the observed data. Figure 3.12(b) compares the convolved estimated distribution for each method, convolving with both a normal measurement-error distribution, and the empirical error distribution. As can be seen, the distributions estimated by P-MLE and `deamer`, when convolved with the empirical error distribution, produce something close to the original data. The convolved distribution estimated by `FEM` is shifted right from the original data. `Decon` estimated a density much closer to the observed data, and as a result, the convolved estimator has higher variance than the observed data. We also see that when a normal error is used, the right tail of the distribution is estimated well by all methods, but the left tail is more challenging.

To give a quantitative measurement of the performance, we calculate the distance between the observed data and the estimated convolved distribution. We use both the Anderson-Darling (AD) distance and the Kolmogorov-Smirnov (KS) distance for measuring the difference. We also calculate the integrated squared difference between the estimated densities and the kernel density estimate from the real data in Figure 3.12(b). Because `deamer` and `decon` do not guarantee that the deconvolved "densities" returned have integral 1, we need to rescale them so that the AD and KS distances can be computed. We do not use Kullback-Leibler divergence because P-MLE directly optimizes likelihood, so choosing a quantity so closely related to the objective function of P-MLE might be considered an unfair advantage to P-MLE, particularly considering that these results are on training data, so there is potential overfitting. The results are in Table 3.10. We see that, as expected, `decon` produces bad results, and that convolving with the empirical error distribution gives a closer result to the observed data than convolving with the normal distribution. For the empirical error distribution, `deamer`, P-MLE and `FEM` perform similarly with P-MLE being better by some measures and `deamer` better under other measures. This is consistent with the simulation results, where for larger sample sizes, the difference in performance between `deamer`, `FEM` and P-MLE was small. We also see that the density functions estimated by P-MLE, `FEM` and `deamer` are quite different, but that the convolutions with the error density are much closer. This is the identifiability issue in the convolution problem, with two distributions having a very similar convolution with the measurement error

Table 3.10: Difference between empirical distribution and convolved estimated distribution

	Normal Error			Empirical Error		
	A.D.	K.S.	ISE	A.D.	K.S.	ISE
P-MLE	2.632	0.0267	0.0135	0.778	0.0085	0.0051
Deamer	3.140.	0.0302	0.0168	0.974	0.0071	0.0056
Decon	4.602	0.0228	0.0272	10.598	0.0412	0.0330
Decon (with FFT)	4.722	0.0237	0.0277	10.648	0.0414	0.0331
FEM	4.196	0.0329	0.0185	0.285	0.0101	0.0087

distribution.

3.7 Conclusion

We have developed a deconvolution method for additive error based on maximum penalized log-likelihood estimation with a smoothness penalty on the estimated density. The smoothness penalty we use has previously been used to good effect in smoothing spline fittings. Our method is applicable to either continuous or discrete error distributions. In cases where the error distribution is unknown, this allows us to substitute the empirical distribution from a pure-error sample.

We have proved that our P-MLE method is consistent. We have also provided methods to address the practical optimization difficulties which arise. In extensive simulation studies, and a real-data example, we have shown that our method has much better performance than existing methods, particularly when sample size and SNR are small. If faster computation is necessary, we provided a quick heuristic to choose the tuning parameter without cross-validation, and showed that with this heuristic, P-MLE produces good results, but with additional time to tune the penalty parameter λ_n by cross-validation, it will perform even better.

There are a number of directions in which the method can potentially be improved in future. Firstly, for practical purposes, we replaced the infinite number of non-negativity constraints by a finite subset of non-negativity constraints at a set of evenly spaced points. The solution to this constrained optimization problem might not satisfy all the non-negativity constraints. It seems likely that with carefully chosen constraints, we should be able to ensure that the estimated density satisfies all the non-negativity constraints. If this is the case, then it should be

possible to develop an adaptive algorithm for choosing the correct points at which to impose non-negativity constraints.

Another issue that could be studied in the future is truncation. From the theory developed, we see that convergence depends upon the estimated support $|u - l|$ not increasing too fast as $n \to \infty$. For common light-tailed distributions, this will almost surely happen. However, for heavy-tailed distributions, the estimated support could grow too fast to achieve consistency. This problem could be resolved via an appropriate truncation method where certain data points are removed from the sample so that the rate of growth of the estimated support is controlled. This would be expected to improve large-sample performance in cases where the underlying true distribution is heavy-tailed. Given that P-MLE performed well in the Cauchy simulation results in Table 3.7, it seems that this is more of a theoretical issue than a practical concern.

Chapter 4

Rank Selection for Non-negative Matrix Factorization

4.1 Introduction

From Chapter 2, we know the only tuning parameter in the NMF algorithm is the rank of the feature matrix, k, which is the number of features. The interpretation of k is the number of underlying sub-structures or subcommunities extracted from the data. The selection of k not only affects the performance of NMF but also relates to the interpretation of NMF results. Too small k may lead to key features missing and too large k value may have overfitting issues. It is therefore necessary to select an appropriate k value for the data. The purpose of this chapter is to develop a new method for selecting this rank k.

The usual approach to NMF rank selection is to use expert knowledge [36]. However, for many analyses in practice, no-one has the insight needed to select the rank. Even if there are experts with a good intuition of what rank is appropriate, being able to support the choice with statistical evidence is valuable.

When the expert knowledge is unavailable, a popular approach for rank selection is to compute some model measures for a range of rank values and choose the rank value according to the measure's criteria. Cophenetic correlation coefficient and Dispersion are the most commonly used measurements [82]. The Cophenetic correlation coefficient method measures the reproducibility of the clustering of observations based on the weight matrix of NMF for many runs with different initial values. Because of the multimodality of the likelihood or squared errors, runs of NMF with different initial values can produce different results. The idea is that for the appropriate number of features, there should be a clear optimum, so runs with different initial values should give the same solution. If the number of features is too large, then the optimal likelihood or mean squared error will vary between runs, leading to a smaller Cophenetic correlation. The method selects the rank

value where the Cophenetic correlation has a steep drop off [11]. The Dispersion method uses a similar procedure to the Cophenetic coefficient method but calculates Dispersion coefficients instead [53]. These two methods both measure the robustness of the clustering analysis results, instead of the goodness-of-fit of the NMF results to the data, thus could lead to suboptimal solutions [31]. Other ideas include comparing residuals, sparseness and Description Length [101]. These methods are all ad hoc in that they select the rank according to the plots based on intuition instead of a solid statistical inference results. The results will be inaccurate when a clear drop off point is not available.

Another popular approach is to use cross-validation. For example, using 2-fold cross-validation to split the data randomly into two halves and apply NMF to the two parts separately. The rank minimizing the average reconstruction error between the two parts is selected for NMF [100]. Lin [69] introduces a rank selection method based on imputation. They randomly delete 30% of the data entry from the original data each time and treat them as missing values in the NMF procedure. The rank is chosen by minimizing the mean square errors of the reconstruction of the missing entries. Their NMF algorithm to handle missing values is available in the R package `NNLM` [69]. Compared with the Cophenetic correlation coefficient method, Dispersion method and 2-fold cross validation method, the NNLM imputed mean squared error method estimates the rank more accurately when applied to Normal synthetic data with different properties [82].

There are also some other rank selection methods such as the Bayesian method [41] and singular value decomposition. The Bayesian method requires an assumption that the rank should be small [41], which is not necessarily true for real data. Singular value decomposition chooses the rank where the singular values become small [36]. The challenge of the method arises when there isn't a clear drop to zero singular value among all ranks. Moreover, the rank of singular values is not closely related to the rank of non-negative matrix factorization.

In this chapter, we develop a goodness-of-fit test based on a deconvolved bootstrap distribution and use the test to choose an appropriate rank for NMF. The application of deconvolution to bootstaps with optimisation error appears to be novel. We then compare our rank selection method with bootstrap distribution

without deconvolution and NNLM on a range of simulated data and real data. We find that our method produces accurate and stable estimation of ranks for NMF.

4.2 Methods

Our rank selection for NMF is based on sequentially performing the following hypothesis test:

H_0: the rank of the feature matrix is k.

H_a: the rank of the feature matrix is at least $k+1$.

After applying the goodness-of-fit test, if H_0 is rejected, let $k = k+1$ and repeat the test until H_0 can not be rejected at the given significance level.

4.2.1 Likelihood ratio test

Suppose $L(k)$ is the likelihood of the factorization using rank k NMF and $L(k+1)$ is the likelihood of the rank $k+1$ NMF results. We construct a likelihood ratio test. Under null hypothesis H_0, the test statistic is given by:

$$\begin{aligned} \lambda_{LR} &= -2\ln\left(\frac{\sup L(k)}{\sup L(k+1)}\right) \\ &= -2(\ln(\sup L(k)) - \ln(\sup L(k+1))) \end{aligned} \tag{4.1}$$

Here sup denotes the supremum and ln represents natural logarithm.

According to standard statistical theory, under certain conditions, the likelihood ratio statistics asymptotically follow a chi-squared distribution. However, because of the non-negative boundary, the conditions for the asymptotic null chi-squared distributions are not met. Moreover, due to the computational difficulty, it is most common that the global maximization of the likelihood is not achieved by applying the NMF algorithm with a single set of initial values. Thus, the results of a given run of the NMF algorithm are subject to computational measurement errors for the likelihood ratio statistic. For the first difficulty, we can use a bootstrap method to compute a correct null distribution. However, the computational measurement error in the likelihood ratio statistic for the bootstrap samples makes

the hypothesis testing method lose its power due to the long tail in the null distribution. We develop a method to get a measurement error deconvolved test to improve the power for the likelihood ratio test in NMF rank selection. This work is an application of our work on a general method for measurement error deconvolution density estimation using penalized likelihood in Chapter 3

4.2.2 Direct testing with bootstrapped null distribution (Boot-test)

For our hypothesis test, we can use a parametric bootstrap to obtain the null distribution of the test statistic. Under the null hypothesis, the data is a realization of a random sample from a distribution F with mean a linear combination of the k features. The k features and their coefficients are estimated by rank k NMF of the original data. In the bootstrap, samples of random datasets are drawn from $\hat{F}$. These samples have the same dimension as the original data. When the data is Poisson distributed, the mean is the only parameter of the distribution. For Normal data, assuming each entry of the data is independently Normally distributed and shares the same variance, both mean and variance need to be estimated from the original data. The variance can be estimated from the residuals of the k rank NMF model. For example, if the rank k NMF gives $X \approx T_{p\times k}W_{k\times n}$. TW is the estimated mean parameter for the Poisson or Normal distributions. The variance of the Normal distribution can be estimated by $\frac{1}{np}\sum_{i,j}(X-TW)^2_{ij}$. The negative entries in the bootstrapped sample are replaced by 0 for Normal distributed data. We apply rank k and rank $k+1$ NMF on each bootstrapped sample to get the test statistic λ^*, from which the empirical null distribution of λ can be obtained.

Unfortunately, this hypothesis test with bootstrapped null distribution does not perform well at selecting the rank of NMF since the likelihood surface for NMF is often multimodal, and it is common for the algorithm to converge to a suboptimal factorization. This creates noise in the bootstrapped null distribution, thus reducing the power of the test.

To improve the performance, we can rerun the algorithm with multiple starting values on the original data to choose the best log-likelihood. The mean $\hat{F}$ is estimated as the result of NMF that has the best log-likelihood. For Normal data, the variance is the average of variances estimated from each run of NMF to avoid

overfitting. We also run NMF with multiple initial values on each bootstrap sample and choose the best log-likelihoods to build the null distribution. The procedure is shown in Algorithm 1.

Algorithm 1 Pseudocode for Boot-test

$k \leftarrow 0$
while p-value is significant **do**
  $k \leftarrow k + 1$
  for m different initial values **do**
    apply rank k NMF and rank $k+1$ NMF to the original data
  end for
  $l_0(k) \leftarrow$ largest loglikelihood of all rank k NMF results;
  $l_0(k+1) \leftarrow$ largest loglikelihood of all rank $k+1$ NMF results;
  compute T_0 and W_0 when rank k NMF has loglikelihood $l_0(k)$
  $\lambda \leftarrow -2(l_0(k) - l_0(k+1))$
  sample bootstrap datasets from distribution with mean $T_0 W_0$ (and variance estimate if data are assumed to be Normal)
  for all bootstrap datasets **do**
    for m different initial values **do**
      apply rank k NMF and rank $k+1$ NMF to the bootstrap data
    end for
    compute λ^* as negative two times the difference of the best loglikelihood of all rank k NMF results and the best loglikelihood of all rank k+1 NMF results
  end for
  get p-value;
end while

The simulation results for this method with different numbers of initial values and different true rank values are shown in Section 4.3.

4.2.3 Testing with deconvolved bootstrap null distribution (Decon-boot-test)

The major problem with the method described above is that to ensure a good bootstrap distribution, we need to fit each bootstrap sample with a large number of different initial values, which is computationally expensive. We therefore develop a new approach to more efficiently compute the bootstrap distribution. The idea is to treat the convergence error as measurement error in the log-likelihood ratio statistic. We can then use a well-developed deconvolution method to estimate the null distribution of the statistic. Deconvolution is a method to estimate a

distribution from data with additive measurement error. In our case, if we run only one set of initial values for NMF for each bootstrap sample for each fixed k, then we can directly compute the estimated maximum loglikelihood $l(k) = l_0(k) + e(k)$, where $l_0(k)$ is the globally maximized log-likelihood value (unobserved) and $e(k)$ is the measurement (convergence) error. Thus

$$\begin{aligned} \lambda^* &= \lambda_0^* + e \qquad (4.2) \\ &= -2(l_0(k) - l_0(k+1) + e(k) - e(k+1)) \\ &= -2(l_0(k) - l_0(k+1)) + \{-2[e(k) - e(k+1)]\} \qquad (4.3) \end{aligned}$$

where λ_0^* is the true likelihood ratio statistic without convergence error and $e = -2(e(k) - e(k+1))$. The deconvolution method will estimate the distribution of λ_0^* when λ_0^* is not observable but $\lambda_0^* + e$ is available.

Most deconvolution approaches are based on the Fourier transformation. While its mathematical solution is very neat, estimation of the characteristic function from data is somewhat unstable, and division by possibly small Fourier coefficients for the error data can greatly magnify errors. So in this chapter, we use the P-MLE deconvolution method developed in Chapter 3 which is based on maximizing a penalized log-likelihood of the data (here λ^*) with a smoothness penalty. P-MLE also has the advantage over many other competing methods that its implementation can estimate the measurement error distribution non-parametrically from a pure measurement error sample (see Page 68 Section 3.2). This is important because the convergence error is unlikely to follow a standard parametric family of distributions.

Figure 4.1 shows an example of null distribution of likelihood ratio statistic without deconvolution and after deconvolution. The deconvolved distribution of likelihood ratio statistic has a much smaller vairance than the bootstrapped null distribution of likelihood ratio statistic when NMF with single initial value applied to each bootstrap sample as convergence error is removed by deconvolution.

The sampling procedure for bootstrapped datasets is the same as in Section 2.2. However, for each sampled data, rank k NMF and rank $k+1$ NMF are run only once to get $\lambda_0^* + e$. In order to get a pure error sample, we perform rank k and rank

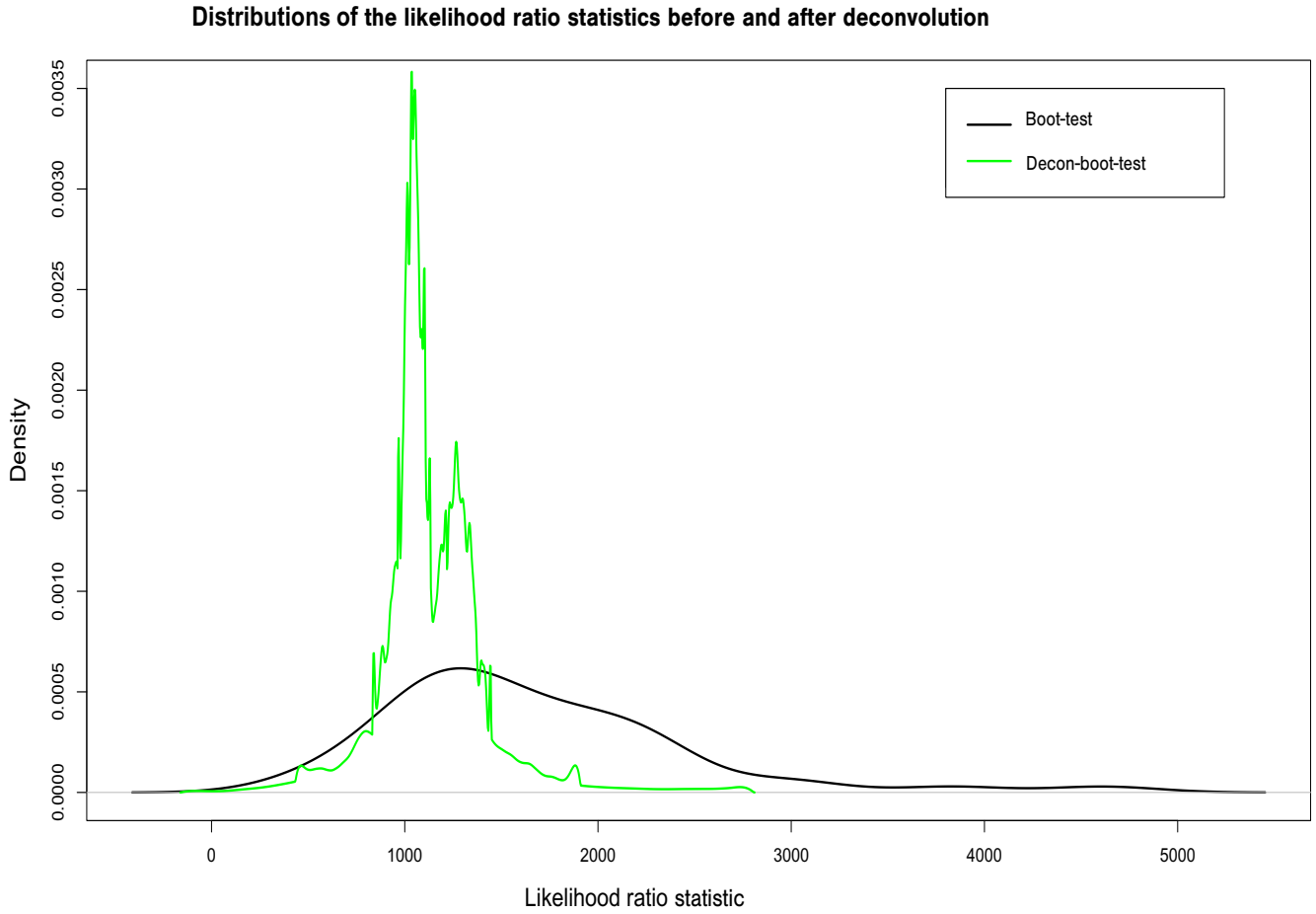


Figure 4.1: Null distribution of the likelihood ratio statistics for rank 4 NMF vs. rank 5 NMF. The data is generated from rank 4 Poisson NMF. The black curve is the kernel estimation of bootstrapped null distribution when apply rank 4 and rank 5 NMF with single initial value for each bootstrap sample with bootstrap size as 50. The green curve is the deconvolved density of likelihood ratio statistic after applying P-MLE deconvolution method to the 50 bootstrapped Likelihood ratios.

$k+1$ NMF with multiple initial values for an additional bootstrap sample. Since we are trying to maximize the log-likelihood, we will assume that the largest likelihood from multiple runs of different initial values is the true maximum likelihood. If the number of different initial values is large enough, this should be close to be true. The measurement error samples of $e(k)$ and $e(k+1)$ can be obtained from the largest loglikelihood among different initial values minus the loglikelihoods of all NMF runs. The pure error sample for e can be obtained as negative two times the difference between $e(k)$ and $e(k+1)$ for all different pairs of $e(k)$ and $e(k+1)$. Our method is implemented in the R package for Decon-boot-test, which is available on `CRAN` with the name `DBNMFrank` [12].

A complete description of the procedure is given in Algorithm 2:

The computation time for each hypothesis test mainly consists of two parts: the NMF calculation part and the deconvolution part. For the deconvolution part, the computation time is fixed for different data. As the time complexity of NMF multiplicative update algorithm [94] is #iterations $\times\, O(npk)$ [68], where n is the sample size, p is the number of variables and k is the rank value, the computation time for the NMF part is proportional to the sample size and number of variables of the data. Also, when the true rank increases, more rounds of hypothesis tests will be applied, which will lead to longer computation time.

4.3 Simulation

In this section, we compare the performance of Boot-test and Decon-boot-test introduced in Sections 4.2.2 and 4.2.3 and NNLM [69] which is based on missing data imputation, on Poisson NMF data, Normal NMF data and non-NMF structure data. NNLM minimizes Kullback-Leibler divergence loss to estimate Poisson data's factorization and square error loss to estimate Normal data's factorization. In the simulation, following the suggestions from [69], we randomly set 30% of the data entries as NA and apply NNLM with a sequence of rank values from 1 to 50 to impute the missing data. The procedure is repeated 10 times and the rank value that minimizes the average loss (KL divergence for Poisson data and MSE for Normal data) of the imputed part is selected as the rank for the data.

For the Boot-test and Decon-boot-test methods, we find that the bootstrap size

Algorithm 2 Pseudocode for Decon-boot-test

$k \leftarrow 0$
while p-value is significant **do**
  $k \leftarrow k + 1$
  for m different initial values **do**
    Applying rank k NMF and rank $k + 1$ NMF to original data
  end for
  $l_0(k) \leftarrow$ largest loglikelihood of all rank k NMF results;
  $l_0(k + 1) \leftarrow$ largest loglikelihood of all rank $k + 1$ NMF results;
  compute T_0 and W_0 when rank k NMF has loglikelihood $l_0(k)$
  $\lambda \leftarrow -2(l_0(k) - l_0(k + 1))$
  sample bootstrap datasets from distribution with mean T_0W_0 (and variance estimate if data are assumed to be Normal)
  for all bootstrap datasets **do**
    compute λ^* by applying rank k NMF and rank $k + 1$ NMF on bootstrap datasets.
  end for
  for m different initial values **do**
    compute $l_i(k)$ by applying rank k NMF on a bootstrap sample;
    compute $l_j(k + 1)$ by applying rank $k + 1$ NMF on the same bootstrap sample;
  end for
  for i $\leftarrow$ 1 **to** m **do**
    $e_i(k) \leftarrow (max_{\mathrm{i}}(l_i(k))) - l_i(k)$;
  end for
  for j $\leftarrow$ 1 **to** m **do**
    $e_j(k + 1) \leftarrow (max_{\mathrm{j}}(l_j(k + 1))) - l_j(k + 1)$;
  end for
  $e \leftarrow -2(e_i(k) - e_j(k + 1))$ for all i and j;
  compute null distribution of λ_0^* by applying P-MLE on λ^* and e;
  get p-value;
end while

doesn't affect the rank estimates too much on the simulation data. We only show the simulation results when bootstrap size is 50 in this section. Also, NMF with 50 different initial values consistently converges to the optimal likelihood when running from simulated data. So for the Boot-test, we follow procedure 1 and set $m = 1$, $m = 3$, $m = 10$, $m = 30$ and $m = 50$ for each dataset and each bootstrap sample. For the Decon-boot-test, we follow procedure 2 and set $m = 50$.

We generate 50 replicates for each scenario. For the generated NMF data, the feature matrix is fixed in each scenario and the weight matrix is randomly generated in each replicate. All three methods are applied to the same data in each replicate. The significance level used for Boot-test and Decon-boot-test in the simulations is 0.1.

4.3.1 Poisson data simulation

We simulate a range of Poisson NMF scenarios with true rank equal to 2, 4, 6, 8, 10 and 30, and a scenario where there is no NMF structure. For true rank 2, 4, 6, 8 and 10, we simulate 100 observations, while for the true rank 30, we simulate 300 observations. For the no NMF structure scenarios, we simulate 131 observations. For rank 2 and rank 4, we simulate five scenarios to assess the sensitivity of the methods when one of the types is close to a non-negative linear combination of the others. For the higher true ranks, we simulate one scenario for each true rank. For ranks 6, 8 and 10, all features used in the simulation are estimated from the healthy individuals in the Qin dataset [89], using Poisson NMF with the given rank. One feature for the true rank 2 simulation and three features for the true rank 4 simulation are also estimated from the healthy individuals in the Qin dataset using NMF with ranks 1 and 3 respectively. The Qin data, as mentioned in Chapter 2, is a human gut metagenomic dataset extracted from 99 healthy people and 25 IBD patients. For rank 30, the features used in the simulation are calculated by applying rank 30 NMF to Person 2's gut data in the moving picture data [14] which also used in Chapter 2. After removing rows consisting of all 0's, the total number of different OTUs is 3131 for Person 2's gut data. Each feature is normalized to sum to one for all scenarios.

For the rank 4 simulations, the fourth feature is chosen in the space spanned

by the other three features and a unit vector with all elements the same, such that the perpendicular from the fourth feature to the plane of the other three features passes through the centroid of these three features. We vary the distance of this fourth type from the plane of the other three to assess the sensitivity of the methods in cases where one type is almost a linear combination of the other three.

For rank 2 simulations, the second feature is on the line in the space spanned by the first feature and a unit vector with all elements the same and in the direction from the first feature to the unit vector.

For rank 2 and rank 4, the further the distance is, the more distinct these features are. The distances are set to be 0, 0.0005 0.0008, 0.001, 0.0015 and 0.002. When the distance is 0, the feature matrix degenerates to a lower rank matrix. Thus, the rank 4 feature matrix degenerates to a rank 3 feature matrix and the rank 2 feature matrix becomes rank 1.

For all Poisson NMF data simulations, columns of the weight matrix are generated independently from a uniform distribution from 0 to 1. We adjust each column of the weight matrix to have a fixed sum sampled from the column sums for the healthy group of the Qin data for true NMF rank 2, 4, 6, 8 and 10. The weight matrix is rescaled to have fixed sum sampled from the column sums for Person 2's gut data of the moving picture data for true NMF rank 30. For microbiome data, these column sums are referred to as sequencing depth. Multiplying the feature matrix and the weight matrix, we get the parameters for the Poisson distribution. Each entry of the data matrix simulated for the Poisson NMF data is generated from a Poisson distribution with the Poisson mean given by the corresponding entry of TW. The estimated results of Boot-test, Decon-boot-test and NNLM in 50 replicates of different scenarios are summarized in Table 4.1, Table 4.3 and Table 4.5. To compare the computational costs of the Boot-test and Decon-boot-test, we recorded the average time required to calculate the estimated rank for 50 replicates in each scenario in Table 4.2, Table 4.4 and Table 4.6. Both methods are coded in R (64-bit 4.0.3) and run on 1 node of the Graham Compute cluster of Canada with 1 Intel Xeon at 2.1Ghz with 2G of RAM and 1 core. The NNLM method is not compared here as it uses a different NMF algorithm while estimating the rank. The efficiency of the NMF algorithm is closely related to the

computation cost of the rank selection method.

To compare the ability of Decon-boot-test and NNLM to detect Poisson data without NMF structure, we design a no NMF structure Poisson data simulation. The dimension of the no NMF structure Poisson data is the same as Person 1's gut data from the moving picture dataset, which has 1864 variables and 131 observations. In each replicate, each row of the mean of the no NMF structure Poisson data is generated from a log-multivariate Normal distribution. The mean of the multivariate Normal is a vector of length 1864 whose entries are independently simulated following a Normal distribution with mean 4 and standard deviation 3. For each sample from the multivariate Normal distribution, the mean is rescaled so that the sum of all means is equal to the log of the total sequencing depth from the corresponding sample in Person 1's gut data. The covariance matrix of the multivariate Normal is generated by 1864 eigenvalues evenly spaced from 3×10^{-7} to 1 and orthogonal eigenvectors uniformly distributed over the 1863-dimensional sphere. If the Poisson data doesn't have NMF structure (which can be thought of as having rank equal to sample size), then rank selection methods should select a high rank (which doesn't satisfy the NMF rank selection rule $k < \frac{n \times p}{n+p}$ [63]). The average estimated ranks and standard deviations for the 50 replicates of the two methods are shown in Table 4.5.

The Boot-test results in these tables indicate that for Poisson NMF data, the accuracy of Boot-test estimates increases with the number of initial values used for each NMF when $d = 0.0005$ and $d = 0.0008$ and is stable in most other cases. The computation time of Boot-test is positively related to the number of different initial values used for each NMF (m). Decon-boot-test's computation time is between the time spent by $m = 3$ and $m = 10$ Boot-test. But Decon-boot-test estimates the ranks more accurately than Boot-test in all scenarios except for true rank 4 with $d = 0.0005$. When the true rank is 4 and $d = 0.0005$, the fourth type is very closed to the plane of the other three types, which means the 3 types can explain the data almost as good as four types. As the Decon-boot-test is more conservative and Boot-test tends to over-estimate the rank, Boot-test chooses rank 4 in more replicates while Decon-boot-test chooses rank 3 in more replicates in this scenario. In Table 4.1 and Table 4.3, the Decon-boot-test selects the true ranks in

more replicates than NNLM especially when the distance from one feature to other features is small, for example $d = 0.0005$ and $d = 0.0008$. In Table 4.5, Decon-boot-test is comparable with the NNLM method when the NMF rank is low. For the high rank case, summaries of the estimated ranks show that both methods underestimate the rank, but Decon-boot-test gives a fairly close estimate for the rank, while NNLM estimates a very low rank. When there is no NMF structure, Decon-boot-test selects a high rank, while NNLM selects 1 for all replicates, which is somewhat misleading.

Table 4.1: Total number of times the true rank is selected out of 50 replicates and the 50 rank estimates' averages and standard deviations when the true rank is 2 for Poisson NMF data when bootstrap size is 50.

d		0		0.0005		0.0008		0.001		0.0015		0.002	
Boot-test	$m = 1$	47	1.06(0.24)	17	1.54(0.68)	22	2.04(0.75)	31	2.00(0.62)	44	2.00(0.35)	46	2.06(0.37)
	$m = 3$	46	1.08(0.27)	28	2.16(0.65)	39	2.10(0.46)	45	2.12(0.39)	40	2.20(0.40)	42	2.16(0.37)
	$m = 10$	47	1.06(0.24)	43	2.20(0.53)	45	2.08(0.28)	44	2.06(0.25)	43	2.14(0.41)	47	2.06(0.24)
	$m = 30$	47	1.06(0.23)	43	2.16(0.42)	45	2.12(0.39)	44	2.12(0.33)	46	2.08(0.27)	45	2.10(0.30)
	$m = 50$	47	1.06(0.24)	40	2.28(0.54)	46	2.08(0.27)	44	2.14(0.40)	45	2.16(0.55)	45	2.04(0.20)
Decon-boot-test		50	1.00(0.00)	50	2.00(0.00)	50	2.00(0.00)	50	2.00(0.00)	50	2.00(0.00)	50	2.00(0.00)
NNLM		50	1.00(0.00)	39	2.12(0.52)	43	2.02(0.38)	44	2.04(0.35)	48	2.04(0.18)	47	2.06(0.24)

m: number of different initial values used for each NMF.
d: distance from one feature to a linear combination of other features.
First column for each value of d in grey: the number of times the true rank is selected out of 50 replicates.
Second column for each value of d: the average of the 50 estimated ranks (before the bracket) and the standard deviation of the 50 estimates (in the bracket).
The most accurate estimates are highlighted in yellow.

4.3.2 Normal data simulation

We generate Normal NMF data with true rank equal to 2, 3, 4, 6, 8 and 10. For each scenario, we simulate 50 replicates.

Each element in the feature matrix for the Normal NMF data is independently sampled from a gamma distribution with shape parameter 3 and rate parameter 2, then multiplied by a Bernoulli random variable with parameter 0.7 to control the sparsity of the feature matrix. All replicates in the same scenario share the same

Table 4.2: Computation time per replicate (average among the 50 replicates) in hours when the true rank is 2 for Poisson NMF data when bootstrap size is 50.

d		0	0.0005	0.0008	0.001	0.0015	0.002
Boot-test	$m = 1$	0.2	0.3	0.4	0.5	0.4	0.5
	$m = 3$	0.8	1.2	1.0	1.3	1.7	1.6
	$m = 10$	1.8	4.6	3.3	3.1	4.2	3.3
	$m = 30$	5.3	9.3	9.5	9.0	8.5	9.2
	$m = 50$	9.0	22.8	17.5	18.4	19.0	17.2
Decon-boot-test		1.0	2.0	1.3	1.4	2.0	2.0

m: number of different initial values used for each NMF.
d: distance from one feature to a linear combination of other features.

Table 4.3: Total number of times the true rank is selected out of 50 replicates and the 50 rank estimates' averages and standard deviations when the true rank is 4 for Poisson NMF data when bootstrap size is 50.

d		0		0.0005		0.0008		0.001		0.0015		0.002	
Boot-test	$m = 1$	45	3.10(0.30)	10	3.28(0.54)	14	3.66(0.85)	18	4.00(0.81)	35	4.20(0.57)	42	4.14(0.45)
	$m = 3$	45	3.12(0.39)	16	3.48(0.65)	31	4.24(0.62)	38	4.22(0.51)	43	4.14(0.35)	43	4.14(0.35)
	$m = 10$	39	3.22(0.42)	30	3.92(0.63)	34	4.26(0.44)	34	4.28(0.50)	39	4.17(0.38)	42	4.16(0.37)
	$m = 30$	40	3.24(0.52)	35	4.24(0.69)	39	4.28(0.57)	41	4.20(0.45)	42	4.16(0.37)	41	4.18(0.39)
	$m = 50$	42	3.22(0.58)	39	4.24(0.59)	38	4.28(0.54)	37	4.28(0.50)	42	4.20(0.49)	43	4.16(0.42)
Decon-boot-test		50	3.00(0.00)	27	3.54(0.50)	50	4.00(0.00)	50	4.00(0.00)	50	4.00(0.00)	50	4.00(0.00)
NNLM		50	3.00(0.00)	18	3.76(0.77)	37	4.14(0.50)	43	4.06(0.37)	49	4.02(0.14)	48	4.04(0.20)

m: number of different initial values used for each NMF.
d: distance from one feature to a linear combination of other features.
First column for each value of d in grey: the number of times the true rank is selected out of 50 replicates.
Second column for each value of d: the average of the 50 estimated ranks (before the bracket) and the standard deviation of the 50 estimates (in the bracket).
The most accurate estimates are highlighted in yellow.

feature matrix. The dimension of the features is 2780 initially. After removing rows with all zero elements in the feature matrix, there are 2544 rows remaining in the type 2 feature matrix and 2721 rows in the feature matrices for other ranks.

The 100 columns of the weight matrix W are generated independently from a Dirichlet distribution with parameters all set to be 1.5 and rescaled by a factor of 10.

Each entry of the data matrix is simulated following a Normal distribution with mean given by the corresponding entry in the matrix product TW and variance 1. Negative entries in the data are replaced by 0 to generate nonnegative data.

Table 4.7 shows for Normal NMF data, the boot-test's accuracy increases when

Table 4.4: Computation time per replicate (average among the 50 replicates) in hours when the true rank is 4 for Poisson NMF data when bootstrap size is 50.

d		0	0.0005	0.0008	0.001	0.0015	0.002
Boot-test	$m = 1$	0.6	0.7	0.8	0.8	1.0	1.2
	$m = 3$	2.0	2.1	2.6	2.5	2.2	2.2
	$m = 10$	9.0	9.4	13.2	13.4	12.0	11.8
	$m = 30$	26.4	31.2	32.4	28.4	27.7	28.0
	$m = 50$	44.0	53.0	55.3	55.1	56.2	54.0
Decon-boot-test		6.4	8.6	10.5	9.0	11.2	10.4

m: number of different initial values used for each NMF.
d: distance from one feature to a linear combination of other features.

Table 4.5: Total number of times the true rank is selected of 50 replicates and the 50 rank estimates' averages and standard deviations when the true rank is 6, 8 and 10 for Poisson NMF data when bootstrap size is 50.

rank		6		8		10		30	nonNMF
Boot-test	$m = 1$	43	6.16(0.42)	41	8.18(0.39)	36	10.24(0.48)		
	$m = 3$	43	6.14(0.35)	38	8.24(0.43)	41	10.18(0.39)		
	$m = 10$	41	6.18(0.39)	39	8.22(0.42)	44	10.12(0.33)		
	$m = 30$	34	6.32(0.47)	37	8.26(0.44)	38	10.24(0.43)		
	$m = 50$	38	3.32(2.95)	34	8.32(0.47)	40	10.20(0.40)		
Decon-boot-test		50	6.00(0.00)	50	8.00(0.00)	47	10.06(0.24)	0	25.04(1.40)	39.69(8.61)
NNLM		50	6.00(0.00)	45	8.10(0.30)	50	10.00(0.00)	0	2.76(1.76)	1.00(0.00)

m: number of different initial values used for each NMF.
First column under each rank in grey: the number of times the true rank is selected out of 50 replicates. (first column is not available for nonNMF data)
Second column under each rank: the average of the 50 estimated ranks (before the bracket) and the standard deviation of the 50 estimates (in the bracket).
The most accurate estimates are highlighted in yellow.

more sets of initial values are used for each NMF. Both the Decon-boot-test and NNLM estimate the true ranks for Normal NMF data more accurately than Boot-test. Decon-boot-test and NNLM are both extremely accurate at estimating the rank for all scenarios. Table 4.8 shows the average time required by Decon-boot-test to calculate the estimated rank for the 50 replicates in each Normal NMF data scenario is usually between the computational expense of $m = 3$ and $m = 10$ Boot-test.

Table 4.6: Computation time per replicate (average among the 50 replicates) in hours when the true rank is 6, 8 and 10 for Poisson NMF data when bootstrap size is 50.

rank		6	8	10
Boot-test	$m = 1$	2.8	3.6	6.0
	$m = 3$	6.0	11.5	20.0
	$m = 10$	19.2	37.0	55.2
	$m = 30$	63.9	102.0	207.6
	$m = 50$	101.3	209.9	323.5
Decon-boot-test		8.5	33.0	35.5

m: number of different initial values used for each NMF.

Table 4.7: Total number of times the true rank is selected out of 50 replicates for Normal NMF data and the 50 rank estimates' averages and standard deviations when bootstrap size is 50.

rank		2		3		4		6		8		10	
Boot-test	$m = 1$	47	2.10(0.46)	40	3.02(0.14)	39	4.22(0.42)	40	6.20(0.40)	42	8.16(0.37)	43	10.14(0.35)
	$m = 3$	41	2.22(0.55)	38	3.24(0.43)	42	4.18(0.44)	44	6.12(0.33)	41	8.18(0.39)	46	10.08(0.27)
	$m = 10$	42	2.20(0.49)	40	3.12(0.33)	42	4.16(0.37)	42	6.16(0.37)	47	8.06(0.24)	41	10.18(0.39)
	$m = 30$	44	2.14(0.40)	40	3.22(0.46)	42	4.16(0.37)	43	6.16(0.42)	45	8.10(0.30)	43	10.14(0.35)
	$m = 50$	46	2.08(0.27)	42	3.16(0.37)	46	4.08(0.27)	43	6.14(0.35)	42	8.16(0.37)	42	10.16(0.37)
Decon-boot-test		50	2.00(0.00)	49	3.02(0.14)	50	4.00(0.00)	50	6.00(0.00)	49	8.02(0.14)	50	10.00(0.00)
NNLM		50	2.00(0.00)	50	3.00(0.00)	50	4.00(0.00)	50	6.00(0.00)	50	8.00(0.00)	50	10.00(0.00)

m: number of different initial values used for each NMF.
First column under each rank in grey: the number of times the true rank is selected out of 50 replicates.
Second column under each rank: the average of the 50 estimated ranks (before the bracket) and the standard deviation of the 50 estimates (in the bracket).
The most accurate estimates are labeled as yellow.

4.3.3 Conclusion of simulations

The simulation results show that Decon-boot-test is better than Boot-test in estimation accuracy for both Poisson data and Normal data. By using a large number of starting points, Boot-test can achieve similar accuracy to Decon-boot-test, but is far more computationally expensive with no benefit in accuracy. When the true rank is small, the performance of Decon-boot-test for Normal data is comparable with NNML, and for Poisson data, its performance is better than NNML, especially when the features are hard to distinguish from each other. When the rank is large, both methods fail to find the true rank but Decon-boot-test's estimated rank is much closer to the true rank. When the data is not NMF structured,

Table 4.8: Computation time per replicate (average among the 50 replicates) in hours for Normal NMF data when bootstrap size is 50.

rank		2	3	4	6	8	10
Boot-test	$m = 1$	0.1	0.5	0.5	1.0	2.1	3.5
	$m = 3$	0.4	1.3	1.3	3.0	7.8	10.2
	$m = 10$	2.7	2.5	4.2	10.0	25.8	33.5
	$m = 30$	3.6	6.3	11.7	28.7	69.7	95.7
	$m = 50$	5.7	12.9	23.0	53.6	104.0	186.5
Decon-boot-test		2.0	2.5	3.0	8.3	23.5	33.5

m: number of different initial values used for each NMF.

Decon-boot-test selects very large rank, while NNLM selects rank 1 for all replicates. Thus, Decon-boot-test gives a much clearer indication of the lack of NMF structure, which can be used to better interpret the data.

4.4 Real Data application

We apply our NMF rank selection method to moving picture data [14]. These data sets have previously been analysed using NMF in Section 2.3.2 Chapter 2, but without a good way to select the rank, it is possible that there are undetected biologically important patterns in the data that can be discovered by reanalysing the data using the number of types selected by the Decon-boot-test method. We therefore apply our method to the gut data from both individuals to choose the number of types, and examine the fitted types and weights to obtain biological insights that were not observed in previous analyses of the data using a different number of types. Our method selects NMF rank as 12 for Person 1's gut data and NMF rank as 6 for Person 2's gut data. The computation time is 22.3 hours for Person 1's gut data and 9.3 hours for person 2's gut data.

To examine the microbiome community structure of the gut from the NMF results, we plot the relative abundance of each genus in each feature in Figure 4.2 for Person 1's gut and Figure 4.4 for Person 2's gut. The elements of each feature sum to 1 so the coefficients can be interpreted as the proportions of each OTU in that feature. As the NMF results are usually highly sparse, we use a cut-off of 3% for each type. That is, only those genera with OTUs above 3% composition in at least one type are shown in the plot. The outstanding OTUs

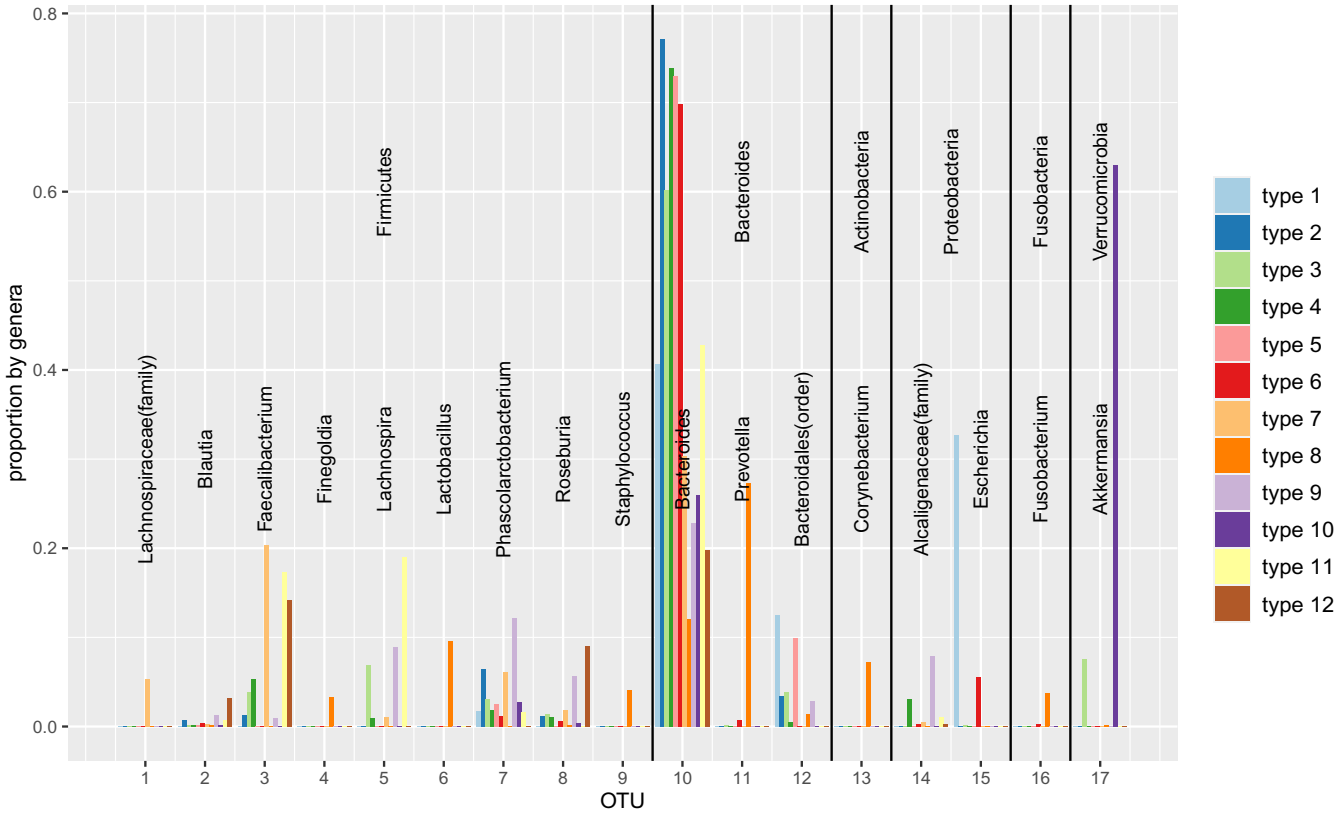


Figure 4.2: Relative abundance of major genera for each type from Person 1's gut feature matrix T. The genera from the same phylum are in the same block which is labeled by their phylum names and the bars are labeled by the genus names or higher level of taxonomic rank if it's unclassified at genus level. Each bar is colored according to its type.

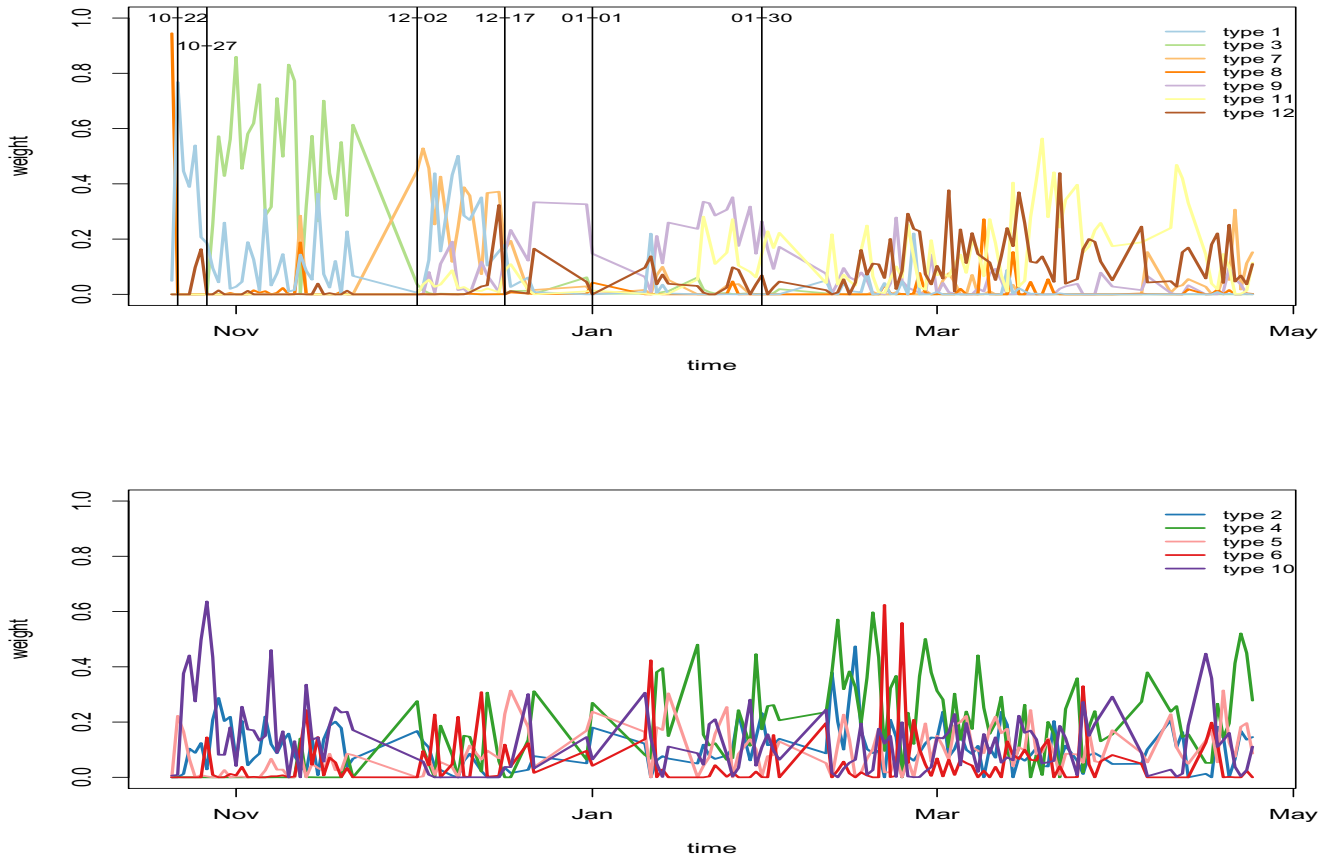


Figure 4.3: Gut weight matrix time series plot for Person 1. For clarity, we separate the 12 time series into two panels. The top plot shows Person 1's gut weight matrix on type 1, type 3, type 7, type 8, type 9, type 11 and type 12 from 12 rank NMF, whose weights are more stable. The bottom plot shows the weight matrix for other types with more fluctuating weights. Each weight is colored the same as its corresponding type.

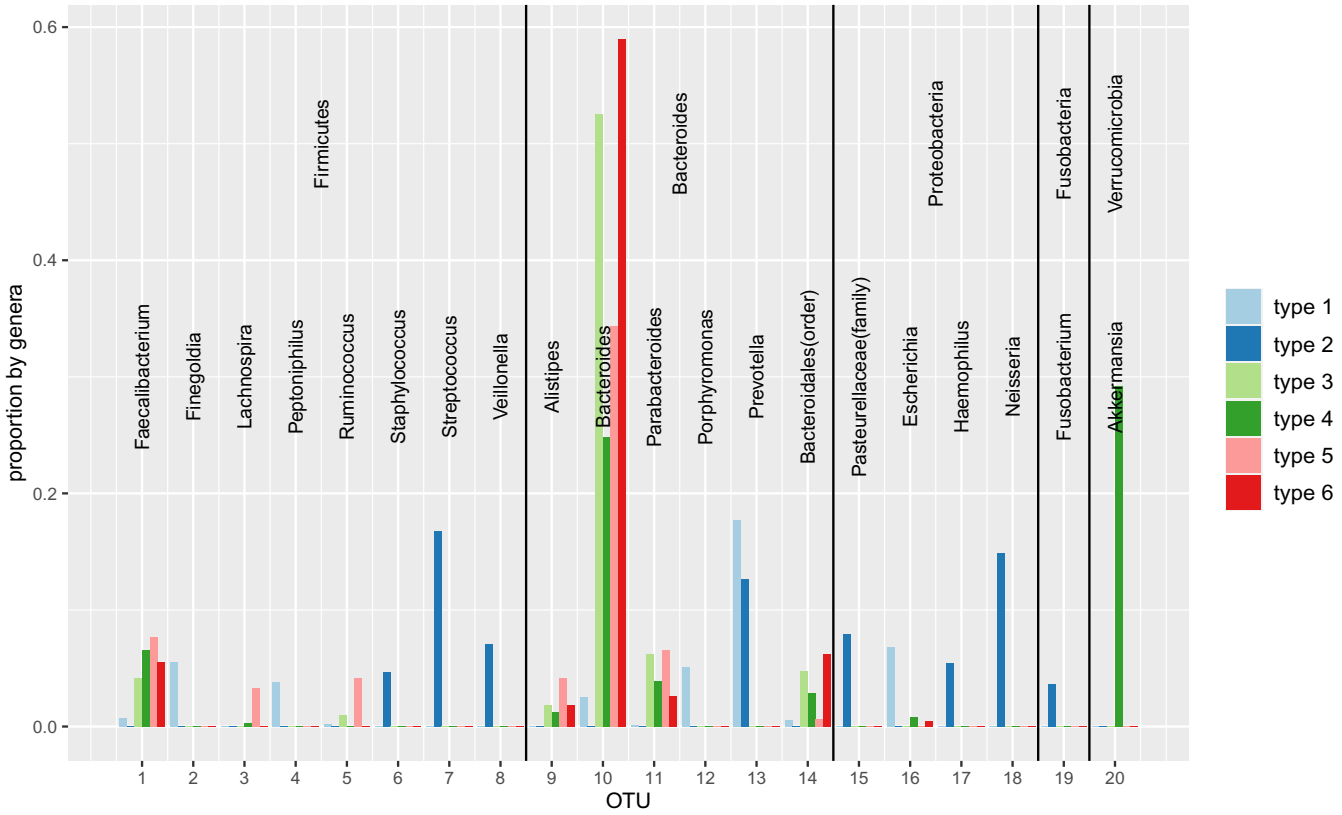


Figure 4.4: Relative abundance of major genera for each type from Person 2's gut feature matrix T. The genera from the same phylum are in the same block which is labeled by their phylum name and the bars are labeled by the genus names or higher level of taxonomic rank if it's unclassified at genus level. Each bar is colored according to its type.

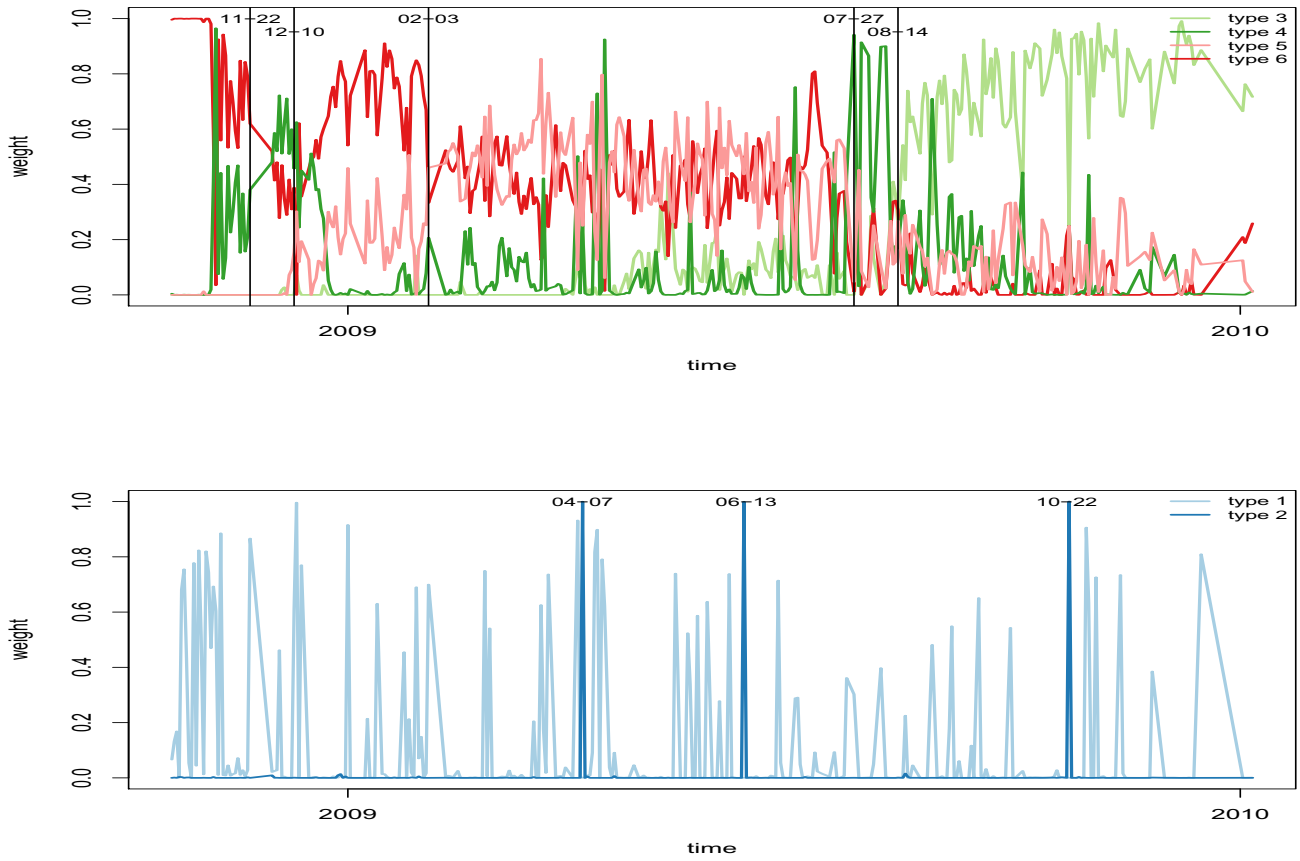


Figure 4.5: Gut weight matrix time series plot for Person 2. The top plot shows Person 2's gut weight matrix normalized on type 3, type 4, type 5 and type 6 from 6 rank NMF. The bottom plot shows the weight matrix for other types. Each weight is colored the same as its corresponding type.

for features of both persons' guts are from the phyla Firmicutes, Bacteroides, Actinobacteria, Proteobacteria, Fusobacteria and Verrucomicrobia. Most of the features have abundant Firmicutes and Bacteroides which is consistent with the fact that Firmicutes and Bacteroidetes constitute 90% of the gut microbiota, while Actinobacteria, Proteobacteria, Fusobacteria, and Verrucomicrobia contribute to the microbial population to a lesser extent [92].

Figure 4.2 shows the abundance of the main genera in Person 1's gut features. We find only 30 out of more than eighteen hundred OTUs are larger than the cut-off of 3%. These 30 OTUs are from 17 genera. From the plot, we see Type 2, 4, 5 and 6 share similar compositions at genus level. They all have extremely high abundance of Bacteroides and much lower abundance of other genera. These types differ in the compositions of other genera. In particular, the secondary genus in the community is Faecalibacterium in Type 4, Phascolarctobacterium in Type 2, an unclassified genus from order Bacteroides in Type 5 and Escherichia in Type 6. Type 10 is dominated by Akkermansia rather than Bacteroides. As lean individuals have more Bacteroides, while obese individuals have more Firmicutes in their intestinal microbiota and the abundance of Akkermansia is negatively correlated to human's weight [103] [108], Type 2, Type 4, Type 5, Type 6 and Type 10 may be healthy communities with regard to body weight. Type 1 mainly consists of Escherichia and an unclassified genus from order Bacteroidales. We know that a number of species of Escherichia are pathogenic [42], so this type may be associated with infection with pathogens. While the species of the OTUs in Escherichia are not classified in the gut data, further research is needed to draw the conclusion. Type 8's major genus is Prevotella whose increased abundance is linked to inflammatory disorders in emerging studies, suggesting that at least some strains exhibit pathobiontic properties [59]. After an intervention with a Mediterranean diet and a low-fat diet, increased Prevotella and Bacteroides levels have also been observed [87], so Type 8 could be associated with a short-term change of diet. Less abundant genera in Type 8 such as Finegoldia, Staphylococcus, Lactobacillus, Corynebacterium and Fusobacterium have, on rare occassions, been found to cause human infection [17] [64] [91] [5] [1]. So Type 8 could also be associated with infection. Type 3 consists of Bacteroides, Lachnospira and Akkermansia. Type

7 has high proportions of Faecalibacterium and an unclassified genus from family Lachnospiraceae. Type 11 and Type 12 also have high proportions of Faecalibacterium while Type 11 has fairly abundant Lachnospira and Type 12 has fairly abundant Roseburia. Previous studies suggest that Roseburia and Lachnospira are strongly associated with vegetable diets, and also negatively associated with the omnivore diet [104]. Also, Faecalibacterium and Roseburia may be two major proponents of weight loss [16]. This suggests that Type 3, Type 7, Type 11 and Type 12 may all be related to low-fat diets. Type 9 has fairly abundant Phascolarctobacterium which has been found to increase under high fat diet and to be positively correlated to a positive mood and body weight [66] [62].

To investigate the temporal dynamics of the Person 1's gut microbiome, the weight matrix is plotted in Figure 4.3. The weights for each time point are normalized to sum to 1. We see that there is substantial short-term fluctuation for all types. Types 6 and 8 have consistently low weights, with the exception of single-day spikes. One explanation could be Types 6 and 8 are associated with short-term infections, but more information about the OTUs' species is needed for this assumption. It could also be consistent with Type 8 being associated with diet, if the association is short-term, then the spikes might correspond to the individual eating particular foods. Type 9 is dominant with a relatively stable weight for around one week at end of December which would be consistent with the diet change and happy mood. Other types show elevated levels for time periods of weeks or months, indicating changes to the microbial community over time. This seems plausible dynamics for healthy types.

Figure 4.4 shows the abundance of the main genera in Person 2's gut features. 33 out of 3131 OTUs have larger than 3% proportions in at least one feature. These 33 OTUs are from 20 genera. Person 2's Type 3, Type 5 and Type 6 are similar. They are all dominated by Bacteroides with the next most abundant genera, Faecalibacterium and Parabacteroides, appearing in much smaller proportions. Type 4 has high proportions of Bacteroides and Akkermansia. Type 1 is more diverse, with no genus exceeding 20% of the total abundance of this type. The most abundant genera are Prevotella, Escerichia, Finegolida and Porphyromonas. Many species from these genera have previously been associated

with periodontitis [77]. Although the species in this dataset are not the ones previously associated with periodontitis, it seems plausible that this type might be associated with periodontic infections. Type 2 consists of Streptococcus (gordonii species and oralis species), Prevotella (nanceiensis species, melaninogenica species and an unclassified species), Neisseria (subflava species) and a lower abundance of Staphylococcus (epidermidis species), Veillonella (parvula species and an unclassified species), an unclassified genus from family Pasteurellaoeae, Haemophilus (parainfluenzae species) and Fusobacterium (unclassified species). The Prevotella species in Type 2 are different from the species in Type 1. S. gordonii, V. parvula and Fusobacterium are known to be opportunistic pathogens that can cause periodontitis [85] [8] [9]. Also, S. gordonii, V. parvula, H. parainfluenzae and N. subflava are recognized in opportunistic infections such as endocarditis and meningitis [8] [9] [60] [26] [50]. P. melaninogenica is also an important human pathogen in various anaerobic infections, often mixed with other aerobic and anaerobic bacteria [10]. Thus, Type 2 might also be associated with infection.

Figure 4.5 shows the weights of each type in Person 2's gut over time. We see that Types 1 and 2 are usually very low abundance, but have occasional spikes. This is consistent with these types corresponding to infections. The other features show much more stability over time, having similar abundance levels over long periods of time (when normalised with respect to the total of these four types, with Types 1 and 2 excluded). There are some very abrupt changes to the abundance levels of these types at various times during the study. This is consistent with these types representing small variations on a healthy community.

4.5 Conclusion

We have developed an NMF rank selection method based on hypothesis testing using a deconvolved bootstrap to estimate the null distribution. The simulations show that our rank selection method can estimate the rank accurately for both Poisson and Normal NMF data when the true rank is small. When the NMF features are close or the true NMF rank is large, our method has better performance than NNLM. We applied our method to a microbiome data set. With the number of ranks selected, we were able to gain new insights beyond previous analyses of

the same data. The new insights are biologically extremely plausible, and will hopefully lead to new research in the field.

There are a number of directions for future research. Firstly, the sequential nature of the test means that we need to recompute the null distribution for each rank, which is computationally very expensive. The test could be made computationally more efficient for large ranks by using a more efficient search. Instead of increasing the tested rank by 1 each time, we could increase it by more. If the test does not reject the null hypothesis, it would then be necessary to perform the tests for lower ranks. This would increase computation when the rank is low, but could decrease it significantly for higher rank. More heuristics could be added to decide which hypothesis tests to perform for maximum efficiency.

Another issue is that a single failure to reject the null hypothesis can cause the method to select the current rank. It might be possible to develop a more robust method that combines the output of additional hypothesis tests to estimate the rank more reliably.

Another direction for future research is estimation of variance for the parametric bootstrap in the Normal case. The variance is estimated from the residuals of the model. However, because the model is fitted to the data, these residuals will be smaller when the rank is larger. For linear regression, there is a correction to get an unbiased estimate for the variance. However, for NMF, the nonnegativity constraint means that this correction is not applicable, so another method is needed to obtain a more stable estimate for the variance. This will make the critical value the test based on from the parametric bootstrap more accurate.

The application of deconvolution to deal with optimization errors in bootstrap samples has potential applications in any field where full optimization is computationally expensive. This is common for discrete optimization problems such as variable selection, clustering, phylogenetics and many other areas. Applying the deconvolved bootstrap in these areas could prove a very fruitful topic for future research.

Chapter 5

Conclusion

There are two main research contributions in this thesis. First a general measurement error deconvolution method based on the penalized likelihood and the associated asymptotic theory is developed. An R package `pmledecon` is available through [13]. Building on this new measurement error deconvolution method based on the penalized likelihood, we further developed a general method for the rank selection for NMF.

The comparison results of supervised NMF with existing methods in Chapter 2 Section 2.2 and Section 2.3 show that supervised NMF can effectively reduce the dimensionality of the data to a non-negative and most often sparse data matrix, which contains sufficient discriminative information for classification purposes and NMF is able to identify biologically meaningful types representing the subcommunities. NMF is a very useful and effective method for analyzing the microbiome data.

Chapter 3 develops a deconvolved density estimation method (pmledecon) which can be used to estimate the true underlying density for variable contaminated with additive error. The pmledecon method is based on maximizing penalized log-likelihood estimation with a smoothness penalty on the estimated density. We prove the consistency of our method and show the strong point of our method at estimating the deconvolved density by comparing with other deconvolution methods in both simulation and real data application. Deconvolution is an important problem which arises in a large number of real-world experiments, so this method should have broad applicability, in addition to being a key component of the method developed in Chapter 4.

Chapter 4 provides an NMF rank selection method (Decon-boot-test) based on hypothesis testing using a deconvolved bootstrap to estimate the null distribution. The simulations show that Decoon-boot-test can estimate the true NMF ranks

accurately for both Poisson and Normal data. Application of NMF with Decon-boot-test estimated rank on the same microbiome data used in Chapter 2 brings more biologically plausible insights than previous results in Chapter 2.

NMF is a powerful tool for identifying the key features of microbial communities. With the NMF rank estimated appropriately by Decon-boot-test, the identified features are very interpretable and can lead to important biological insights into the structure of the communities. In addition, the low-dimensional representation of NMF applied on the extremely complex microbial data allows a number of analyses to be performed more easily—for example, searching for temporal patterns in the microbiome.

Bibliography

[1] Kevin Afra, Kevin Laupland, Jenine Leal, Tracie Lloyd, and Daniel Gregson. Incidence, risk factors, and outcomes of fusobacterium species bacteremia. *BMC infectious diseases*, 13(1):1–6, 2013.

[2] Kevin R. Arrigo. Marine microorganism and global nutrient cycles. *Nature*, 437:349–355, 2005.

[3] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[4] Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pages 1165–1188, 2001.

[5] Kathryn Bernard. The genus corynebacterium and other medically relevant coryneform-like bacteria. *Journal of clinical microbiology*, 50(10):3152–3158, 2012.

[6] David Berry and Stefanie Widder. Deciphering microbial interactions and detecting keystone species with co-occurrence networks. *Frontiers in microbiology*, 5:219, 2014.

[7] M.W. Berry and M. Browne. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52:155–173, 2007.

[8] GPA Bongaerts, BW Schreurs, F Verduyn Lunel, JAM Lemmens, M Pruszczynski, and MAW Merkx. Was isolation of veillonella from spinal osteomyelitis possible due to poor tissue perfusion? *Medical hypotheses*, 63(4):659–661, 2004.

[9] Marissa Broadley and Steven J Schweon. Get the facts about fusobacterium. *Nursing2020*, 47(5):64–65, 2017.

[10] Itzhak Brook. *Anaerobic infections: diagnosis and management*. CRC Press, 2007.

[11] Jean-Philippe Brunet, Pablo Tamayo, Todd R Golub, and Jill P Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the national academy of sciences*, 101(12):4164–4169, 2004.

[12] Yun Cai, Hong Gu, and Toby Kenney. Dbnmfrank: Rank selection for non-negative matrix factorization. https://CRAN.R-project.org/package=DBNMFrank, 2021.

[13] Yun Cai, Hong Gu, and Toby Kenney. pmledecon: Deconvolution density estimation with penalised mle. https://CRAN.R-project.org/package=pmledecon, 2021.

[14] J Gregory Caporaso, Christian L Lauber, Elizabeth K Costello, Donna Berg-Lyons, Antonio Gonzalez, Jesse Stombaugh, Dan Knights, Pawel Gajer, Jacques Ravel, Noah Fierer, et al. Moving pictures of the human microbiome. *Genome biology*, 12(5):1–8, 2011.

[15] R.J. Carroll, D. Ruppert, L.A. Stefanski, and C.M. Crainiceanu. *Measurement Error in Nonlinear Models: A Modern Perspective, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press, 2006.

[16] Chandra Kanti Chakraborti. New-found link between microbiota and obesity. *World journal of gastrointestinal pathophysiology*, 6(4):110, 2015.

[17] Fernando Cobo. Infections caused by anaerobic microorganisms. 2021.

[18] Fabienne Comte and Claire Lacour. Data-driven density estimation in the presence of additive noise with unknown distribution. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):601–627, 2011.

[19] Fabienne Comte, Yves Rozenholc, and Marie-Luce Taupin. Penalized contrast estimator for adaptive density deconvolution. *Canadian Journal of Statistics*, 34(3):431–452, 2006.

[20] Fabienne Comte, Adeline Samson, and Julien J Stirnemann. Deconvolution estimation of onset of pregnancy with replicate observations. *Scandinavian Journal of Statistics*, 41(2):325–345, 2014.

[21] Massimo Costalonga and Mark C Herzberg. The oral microbiome and the immunobiology of periodontal disease and caries. *Immunology letters*, 162(2):22–38, 2014.

[22] RP Darveau, G Hajishengallis, and MA Curtis. Porphyromonas gingivalis as a potential community activist for disease. *Journal of dental research*, page 0022034512453589, 2012.

[23] Aurore Delaigle and Irène Gijbels. Bootstrap bandwidth selection in kernel density estimation from a contaminated sample. *Annals of the Institute of Statistical Mathematics*, 56(1):19–47, 2004.

[24] Aurore Delaigle, Peter Hall, Alexander Meister, et al. On deconvolution with repeated measurements. *The Annals of Statistics*, 36(2):665–685, 2008.

[25] Karthik Devarajan. Nonnegative matrix factorization: an analytical and interpretive tool in computational biology. *PLoS Comput Biol*, 4(7):e1000029, 2008.

[26] Gustavo Deza, Gemma Martin-Ezquerra, Julià Gómez, Judit Villar-García, August Supervia, and Ramon M Pujol. Isolation of haemophilus influenzae and haemophilus parainfluenzae in urethral exudates from men with acute urethritis: a descriptive study of 52 cases. *Sexually transmitted infections*, 92(1):29–31, 2016.

[27] Deacqunita L Diggs et al. Polycyclic aromatic hydrocarbons and digestive tract cancers: a perspective. *Journal of Environmental Science and Health, Part C*, 29(4):324–357, 2011.

[28] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135, 2006.

[29] Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, pages 642–669, 1956.

[30] M Émile Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 27(1):247–271, 1909.

[31] Attila Frigyesi and Mattias Höglund. Non-negative matrix factorization for the analysis of complex gene expression data: identification of clinically relevant tumor subtypes. *Cancer informatics*, 6:CIN–S606, 2008.

[32] Kei E Fujimura, Nicole A Slusher, Michael D Cabana, and Susan V Lynch. Role of the gut microbiota in defining human health. *Expert review of anti-infective therapy*, 8(4):435–454, 2010.

[33] Renaud Gaujoux and Cathal Seoighe. *The package NMF: manual pages*, 2015. R package version 0.20.6.

[34] S. Giguere, J. F. Prescott, J. D. Baggot, R. D. Walker, and P. M. Dowling. *Antimicrobial Therapy in Veterinary Medicine (4th ed.)*. Wiley-Blackwell, USA, 2006.

[35] J.A. Gilbert, J.A. Steele, and J.G. Caporaso. Defining seasonal marine microbial community dynamics. *The ISME journal*, 6:298–308, 2012.

[36] Nicolas Gillis. The why and how of nonnegative matrix factorization. *Regularization, optimization, kernels, and support vector machines*, 12(257):257–291, 2014.

[37] E. Gonzalez and Y. Zhang. Accelerating the lee-seung algorithm for non-negative matrix factorization. *Dept. Comput. & Appl. Math., Rice Univ., Houston, TX, Tech. Rep. TR-05-02*, 2005.

[38] Peter J Green. On use of the em algorithm for penalized likelihood estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 52(3):443–452, 1990.

[39] Peter J Green and Bernard W Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.

[40] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning: Data mining, inference, and prediciton. *The Mathematical Intelligencer*, 27:83–85, 2005.

[41] Matthew D Hoffman, David M Blei, and Perry R Cook. Bayesian nonparametric matrix factorization for recorded music. In *ICML*, 2010.

[42] C Michael Hogan. Bacteria. *Encyclopedia of Earth. Washington DC: National Council for Science and the Environment*, 2010.

[43] C. Holling. Some characteristics of simple types of predation and parasitism. *The Canadian Entomologist*, 91:385–398, 1959.

[44] I Holmes, K Harris, and C Quince. Dirichlet multinomial mixtures: Generative models for microbial metagenomics. *PLos One*, 7:e30126, 2012.

[45] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

[46] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(9), 2004.

[47] David A Hughes. Effects of carotenoids on human immune function. *Proceedings of the Nutrition Society*, 58(03):713–718, 1999.

[48] Xingpeng Jiang, Morgan GI Langille, Russell Y Neches, Marie Elliot, Simon A Levin, Jonathan A Eisen, Joshua S Weitz, and Jonathan Dushoff. Functional biogeography of ocean microbes revealed through non-negative matrix factorization. *PloS one*, 7(9):e43866, 2012.

[49] Fabienne Comte. Contribution from Claire Lacour Julien Stirnemann, Adeline Samson. *Deconvolution density estimation with adaptive methods for a variable prone to measurement error*, 2012. R package version 1.0.

[50] Jeffrey B Kaplan and Daniel H Fine. Biofilm dispersal of neisseria subflava and other phylogenetically diverse oral bacteria. *Applied and Environmental Microbiology*, 68(10):4943–4950, 2002.

[51] Gérard Kerkyacharian, Thanh Mai Pham Ngoc, Dominique Picard, et al. Localized spherical deconvolution. *The Annals of Statistics*, 39(2):1042–1068, 2011.

[52] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23:1495–1502, 2007.

[53] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[54] D. Knights, E.K. Costello, and R. Knight. Supervised classification of human microbiota. *FEMS microbiology reviews*, 35:343–359, 2011.

[55] D Knights, J Kuczynski, and E S Charlson. Bayesian community-wide culture-independent microbial source tracking. *Nat Methods*, 8:761–3, 2011.

[56] Mikael Kuusela and Victor M Panaretos. Statistical unfolding of elementary particle spectra: Empirical bayes estimation and bias-corrected uncertainty quantification. *The Annals of Applied Statistics*, 9(3):1671–1705, 2015.

[57] B Kwiatkowska and M Maślińska. Macrolide therapy in chronic inflammatory diseases. *Mediators of inflammation*, 2012, 2012.

[58] Nan Laird. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73(364):805–811, 1978.

[59] Jeppe Madura Larsen. The immune response to prevotella bacteria in chronic inflammatory disease. *Immunology*, 151(4):363–374, 2017.

[60] Yevgeniy Latyshev, Aswin Mathew, Jeffrey M Jacobson, and Eron Sturm. Purulent pericarditis caused by haemophilus parainfluenzae. *Texas Heart Institute Journal*, 40(5):608, 2013.

[61] Hans Laurberg, Mads Græsbøll Christensen, Mark D Plumbley, Lars Kai Hansen, and Søren Holdt Jensen. Theorems on positive data: On the uniqueness of nmf. *Computational intelligence and neuroscience*, 2008, 2008.

[62] Virginie Lecomte, Nadeem O Kaakoush, Christopher A Maloney, Mukesh Raipuria, Karina D Huinao, Hazel M Mitchell, and Margaret J Morris. Changes in gut microbiota in rats fed a high fat diet correlate with obesity-associated metabolic parameters. *PloS one*, 10(5):e0126931, 2015.

[63] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[64] Ezra Lee and Fatima Anjum. Staphylococcus epidermidis. In *StatPearls [Internet]*. StatPearls Publishing, 2021.

[65] M A Leibold, M Holyoak, and N Mouquet. Bacterial community assembly based on funcional genes rather than species. *PLos Natl Acad Sci*, 108:14288–93, 2011.

[66] L Li, Qiang Su, B Xie, L Duan, W Zhao, D Hu, R Wu, and Hong Liu. Gut microbes in correlation with mood: case study in a closed experimental human life support system. *Neurogastroenterology & Motility*, 28(8):1233–1240, 2016.

[67] C J Lin. On the convergence of multiplicative update algorithms for non-negative matrix factorization. *Neural Networks, IEEE Transactions on*, 18:1589 – 1596, 2007.

[68] Chih-Jen Lin. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 18(6):1589–1596, 2007.

[69] Xihui Lin and Paul C Boutros. Optimization and expansion of non-negative matrix factorization. *BMC bioinformatics*, 21(1):1–10, 2020.

[70] Lei Liu, Michael Levine, and Yu Zhu. A functional em algorithm for mixing density estimation via nonparametric penalized likelihood maximization. *Journal of Computational and Graphical Statistics*, 18(2):481–504, 2009.

[71] Ming Chung Liu and Robert L Taylor. A consistent nonparametric density estimator for the deconvolution problem. *Canadian Journal of Statistics*, 17(4):427–438, 1989.

[72] Catherine Lozupone, Manuel E Lladser, Dan Knights, Jesse Stombaugh, and Rob Knight. Unifrac: an effective distance metric for microbial community comparison. *The ISME journal*, 5(2):169, 2011.

[73] Oscar-Hernan Madrid-Padilla, Nicholas G Polson, and James Scott. A deconvolution path for mixtures. *Electronic Journal of Statistics*, 12(1):1717–1751, 2018.

[74] P. Massart. The Tight Constant in the Dvoretzky-Kiefer-Wolfowitz Inequality. *The Annals of Probability*, 18(3):1269 – 1283, 1990.

[75] Paul J McMurdie and Susan Holmes. Waste not, want not: why rarefying microbiome data is inadmissible. *PLoS Comput Biol*, 10(4):e1003531, 2014.

[76] Paul J McMurdie and Susan Holmes. Waste not, want not: why rarefying microbiome data is inadmissible. *PLoS computational biology*, 10(4):e1003531, 2014.

[77] Feng Mei, Mengru Xie, Xiaofei Huang, Yanlin Long, Xiaofeng Lu, Xiaoli Wang, and Lili Chen. Porphyromonas gingivalis and its systemic impact: Current status. *Pathogens*, 9(11):944, 2020.

[78] A Mencarelli, E Distrutti, B Renga, and et al. Development of non-antibiotic macrolide that corrects inflammation-driven immune dysfunction in models of inflammatory bowel diseases and arthritis. *European journal of pharmacology*, 665(1):29–39, 2011.

[79] John Mendelsohn and John Rice. Deconvolution of microfluorometric histograms with b splines. *Journal of the American Statistical Association*, 77(380):748–753, 1982.

[80] Lidan Miao and Hairong Qi. Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 45(3):765–777, 2007.

[81] B. D. Muegge et al. Diet drives convergence in gut microbiome functions across mammalian phylogeny and within humans. *Science*, 332:970–974, 2011.

[82] Laura Muzzarelli, Susanne Weis, Simon B Eickhoff, and Kaustubh R Patil. Rank selection in non-negative matrix factorization: systematic comparison and a new mad metric. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[83] R Nelder, John A.and Mead. A simplex method for function minimization. *The Computer Journal*, 7(308):13, 1965.

[84] Paul C Odiachi and Dennis C Prieve. Removing the effects of additive noise from tirm measurements. *Journal of colloid and interface science*, 270(1):113–122, 2004.

[85] Ok-Jin Park, Yeongkag Kwon, Chaeyeon Park, Yoon Ju So, Tae Hwan Park, Sungho Jeong, Jintaek Im, Cheol-Heui Yun, and Seung Hyun Han. Streptococcus gordonii: Pathogenesis and host response to its cell wall components. *Microorganisms*, 8(12):1852, 2020.

[86] V. V. Phelan, W. T. Liu, K. Pogliano, and P. Dorrestein. Microbial metabolic exchange—the chemotype-to-phenotype link. *Nat Chem Biol*, 8:26–35, 2012.

[87] Gabriela Precup and Dan-Cristian Vodnar. Gut prevotella as a possible biomarker of diet and its eubiotic versus dysbiotic roles: a comprehensive literature review. *British Journal of Nutrition*, 122(2):131–140, 2019.

[88] J. Qin et al. A human gut microbial gene catalogue established by metagenomic sequencing. *nature*, 464:59–65, 2010.

[89] Junjie Qin, Ruiqiang Li, Jeroen Raes, Manimozhiyan Arumugam, Kristoffer Solvsten Burgdorf, Chaysavanh Manichanh, Trine Nielsen, Nicolas Pons, Florence Levenez, Takuji Yamada, et al. A human gut microbial gene catalogue established by metagenomic sequencing. *nature*, 464(7285):59–65, 2010.

[90] Alban Ramette. Multivariate analyses in microbial ecology. *FEMS Microbiology Ecology*, 62(2):142–160, 2007.

[91] Franca Rossi, Carmela Amadoro, and Giampaolo Colavita. Members of the lactobacillus genus complex (lgc) as opportunistic pathogens: a review. *Microorganisms*, 7(5):126, 2019.

[92] Emidio Scarpellini, Gianluca Ianiro, Fabia Attili, Chiara Bassanelli, Adriano De Santis, and Antonio Gasbarrini. The human gut microbiota and virome: Potential therapeutic implications. *Digestive and Liver Disease*, 47(12):1007–1012, 2015.

[93] Inna Sekirov, Shannon L Russell, L Caetano M Antunes, and B Brett Finlay. Gut microbiota in health and disease. *Physiological reviews*, 90(3):859–904, 2010.

[94] H Sebastian Seung and Daniel D Lee. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems*, 13:556–562, 2001.

[95] M. Shafiei, K.A. Dunn, E. Boon, S.M. MacDonald, D.A. Walsh, H. Gu, and J.P. Bielawski. Biomico: a supervised bayesian model for inference of microbial community structure. *Microbiome*, 3(1):1, 2015.

[96] M. Shafiei, K.A. Dunn, H. Chipman, H. Gu, and J.P. Bielawski. Biomenet: A bayesian model for inference of metabolic divergence among microbial communities. *PLoS Computational Biology*, 10:e1003918, 2014.

[97] F. Shahnaz, M. Berry, and R. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42:373–386, 2006.

[98] M Shinkai, M O Henke, and B K Rubin. Macrolide antibiotics as immunomodulatory medications: proposed mechanisms of action. *Pharmacology & therapeutics*, 117(3):393–405, 2008.

[99] BW Silverman, MC Jones, JD Wilson, and DW Nychka. A smoothed em approach to indirect estimation problems, with particular reference to stereology and emission tomography. *Journal of the Royal Statistical Society: Series B (Methodological)*, 52(2):271–303, 1990.

[100] Aristeidis Sotiras, Jon B Toledo, Raquel E Gur, Ruben C Gur, Theodore D Satterthwaite, and Christos Davatzikos. Patterns of coordinated cortical remodeling during adolescence and their associations with functional specialization and evolutionary expansion. *Proceedings of the National Academy of Sciences*, 114(13):3527–3532, 2017.

[101] Steven Squires, Adam Prügel-Bennett, and Mahesan Niranjan. Rank selection in nonnegative matrix factorization using minimum description length. *Neural computation*, 29(8):2164–2176, 2017.

[102] Edward Susko and Robert Nadon. Estimation of a residual distribution with small numbers of repeated measurements. *Canadian Journal of Statistics*, 30(3):383–400, 2002.

[103] Peter J Turnbaugh, Ruth E Ley, Michael A Mahowald, Vincent Magrini, Elaine R Mardis, and Jeffrey I Gordon. An obesity-associated gut microbiome with increased capacity for energy harvest. *nature*, 444(7122):1027–1031, 2006.

[104] Mirco Vacca, Giuseppe Celano, Francesco Maria Calabrese, Piero Portincasa, Marco Gobbetti, and Maria De Angelis. The controversial role of human gut lachnospiraceae. *Microorganisms*, 8(4):573, 2020.

[105] Xiao-Feng Wang and Bin Wang. Deconvolution estimation in measurement error models: the r package decon. *Journal of statistical software*, 39(10), 2011.

[106] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, pages 80–83, 1945.

[107] Simon N Wood. Inferring uk covid-19 fatal infection trajectories from daily mortality data: Were infections already in decline before the uk lockdowns? *Biometrics*, 2021.

[108] Qi Zhou, Yanfeng Zhang, Xiaoxia Wang, Ruiyue Yang, Xiaoquan Zhu, Ying Zhang, Chen Chen, Huiping Yuan, Ze Yang, and Liang Sun. Gut bacteria akkermansia is associated with reduced risk of obesity: evidence from the american gut project. *Nutrition & metabolism*, 17(1):1–9, 2020.