Mat-2.4108 - Independent Research Project in Applied Mathematics

Unsupervised learning from HITChip microarray data

November 8, 2011

Max Sandholm

Aalto University School of Science
Systems Analysis Laboratory

Contents

1 Introduction

Rather recently different high-throughput data gathering methods have opened up new avenues for the research of biology in general. For example in terms of the human intestinal tract, technologies such as HITChip (Rajilic-Stojanovic et al., 2009) and PhyloChip (DeAngelis et al., 2011) have allowed researchers to collect even thousands of samples that describe the microbial profiles in the gut. The benefits in terms of cost and speed are tremendous compared to having to take and analyze each sample individually in a laboratory setting.

However, the difficulty is how to find patterns in these huge amounts of data. It is not possible to just have a look at the data and make inferences straight away. The dimensionality of the data has to be reduced significantly for human inference to be feasible. An example of this approach is the paper by Arumugam et al. (2011), where three robust clusters, called enterotypes, were found across different sets of samples.

Thus the aim of this study is to examine and implement unsupervised learning methods that can be used for inference from large data sets describing the amount of different bacteria in human fecal samples. The goal of unsupervised learning in general is to find regularities in the data without the preset user input that characterizes supervised learning (Alpaydin, 2010).

Unsupervised learning methods most relevant to this special study include clustering, component analysis and generative modeling. In clustering, we try to find points that best represent the groups present in our data set. Component analysis on the other hand tries to find out components that when combined with different multiplying coefficients can be seen to create each of the samples. For instance there could be a component that is common for everyone and then components that vary based on the physical condition, age, or living environment of the person in question. Finally, generative probabilistic models can be used not only for both clustering and finding components, but to create an estimate of the probability distribution of the data and relevant parameters. In particular in this case the generative models produce Dirichlet distributed data, which influences the computational difficulties that accompany the model. The implementation of these methods and the computational procedure involving the generative model will be discussed.

The data set used in this study consists of 1963 samples collected with the aforementioned HITChip technology. Each sample consists of 130 different variables that represent different groups of bacteria. The information used

is L2 data, meaning that certain species related to each other are combined to produce the L2 data from species level data.

This special study is structured as follows. First the different methods for unsupervised learning will be discussed. These include the choice of clustering algorithms, related distance metrics, different component analysis methods, and generative models. A high emphasis will also be put on validation techniques, because it is vital for the reliability and interpretability of the results that one chooses the right number of clusters or components to examine. After the chosen methodologies have been discussed, the results of different methods will be presented. Finally in the discussion section the goal is to examine the reliability of these results and discuss further possibilities for research in these areas.

2 Learning methods

2.1 Clustering

Clustering is an unsupervised learning method widely used to learn groups in data. An example in the human microbiota research is the paper from Arumugam et al. (2011) where three robust clusters were found. The most common way to do clustering is the traditional k-means algorithm. Lloyd's implementation follows the guideline described in Algorithm 1.

Alg. 1 Lloyd's k-means algorithm

begin
Initialize cluster centroids
until Cluster assignments remain the same
1) Assign each point to closest cluster centroid
2) Update cluster centroids: $\vec{c}_i = \frac{1}{|C_i|} \sum_{a_k=i} \vec{x}_k$
end

In Algorithm 1 the points are assigned to the closest cluster centroid and then the centroids are updated to the mean of the points belonging to the same clusters. It is apparent that the choice of distance measure holds meaning in both steps: the measure influences the choice of closest cluster centroid in step one and also the updating formula in step two. Here Algorithm 1 is based on Euclidean distances, which means that the second step minimizes the within cluster distances.

Readily available k-means implementations often have a possibility for using for example squared Euclidean distances or the Manhattan distance. Whether it is wise to use them for clustering depends on the nature of the data set. The original HITChip paper (Rajilic-Stojanovic et al., 2009) presented that the HITChip probe readings don't represent the absolute amounts of different species of bacteria, although the correlation is relatively high. Instead, the readings describe better the relative abundances of different species.

This leads us to consider data transformations and different choices of a distance metric to remove the possibility that similar abundance profiles may seem different due to high differences in absolute values. In the paper by Arumugam et al. (2011) the authors scaled the data to sum up to one and used the square root of Jensen-Shannon divergence (Endres and Schindelin, 2003) as a distance measure. Jensen-Shannon divergence is a measure specifically used to define distances between different probability distributions, and its square root is a metric (Endres and Schindelin, 2003). It is defined as

$$\begin{aligned} JSD(P||Q) &= \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q\ ||M) \\ &= \frac{1}{2}\sum_i P(i)\log\frac{P(i)}{M(i)} + \frac{1}{2}\sum_j Q(j)\log\frac{Q(j)}{M(j)}, \end{aligned}$$

where D_{KL} is the Kullback-Leibler divergence and $M = \frac{P+Q}{2}$.

In this case transforming the data vectors to describe probability distributions removes the problem with differences in absolute values. Using the square root of Jensen-Shannon divergence as a distance measure is well founded, because it is a metric developed specifically for probability distributions, and this choice also guarantees comparability to the results of Arumugam et al. (2011).

As mentioned earlier in this section, the choice of a distance measure influences also the behavior of the clustering algorithm. The problem with using k-means with a different distance measure is to derive the centroid updating formula (step two in algorithm 1).

This leads us into using the partitioning around medoids algorithm (Kaufman and Rousseeuw, 1990) (Algorithm 2). Partitioning around medoids is similar to k-means but uses always an actual data point as a cluster centroid instead of an arbitrary point somewhere in the middle. This allows the algorithm to operate using purely the pairwise distances between data points,

Alg. 2 Partitioning around medoids algorithm

begin
Initialize cluster medoids to some data points
until Cluster assignments remain the same
1)Assign each point to closest cluster medoid
2)Update each medoid to the data point which minimizes within cluster dissimilarities
end

and enables the use of different distance metrics. The same algorithm was also used by Arumugam et al. (2011).

2.2 Component analysis

In clustering, we allow each data vector to be assigned only to a single cluster. Next we relax that requisite by assigning each data point a $k \times 1$ vector (k is the number of components) to give scores for different components. There are many different methods for finding components from data such as principal component analysis (PCA), independent component analysis (ICA) (Comon, 1994), factor analysis, non-negative matrix factorization (NMF) (Lee and Seung, 1999) and the recently develop semi-nonnegative matrix factorization (semi-NMF) (Ding et al., 2010). An important part in choosing the method is considering what the goal is when doing the analysis. Here the aim is to find components that can then be interpreted and analyzed by the biologists. The original hypothesis about how the components work is that when the amount of certain bacteria increase, some may decrease at the same time.

All of the component analysis methods can produce a matrix factorization of the form $X \approx FG^T$, where X is a $m \times n$ data matrix, F is a $m \times k$ matrix of components and G is a $n \times k$ matrix of component scores for each data sample. One of the differences between all of these methods are the assumptions that they make about the matrices F, G and X. Although the underlying principles are in some ways different, PCA, ICA and factor analysis all produce components that may have positive or negative values and also the component scores may be positive or negative. This results in a problem in the interpretation of the components. Let us assume that there are two components, A and B, and two people, X and Y. X has scores 0.7 for A and -0.5 for B, and Y has -0.4 for A and 0.8 for B. Now if A

has both positive and negative values, the interpretation seems reasonable when X is concerned: some values go up and some down when the score for A is increased. But when the score is negative with person Y, the signs in the component itself are basically reversed. It is hard to say that it is even the same component explaining the data anymore as the effect is the exact opposite of the component that we found with a positive score. It would be much easier if components could have only positive scores and the interpreter could look at them as different parts that are added on top of each other.

These difficulties in interpretation make NMF and semi-NMF more attractive possibilities. However, NMF demands that both the components and their scores are non-negative. This requirement limits the kind of components that can be found. Semi-NMF relaxes this condition and allows negative values in the data matrix and the components but keeps the component scores positive. Thus semi-NMF seems the most appropriate method to produce results that can be readily interpreted. Ding et al. (2010) present an iterative algorithm that fulfills the Karush-Kuhn-Tucker conditions for the solution of a constrained optimization problem.

As a preparation for the semi-NMF algorithm the data is scaled so that all column vectors sum up to one. This eliminates the possibility that the components would try to explain large differences in absolute values. Ding et al. (2010) argue that the algorithm, initialized using k-means, will converge into a solution where the components resemble the cluster centroids. Here that is not exactly the case as there will be negative values in the components whereas the data describes probability distributions, and so includes values $w_i \in (0, 1), \sum_i^M w_i = 1$.

However, it is reasonable to assume that the components will resemble to a certain extent the cluster centroids. If there is a cluster centroid that has a relatively high value for variable A and a low value for variable B, there will be a component that also has a low or a negative value for B and a higher value for A. The initialization is done using the earlier mentioned partitioning around medoids clustering algorithm, but otherwise with the same principle as in the original semi-NMF paper where the authors suggested the use of k-means. To assess the convergence of the algorithm it is possible to check how much the approximated FG^T matrix, F or G themselves change from one iteration to another. Here it is required that all the values in G change less than 0.0001, otherwise the iteration will be continued.

We should note that none of these methods really produce the kind of data we started with. Even basic NMF would produce an approximate data

matrix that doesn't sum up to one even though the data would stay positive. Nevertheless, the eventual results can be used to describe relationships between different bacterial groups.

2.3 Validation methods

A major question related to clustering and component analysis is how to choose the correct number of clusters or components. Using known clustering algorithms or component analysis methods we can always find any number of clusters or components. The goal is to find an appropriate number of them to describe the data well enough, and at the same time avoid significant overfitting to the training data.

In this study three different approaches are considered as possible measures for cluster validation:

- Calinski-Harabasz index (Calinski and Harabasz, 1974),
- Silhouette validation technique (Rousseeuw, 1987),
- cross-validation measuring the decrease of within cluster differences.

Cross-validation is also used for validating the component analysis results. Each of these methods will be discussed next in the order above.

These three methods were chosen for a few reasons. CH-index was the measure used by Arumugam et al. (2011) which makes it important to use it for comparison. CH-index has also been found to produce comparably good approximations of the actual number of clusters in the data set (Milligan and Cooper, 1985), although there are also studies that notice occasional poor performance (Maulik and Bandyopadhyay, 2002). Silhouette was also briefly discussed by Arumugam et al. (2011), and average Silhouette has been found to be a good measure in determining the number of clusters (van der Laan et al., 2003). Finally, using cross-validation to examine plainly the error of using a specific number of clusters to describe the data provides another way of looking at the data. By using different measures one can examine the robustness of the results. If the indices agree with each other the results can be seen to be more reliable than before, and differing results warrant more caution when actually interpreting the results.

2.3.1 CH-index

Calinski-Harabasz (CH) index (Calinski and Harabasz, 1974) is defined in Euclidean distances by the formula

$$CH(k) = \frac{B(k)}{W(k)} \frac{N-k}{k-1},$$

where k is the number of clusters, N is the number of samples, $B(k)$ is the total between cluster sum of squares, $W(k)$ is the total within cluster sum of squares. The optimum of the index comes at its maximum point. Intuitively the index rewards for large separation between clusters and small variance inside a single cluster while punishing for having too many clusters. An extension to accommodate different distance measures is given by Hennig and Liao (2010). The authors defined $B(k)$ and $W(k)$ as

$$W(k) = \sum_{h=1}^{k} \frac{1}{|C_h|} \sum_{w_i, w_j \in C_h} d(w_i, w_j)^2,$$

$$B(k) = \frac{1}{n} \sum_{i,j=1}^{n} d(w_i, w_j)^2 - W(k).$$

In the definition of $W(k)$ we sum for each point i its squared distances to all points j that belong to the same cluster as i, do this for all points, and average the results within the cluster. Finally we sum the results of different clusters to obtain the within cluster measure. $B(k)$ is defined similarly to traditional between cluster sum of squares as the difference of a term that describes the variation in the whole data set and the within cluster measure.

2.3.2 Silhouette validation

Silhouette width is defined as

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}},$$

where $b(i)$ is the average dissimilarity from a certain point to the nearest neighboring cluster and $a(i)$ is the average dissimilarity to the points in the point's own cluster. One may notice from the equation for $s(i)$ that it is confined to the interval $[-1, 1]$, where value one means that all the points in a points own cluster are at the exact same position as the point itself. Values

below zero signify a misclassification and the closer $s(i)$ is to one the better. The way to use different Silhouettes as a validation technique is to use the average Silhouette width.

Both CH-index and Silhouette width were calculated using the R-package fpc and its function cluster.stats (Hennig, 2010). We get the parameters from calculating the distance matrix with the square root of Jensen-Shannon divergence and running the partitioning around medoids algorithm.

2.3.3 Cross-validation

Compared to CH-index and Silhouette validation technique that both take the between cluster differences into consideration, 10-fold cross-validation can be used to measure purely how well the distribution of the data is described. Here we measure the average squared distances to the closest cluster medoid. The clustering is found with each training set, and then tested with a corresponding validation set. The measure used to compare different numbers of clusters is then

$$D(k) = \frac{1}{F} \sum_{m=1}^{F} \frac{1}{|V_m|} \sum_{i=1}^{|V_m|} d(x_i, c_i)^2,$$

where F is the number of folds (10), x_i is a validation set vector, c_i is the closest cluster medoid for i and $|V_m|$ is the size of each validation set.

The cross-validation procedure operates as follows. First, we divide the data randomly to ten different equally sized parts. Then we run ten iterations, where we do the pam-clustering with nine of the parts, and calculate the average squared distance in terms of Jensen-Shannon divergence to the nearest cluster medoid for the left out validation set. It seems clear that when plotting the average squared distances for validation sets, the measure will become smaller when the number of clusters increases. However, if we can spot an "elbow" in the plot after which the improvement slows down, we can conclude that the amount of clusters representing the elbow explains the data well enough. The small increase in accuracy indicates that overfitting is highly probable if the number of clusters is still increased.

Cross-validation is also used in this study for validating the semi-NMF results. Validation techniques are a research field that is yet to achieve a final consensus on which methods would be the best as far as component analysis is concerned. As semi-NMF is a newly developed method there is no research that would be directly aimed at deciding the correct number of

components to be taken out from a data set with semi-NMF (for a review and testing of techniques for NMF, see Maisog (2009)). This is why 10-fold cross-validation is used again in this context. We first find the component matrix F with the training set, and optimize the component scores for the validation set using the update rule found in the semi-NMF algorithm. The measure indicating how good of an approximation does a certain amount of components provide is the Frobenius norm of the matrix $X - FG^T$, where X is the actual validation set and FG^T is the estimated validation set. The Frobenius norm of a matrix A is defined as

$$\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2}.$$

The Frobenius norm is calculated for each different validation set, and these results are averaged over all the sets. If there is an elbow in the plot, where the improvement from one amount of components to the next slows down, the corresponding number of components is the optimum.

2.4 Generative modeling

In addition to the purely iterative methods discussed above, building generative probabilistic models is also a possibility. In generative models we assume that the data has been generated from a certain probability distribution and generally infer the needed parameters conditioning on the data itself. A generative model forms an estimate of the probability distribution of the data and allows us to engage in more ways of examining the data. For example, a probabilistic model would make it possible to consider how well a data point fits into a certain cluster in terms of posterior probabilities instead of just hard clustering.

Another advantage of using generative models is the possibility of modeling a process that produces the kind of data we started with. As was seen when discussing component analysis methods, this property cannot be taken for granted and some other approaches lead to the model constructing data that may be impossible by the nature of the original data itself. For example, the abundances of different bacteria certainly cannot be negative but most component analysis methods allow and may produce negative values with their approximations. The possibility of having a model that in principle models the right thing makes pursuing generative models even more worthwhile.

2.4.1 Generative model for clustering

When the data is scaled to sum up to one, we can think of data samples as separate draws from a Dirichlet distribution with some parameters $\Theta_{\cdot k}$. The density function of the Dirichlet distribution is given by

$$p(x_{\cdot j}|\Theta_{\cdot k}) = \frac{\Gamma(\sum_{i=1}^{M}\Theta_{ik})}{\prod_{i=1}^{M}\Gamma(\Theta_{ik})}\prod_{i=1}^{M}x_{ij}^{\Theta_{ik}-1}; \forall(i,k)\Theta_{ik} > 0,$$

where $x_{\cdot j}$ is a sample vector describing a data point and $\Theta_{\cdot k}$ is a vector of Dirichlet parameters coming from cluster k. When we assume the data to come from the Dirichlet distribution, we can build a generative model in an attempt to infer the parameters: the clusters Θ and cluster assignments Z. Generative models are often presented in graphical form to display the relationships that different variables have with each other. The model is presented here in Figure 1.

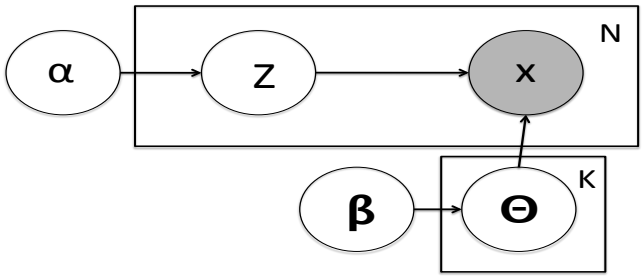

Figure 1: *Plate presentation of the generative model for clustering Dirichlet distributed data.*

The joint distribution of the model presented in Figure 1 is given by

$$p(X, \Theta, Z|\alpha, \beta) = \prod_{k}^{K} \Big(p(\Theta_{\cdot k}|\beta) \Big) \prod_{j}^{N} \Big(p(z_j|\ \alpha) p(x_{\cdot j}|z_j, \Theta_{\cdot z_j}) \Big).$$

The joint distribution displays how the likelihood of a data point depends on the corresponding cluster assignment z_j and cluster parameters $\Theta_{\cdot z_j}$. We can sum over the cluster assignments and give the likelihood of the data using soft clustering and conditioning on the cluster parameters:

$$p(X|\Theta, \alpha) = \prod_{j}^{N} \Big(\sum_{i}^{K} p(z_j = i|\alpha) p(x_{\cdot j}|z_j = i, \Theta_{\cdot i}) \Big).$$

The individual distributions in the joint distribution are predetermined for other variables than cluster parameters Θ. As was mentioned, the data is drawn from a Dirichlet distribution and the cluster assigments are naturally drawn from a discrete distribution with possible states $z_j \in \{1, 2, 3, ..., K\}$, where K is the number of clusters. The choice of distribution for Θ will be discussed in the next section.

2.4.2 Parameter estimation

The problem with this generative model is estimating the parameters. Exact inference is intractable, and achieving decent approximate posterior inference also proves difficult. Examining the posterior distribution allows us to give a prior distribution for each different variable, and then sharpen our knowledge by conditioning on the data. Markov Chain Monte Carlo (MCMC) methods can be used to collect samples from the posterior distributions of each different variable, always conditioning on the data and the state of the other latent variables. Here two MCMC methods for approximate posterior inference are discussed: Gibbs sampling and the Metropolis-Hastings algorithm (Gelman et al., 2003).

In generative topic modeling, like Latent Dirichlet Allocation (Blei et al., 2003), it is possible to use conjugate prior distributions so that the posterior distributions are of the same known form as the prior, just with different parameters. As an example about conjugate prior distributions we can think of making a test which has two possible results: 1 or 0. Clearly the probability distribution to model this test is a Bernoulli distribution. The density

function of a Bernoulli distribution is given by

$$p(x|q) = q^x(1-q)^{1-x} = q^a(1-q)^b; a, b \in \{0,1\}, a+b=1,$$

where q is the probability of 1 and $1-q$ corresponds to the probability of 0. Now if we want to give a prior distribution also to q, we can try the beta distribution, which is defined as

$$p(q|\alpha,\beta) = \frac{1}{B(\alpha,\beta)} q^{\alpha-1}(1-q)^{\beta-1},$$

where $B(\alpha,\beta)$ is the beta function. Now the posterior distribution $p(q|x,\alpha,\beta)$ is given by

$$p(q|x,\alpha,\beta) \propto p(q|\alpha,\beta)p(x|q) = q^{a+\alpha-1}(1-q)^{b+\beta-1},$$

which shows that the posterior distribution is of the same form as the prior.

The advantage in using conjugate prior distributions comes from the fact that often we can give different variables distributions that we know how to sample from. Then naturally if the posterior is of the same form as the prior we gave, we can sample from the posterior distributions as well. However, the conjugate prior of the Dirichlet is not a standard distribution, and even though we can solve it up to a normalizing constant (Lefkimmiatis et al., 2009), there is no method available to directly sample from it. One is then forced to use the Metropolis-Hastings algorithm instead of Gibbs sampling.

With Gibbs sampling, the idea is to initialize the variables somewhere and then repeatedly sample from each of the posterior distributions of different latent variables in turn, always conditioning with the data and the other latent variables. This procedure then produces a Markov chain as each different new sample only depends on the current state of the chain and not the entire history. Because Gibbs sampling operates by sampling directly from the posterior distributions, the Markov chain can move freely in the area where there is some likelihood, and finally ends up in an area of high posterior probability for the parameters.

In contrast, Metropolis-Hastings algorithm is generally used when straightforward sampling is not possible. In Metropolis-Hastings, we choose an appropriate jumping distribution from which we randomly draw a new sample, and then test whether it is a point worth going to based on its posterior likelihood. Intuitively speaking the jumps take the previous value of the latent

variable in question as a starting point and make a jump to some point in the neighborhood. The choice of jumping distribution influences how the new point is chosen in this procedure.

Compared to Gibbs sampling the difficulty in using Metropolis-Hastings algorithm is to find a jumping distribution that is able to move around the possible parameter space efficiently: it should move with large enough jumps so that the chain doesn't get stuck in a single area of locally high probability, but also small enough so jumps are actually accepted with a reasonable rate (Andrieu et al., 2003).

In this specific model the cluster assignments z_i can be found using Gibbs sampling as the cluster probabilities can be updated with the help of the Bayes theorem. The Bayes theorem states that

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}.$$

In this case A corresponds to $p(z_i = k)$ while B is the distribution of data X and clusters Θ. Although the probability distribution describing the data is continuous in the form of a Dirichlet, we can use its likelihoods in place of discrete probabilities to take advantage of the Bayes theorem:

$$p(z_i = k \mid x, \Theta) = \frac{p(z_i = k)p(x_i|\Theta_{\cdot k}, z_i = k)}{\sum_{j=1}^{K} p(z_i = j)p(x_i|\Theta_{\cdot j}, z_i = j)}.$$

In this case the prior probabilities would just be equal numbers depending on the number of clusters that sum up to one.

The problem here is the sampling of the cluster parameters, Θ in Figure 1. First we have to decide what distribution we put the clusters to be generated from. Possible alternatives may include the Dirichlet conjugate prior and the Dirichlet distribution itself. Both require using the Metropolis-Hastings algorithm to get samples from the posterior distribution. The choice is especially relevant for making it as easy as possible for the sampling procedure to work properly. One of the reasons using conjugate prior distributions is generally favorable is naturally that it makes the sampling much easier and more reliable by allowing straightforward Gibbs sampling.

Possible jumping distributions in these cases may include (corresponding cluster distributions are in brackets)

- multivariate normal distribution (Dirichlet conjugate prior),

- Dirichlet distribution (Dirichlet),
- gamma distribution (Dirichlet conjugate prior).

When using the multivariate normal distribution the non-negativity constraint of the parameters of a Dirichlet distribution has to be taken into consideration. This problem can be solved for instance by taking the logarithm of parameters x and making the jumps in that space, and then converting back with $\exp(x)$. With a Dirichlet distribution one has to choose an appropriate multiplier for the current values of the clusters so that the new guess would be relatively close to the old one. When using the gamma distribution one should choose the shape parameter as some positive constant C and the scale parameter as θ_{t-1}/C, where θ_{t-1} is the current value of the variable in question. This way the expectation of the gamma distribution, scale parameter $\times$ shape parameter, is equal to the current value, while the constant C controls the variance of the distribution. The computational implications of the different choices will be further discussed in later sections.

2.4.3 Generative models for component analysis

In addition to the already presented probabilistic clustering model, generative modeling could be considered to find components in data. This basically requires doing different assumptions about the Dirichlet parameters Θ and the cluster assignments Z. The data, although continuous, correlates significantly with the absolute quantities of different bacteria (Rajilic-Stojanovic et al., 2009). This may allow the use of Latent Dirichlet Allocation (Blei et al., 2003) by discretizing the data and treating the numbers as counts of different bacteria. In Latent Dirichlet Allocation each data sample would be treated as a bag of bacteria. The components would be Dirichlet distributed and would describe the probability of a certain bacterium presenting itself from that specific component. Each bacterium from the data would be assigned a component from which it comes from, and all the data samples would be assigned a vector that includes the probabilities of choosing different components.

Models that may avoid the discretization resemble the generative model in Figure 1. As a change the models would assign a vector of component weights instead of a cluster assignment to each data sample. As the computational difficulties regarding these models would most likely be even harder to tackle than the more simple clustering model, this study will concentrate only on

the model presented by Figure 1. Also the implementation of Latent Dirichlet Allocation is left for later examination.

3 Results

3.1 Clustering

In this section first the results of different validation techniques will be presented that then decide the number of clusters we choose for the partitioning around medoids algorithm to find.

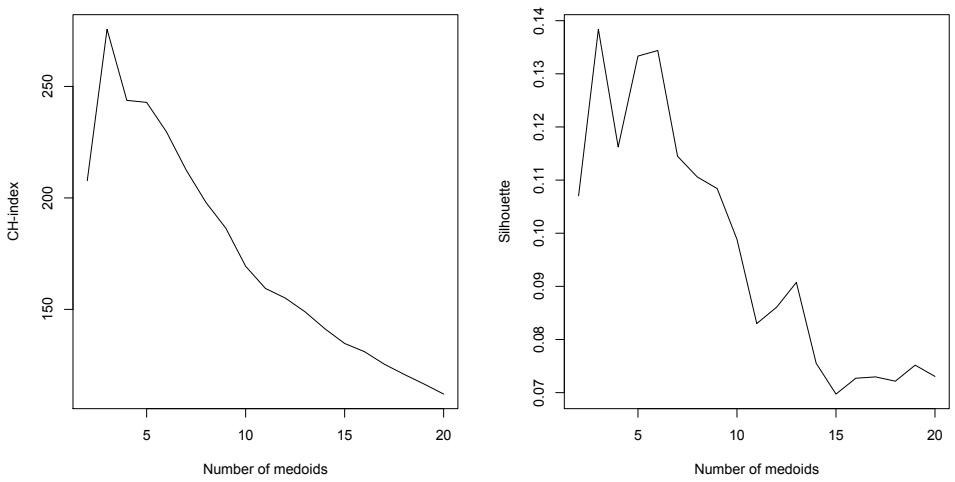

Figure 2: *Calinski-Harabasz index and Silhouette results.*

Figure 2 displays the clustering validation results using CH-index and average Silhouette. Both plots indicate that three clusters is the optimal choice for this data. This is consistent with the work of Arumugam et al. (2011). Figure 2 also suggests that choosing more than six clusters seems totally unfounded. The value of both indices starts quickly decreasing with seven clusters and above. The behavior of CH-index and average Silhouette is further examined in Appendix A, where the results they give were cross-validated. As can be seen from Figure 9, the results resemble a lot those of Figure 2.

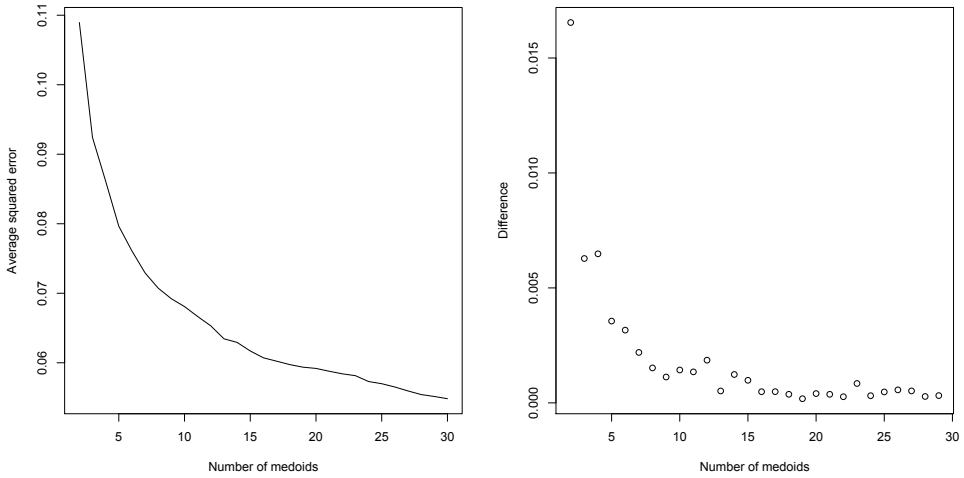

Figure 3: *Clustering cross-validation results. The average squared distance from the closest cluster medoid for validation set samples is drawn on the Y-axis.*

Figure 3 shows the clustering cross-validation results and is not as conclusive as Figure 2 in determining the number of clusters to pick up. On the right the absolute value of the "derivative" of the cross-validation curve is presented, meaning that the first point in the figure describes the value of "result with two medoids - result with three medoids". This helps identify when the improvement starts slowing down. We see that although there is some room for interpretation, Figure 3 is not really contradictory to Figure 2. There is a big improvement from two clusters to three, but then the improvement clearly slows down. The maximum number of clusters one could see as the optimum seems to be seven as after that the improvement is steady and slow.

Because the validation techniques limit the possible number of clusters to be between three and seven it is useful to determine how the found clusters behave in this interval. The expectation is that the same center points that are found with three clusters more or less remain and then new clusters are formed from the edges of the data. This would also indicate that the original three clusters are robust and not only some centroids that are in the center but actually are not able to describe the data well.

Luckily the data is sparse in the sense that most families of bacteria have very low counts throughout the samples. Thus the few that have the largest variance are the ones that mostly decide the cluster assignments. This makes visualizing the results much easier.

With three found clusters the centers are driven by the following bacteria:

- Prevotella,
- Ruminococcus,
- Subdoligranulum and Faecalibacterium.

Out of these three, two are consistent with the recently found enterotypes in smaller data sets Arumugam et al. (2011). With five clusters all these clusters remain similar with additional clusters being driven by Bacteroides and Streptococcus. Figures 4 and 5 illustrate the differences between the samples that belong to each cluster.

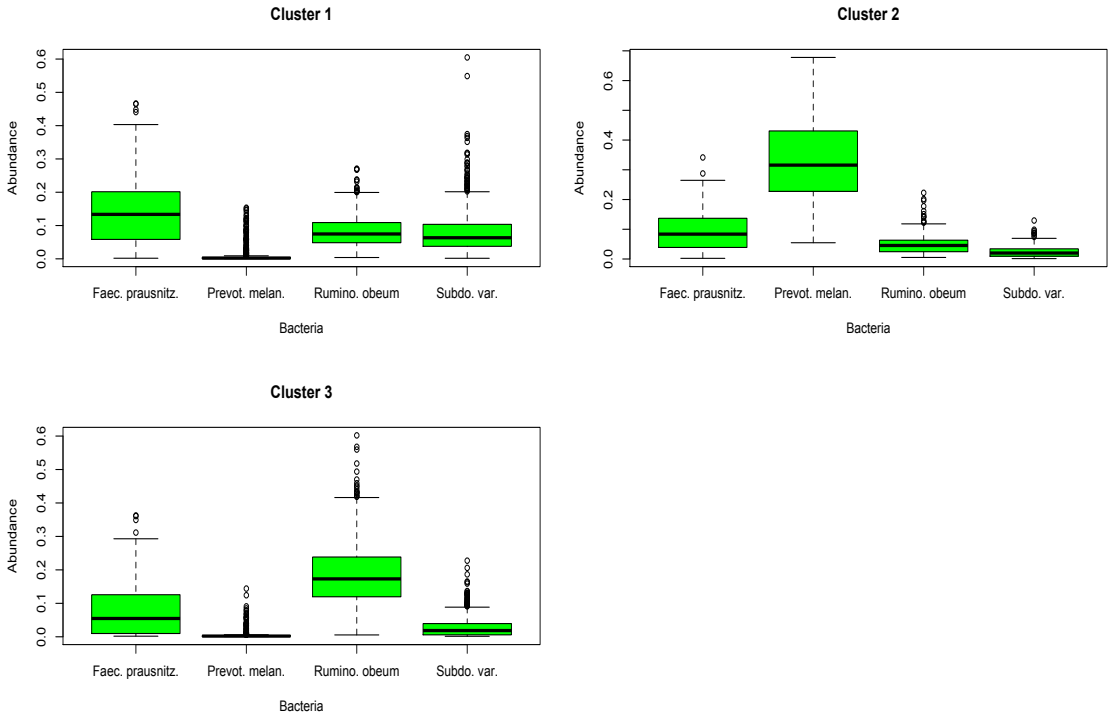

Figure 4: *Abundances of four different bacteria with three clusters. The bacteria are from left to right Faecalibacterium prausnitzii et rel., Prevotella melaninogenica et rel., Ruminococcus obeum et rel., and Subdoligranulum variable et rel..*

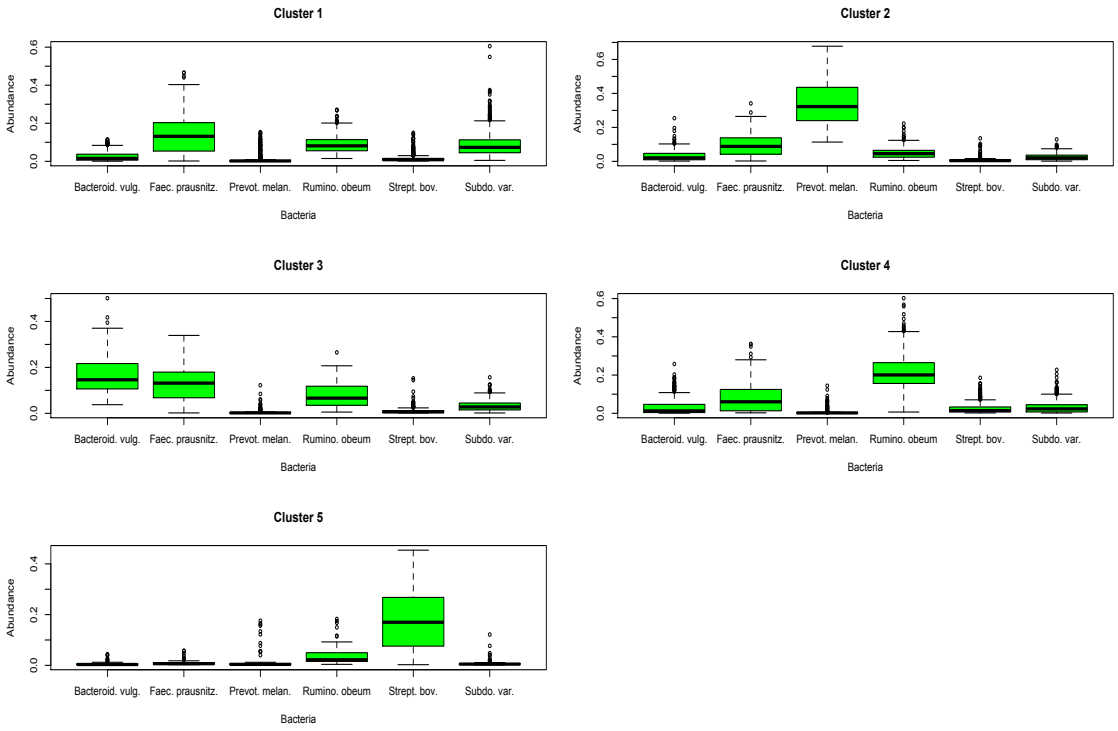

Figure 5: *Abundances of six different bacteria with five clusters. The bacteria are from left to right Bacteroides vulgatus et rel., Faecalibacterium prausnitzii et rel., Prevotella melaninogenica et rel., Ruminococcus obeum et rel., Streptococcus bovis et rel. and Subdoligranulum variable et rel..*

Table 1: Number of samples belonging to each cluster.

Driver	3 clusters	5 clusters
Faec. and Subd.	956	770
Prevotella	208	200
Ruminococcus	799	601
Bacteroides	-	300
Streptococcus	-	92

With seven clusters an additional cluster with high abundance of Bifidobacterium presents itself. Some of the other clusters already start to mix slightly, although Prevotella, Ruminococcus, Bacteroides and Streptococcus clusters remain intact. These results indicate that there are only a handful of bacterial profiles that are robust because the clusters found here are mostly consistent with the ones found by Arumugam et al. (2011).

Table 1 displays the number of points that belong to each cluster when grouping with either three or five clusters. It should be noted that the clusters with relatively high abundances of Faecalibacterium and Subdoligranulum are the most common in the data set of approximately 2000 samples. Also the cluster driven by Ruminococcus is very common. On the other hand especially the cluster with a high relative amount of Sreptococcus is small compared to the others. This may indicate that it is specific to a certain condition or illness.

3.2 Component analysis

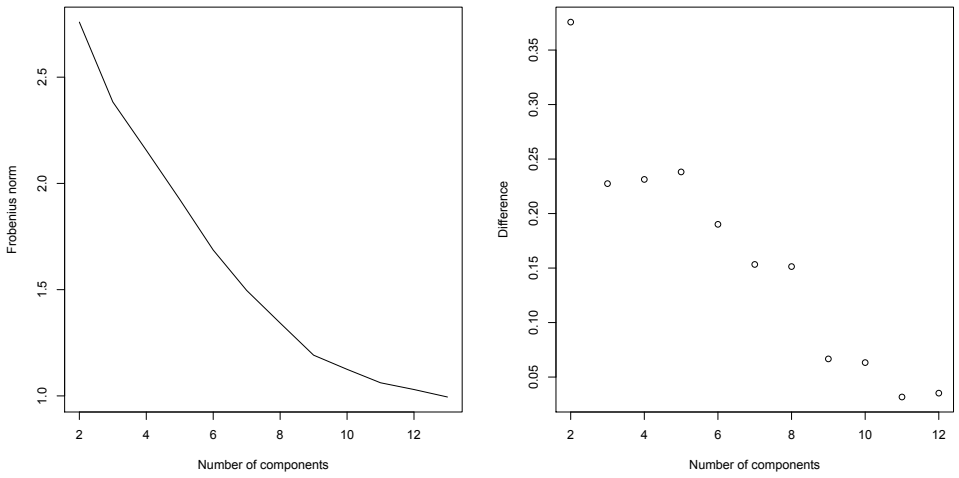

Figure 6: *semi-NMF cross-validation results. Average frobenius norm of* $X - FG^T$ *is drawn on the Y-axis on the left-hand side. The absolute value of the "derivative" of the curve is on the right.*

In this section the cross-validation results of semi-NMF will be discussed similarly to previous section before the actual components are described. Figure 6 shows that the benefit of an additional component clearly decreases when the amount of components is nine. This indicates that nine components is the optimal value to pick up from the data. Differences between components are displayed in Figure 7 with the bacteria that have the largest variance between the components.

Figure 7 shows that the components found for the most part resemble the clusters themselves as is expected. Perhaps the most striking relationship can be seen from components two and six. When the amount of Streptococcus bovis rises, the amounts of the bacteria that constitute the most common bacterial profile, Faecalibacterium and Subdoligranulum, become smaller, and vice versa. The fact that we took nine components (more than the amount of clusters tested) is probably what makes most components to be mostly about a single component. Also the sparsity of the data contributes to this because most of the groups in the data set need not be even taken into consideration during the analysis. It seems that although the amount

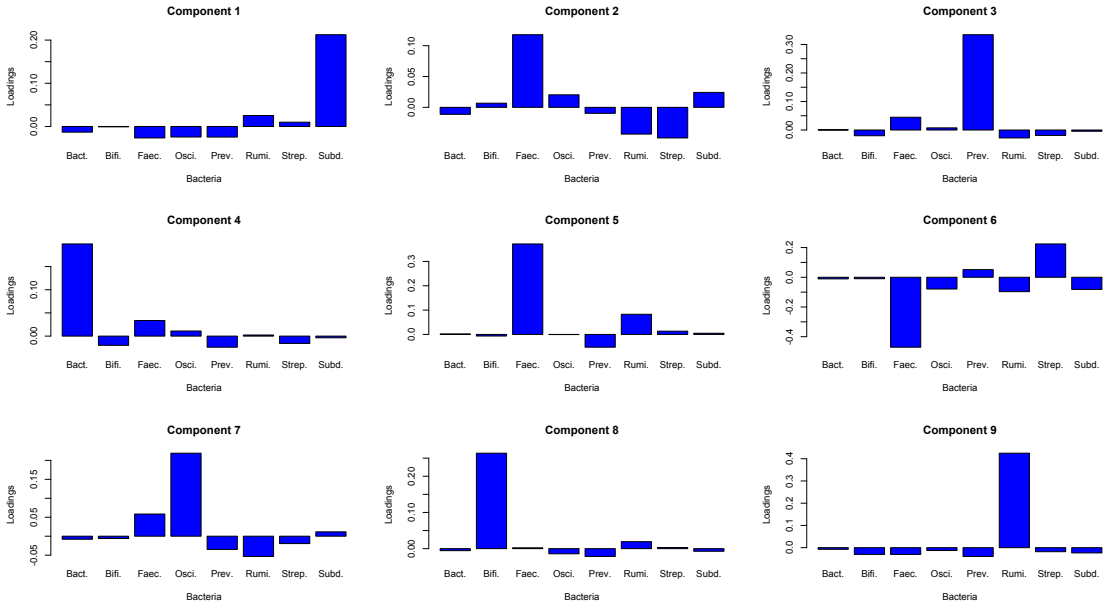

Figure 7: *Nine components found with semi-NMF. Names of the bacteria are shortened to fit, but they the following from left to right: Bacteroides vulgatus et rel., Bifidobacterium, Faecalibacterium prausnitzii et rel., Oscillospira guillermondii et rel., Prevotella melaninogenica et rel., Ruminococcus obeum et rel., Streptococcus bovis et rel., Subdoligranulum variable et rel..*

of components is not large at all compared to the amount of bacteria in the data (130), the data is so sparse that already this amount of components produces overlap between them. This overlap gives encouragement to try a lower amount of components to make inferences.

Figure 8 displays the components when the amount is decreased to seven. Clearly a positive correlation between Oscillospira and Subdoligranulum becomes visible in the first component.

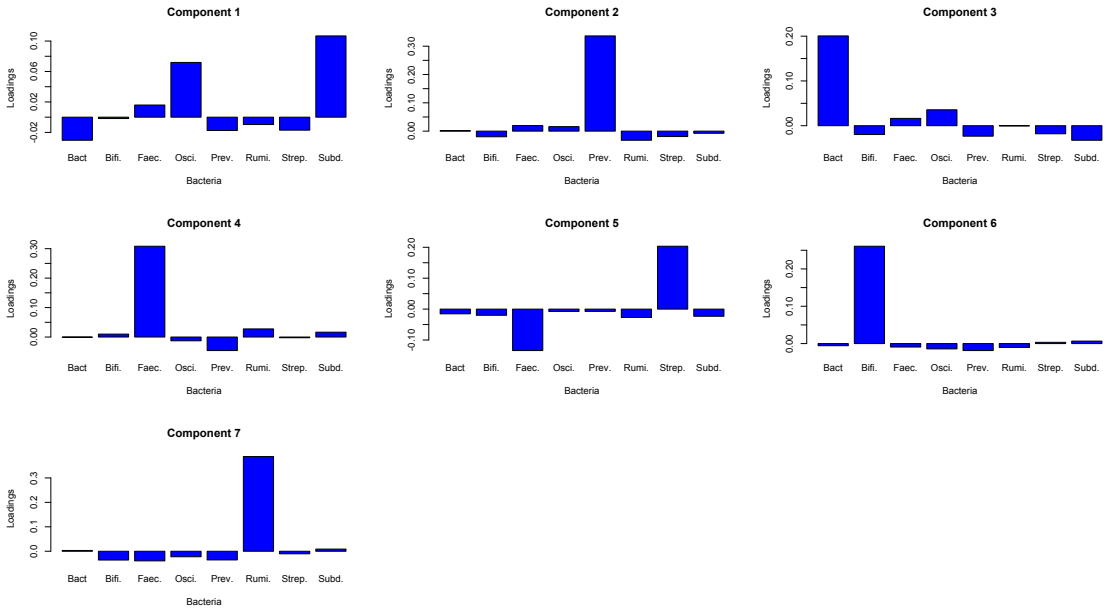

Figure 8: *Seven components found with semi-NMF. Names of the bacteria are the same as in Figure 7.*

3.3 Generative model

Figure 1 describes the generative model with a plate diagram. As discussed earlier, the main difficulty with this model is to find a way to achieve decent behavior with approximate posterior inference using Markov Chain Monte Carlo methods. Cluster assignments Z can be drawn by building a Gibbs sampler as was already presented, but the more difficult part is to build a Metropolis-Hastings algorithm that can sequentially draw the cluster matrix Θ and actually converge to an area where the parameters have high posterior probability.

To achieve this, three different possible jumping distributions were considered. It turns out that using the Dirichlet distribution seems to be the only one that may be possibly viable without extra modifications. This is mostly due to the nature of the data distribution itself. Because the data vectors consist of a 130 dimensional Dirichlet distribution, the likelihood function is revealed to be extremely spiky in the areas where likelihood exists. This results in two problems. First, if the Metropolis-Hastings jumps end up in an area where there is some likelihood, it will have basically zero probability of jumping away from there as the surrounding area has minimal values for the likelihood function. Secondly, the large differences in likelihoods also make the cluster assignments get stuck in certain clusters and make them not to mix naturally.

If we choose the prior for the parameters to also be a Dirichlet distribution, the parameters naturally are restricted to be between zero and one, and to sum up to one. If the parameter values would be allowed to grow freely, increasing the parameter values around the data points would produce higher and higher likelihoods and even result in a computational overflow due to the gamma functions getting extremely high values in the likelihood function. By restricting the parameters to remain small, the likelihoods also remain lower.

This restriction when using the Dirichlet does not take away the model's predicted ability to find cluster vectors, because the relative values within the cluster centroids are much more important than the absolute ones. Additionally restricting the likelihood values to be smaller should in theory allow the Markov chain to mix better, both in terms of the cluster assignment vector and the cluster matrix. As the likelihoods remain smaller, the probabilities of assigning a certain point to a cluster should remain more reasonable.

However, if we were to choose either the normal distribution or the gamma distribution to choose the new candidate values, the ability of the model could

be hindered. If we wanted to limit the parameters from rising too high, we should initialize them close to zero and use extremely tight prior distributions so that the likelihoods of the Dirichlet conjugate prior would be small. This could also hinder the relative changes in the values as the prior part of the posterior would give high values only in a very small area.

In the end it turns out that even choosing the Dirichlet distribution as a prior and as a jumping distribution doesn't completely solve the problem. With small test sets it works nicely, but the large dimension of the actual data set still gives problems. The fact remains that the likelihood of one cluster may be in the range of 10^{30} times the likelihood of another cluster, which clearly then results in the posterior probabilities being very close to one for that cluster, and basically zero for others. This behavior makes the Markov chain to get stuck in a small area, which means that the model doesn't really provide any new or reliable results.

The next thing to try is to smoothen the likelihood function. Smoothing was tried during the burn-in period by taking the tenth root of the likelihoods when choosing the cluster assignments. The power is then linearly increased until, in the end of the burn-period, the sampler uses the actual likelihoods.

However, even with the smoothing the algorithm ends up in a state where most of the points are stuck to a single cluster, which prevents the clusters themselves to properly update based on the data. The smoothing helps during the first rounds of the iteration but when the smoothing decreases, the same problem appears once again. This means that the generative model proves to be computationally too difficult to be able to produce reliable results in this special study.

4 Discussion and conclusions

Using a data set that originated from the use of HITChip technology (Rajilic-Stojanovic et al., 2009) with human fecal samples, the goal of this special study was to present, analyze and implement clustering and component analysis tools for unsupervised learning. In addition, different validation methodologies were discussed and used. In terms of clustering, significant results were found using the partitioning around medoids algorithm. The results partly correspond to the enterotypes presented by Arumugam et al. (2011), but also provide new clusters that clearly separate themselves from the other samples.

The clusters that were found can be found reliable and significant as similar cluster medoids were picked up even when the cluster count was lifted from the original three. This indicates that the clusters are in fact present in the data set, and the found centroids aren't only some random points in the middle of the data set. Different validation techniques were considered and used, all of which indicated that the real number of clusters present in the data is no more than seven even with close to 2000 samples. The results seem to be a sign that there are not that many stable bacterial compositions that may be present inside the human digestive system. This once again corresponds to the findings of Arumugam et al. (2011).

With component analysis the results are in many ways in accordance with the clustering results. The components represent the bacteria that have the largest variance between the data samples, and basically each of the significant bacteria that drove their own clusters also have their own components. While not many significant or surprising relationships between different bacteria were discovered, the main contribution of these findings is perhaps the chance to be able to examine the scores that each data sample has for different components. This examination may provide information on which samples are heavily related to certain components and thus provide interesting research opportunities. However, this examination is left outside this special study because the background information about the samples was extremely incomplete.

The generative model for clustering was deemed computationally difficult due to the nature of the Dirichlet distribution and it didn't yet provide any usable results. In future attempts the concentration should first be on finding appropriate measures to take which would allow for the model to mix better instead of getting stuck in a certain, suboptimal position. This would require more examination into smoothing the likelihood function, and perhaps also considering different ways of implementing the jumping distribution. For instance, using the normal distribution as the Metropolis-Hastings jumping distribution and changing only one variable at the time may be able to find new suggestions better than simple Dirichlet jumps do. However, this implementation would require some upper limits for the variables to avoid additional computational issues. Additionally even longer burn-in periods could be pursued so that the cluster centers would have time to update themselves to reasonable locations. Currently the burn-in period was set to at most 2000 iterations, but it still was not long enough for the smoothing to be effective.

5 References

Alpaydin, E. (2010). *Introduction to Machine Learning*. The MIT Press, 2nd edition.

Andrieu, C., de Freitas, N., Doucet, A. and Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43.

Arumugam, M., Raes, J., Pelletier, E., Le Paslier, D., Yamada, T., Mende, D. R., Fernandes, G. R., Tap, J., Bruls, T., Batto, J.-M., Bertalan, M., Borruel, N., Casellas, F., Fernandez, L., Gautier, L., Hansen, T., Hattori, M., Hayashi, T., Kleerebezem, M., Kurokawa, K., Leclerc, M., Levenez, F., Manichanh, C., Nielsen, H. B., Nielsen, T., Pons, N., Poulain, J., Qin, J., Sicheritz-Ponten, T., Tims, S., Torrents, D., Ugarte, E., Zoetendal, E. G., Wang, J., Guarner, F., Pedersen, O., de Vos, W. M., Brunak, S., Dore, J., Antolin, M., Artiguenave, F., Blottiere, H. M., Almeida, M., Brechot, C., Cara, C., Chervaux, C., Cultrone, A., Delorme, C., Denariaz, G., Dervyn, R., Foerstner, K. U., Friss, C., van de Guchte, M., Guedon, E., Haimet, F., Huber, W., van Hylckama-Vlieg, J., Jamet, A., Juste, C., Kaci, G., Knol, J., Lakhdari, O., Layec, S., Le Roux, K., Maguin, E., Merieux, A., Melo Minardi, R., M'rini, C., Muller, J., Oozeer, R., Parkhill, J., Renault, P., Rescigno, M., Sanchez, N., Sunagawa, S., Torrejon, A., Turner, K., Vandemeulebrouck, G., Varela, E., Winogradsky, Y., Zeller, G., Weissenbach, J., Ehrlich, S. D. and Bork, P. (2011). Enterotypes of the human gut microbiome. *Nature*, 473(7346):174–180.

Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003). Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3(Jan):993–1022.

Calinski, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27.

Comon, P. (1994). Independent component analysis, a new concept? *Signal Process.*, 36(3):287–314.

DeAngelis, K., Wu, C., Beller, H., Brodie, E., Chakraborty, R., DeSantis, T., Fortney, J., Hazen, T., Osman, S., Singer, M., Tom, L. and Andersen, G. (2011). PCR Amplification-Independent Methods for Detection of Microbial Communities by the High-Density Microarray PhyloChip. *Applied and Environmental Microbiology*, 77(18):6313–6322.

Ding, C., Li, T. and Jordan, M. I. (2010). Convex and Semi-Nonnegative Matrix Factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55.

Endres, D. M. and Schindelin, J. E. (2003). A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860.

Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. (2003). *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd edition.

Hennig, C. (2010). *fpc: Flexible procedures for clustering*. R package version 2.0-3.

Hennig, C. and Liao, T. F. (2010). Comparing latent class and dissimilarity based clustering for mixed type variables with application to social stratification. Research report no. 308, Department of Statistical Science, UCL.

Kaufman, L. and Rousseeuw, P. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Interscience, New York.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lefkimmiatis, S., Maragos, P. and Papandreou, G. (2009). Bayesian inference on multiscale models for Poisson intensity estimation: applications to photon-limited image denoising. *Trans. Img. Proc.*, 18(8):1724–1741.

Maisog, J. (2009). Non-negative matrix factorization: Assessing methods for evaluating the number of components, and the effect of normalization thereon. Master's thesis, Georgetown University.

Maulik, U. and Bandyopadhyay, S. (2002). Performance Evaluation of Some Clustering Algorithms and Validity Indices. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(12):1650–1654.

Milligan, G. and Cooper, M. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

Rajilic-Stojanovic, M., Heilig, H. G. H. J., Molenaar, D., Kajander, K., Surakka, A., Smidt, H. and De Vos, W. M. (2009). Development and application of the human intestinal tract chip, a phylogenetic microarray: analysis of universally conserved phylotypes in the abundant microbiota of young and elderly adults. *Environmental Microbiology*, 11(7):1736–1751.

Rousseeuw, P. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65.

van der Laan, M., Pollard, K. and Bryan, J. (2003). A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73(8):575–584.

A Cross-validation of CH-index and Silhouette

This appendix includes the plots showing the cross-validation of CH-index and Silhouette. The cross-validation was done for the purpose of making sure the indices give as reliable results as possible. First the data was divided into folds and clustering was done with partitioning around medoids algorithm for each training set. Then the optimal clustering, in terms of the square root of the Jensen-Shannon divergence, was found for each validation set. Corresponding CH-indices and average Silhouettes were then averaged over validation sets. In this case only five folds were used because with ten folds (smaller validation sets, larger training sets) all clusters weren't given any points from the validation sets when the amount of clusters was close to ten.

The results of the cross-validation are presented in Figure 9. The plots show that CH-index indeed gives three as an optimum also across the validation sets. Silhouette on the other hand gives two as a clear optimum, but the rest of the curve is very similar to Figure 2.

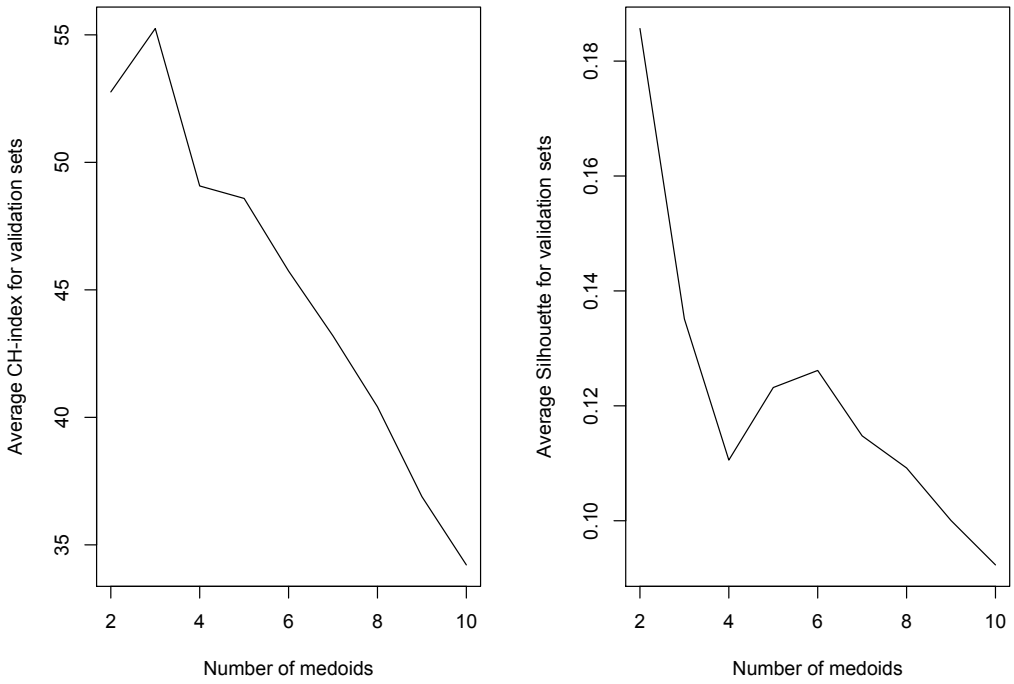

Figure 9: *Cross-validation of CH-index and Silhouette.*