Doc Start BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–10

# SEK: Sparsity exploiting $k$-mer-based estimation of bacterial community composition

Saikat Chatterjee [1]*, David Koslicki [2], Siyuan Dong [3], Nicolas Innocenti [4], Lu Cheng [5], Yueheng Lan [6], Mikko Vehkaperä [1,8], Mikael Skoglund [1], Lars K. Rasmussen [1], Erik Aurell [4,7], Jukka Corander [5]

[1]Dept of Communication Theory, KTH Royal Institute of Technology, Sweden
[2]Dept of Mathematics, Oregon State University, Corvallis, USA
[3]Systems Biology program, KTH Royal Institute of Technology, Sweden and Aalto University, Finland
[4]Dept of Computational Biology, KTH Royal Institute of Technology, Sweden
[5]Dept of Mathematics and Statistics, University of Helsinki, Finland
[6]Dept of Physics, Tsinghua University, Beijing, China
[7]Dept of Information and Computer Science, Aalto University, Finland
[8]Dept of Signal Processing, Aalto University, Finland

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

**ABSTRACT**

Motivation: Estimation of bacterial community composition from a high-throughput sequenced sample is an important task in metagenomics applications. Since the sample sequence data typically harbors reads of variable lengths and different levels of biological and technical noise, accurate statistical analysis of such data is challenging. Currently popular estimation methods are typically very time consuming in a desktop computing environment.

Results: Using sparsity enforcing methods from the general sparse signal processing field (such as compressed sensing), we derive a solution to the community composition estimation problem by a simultaneous assignment of all sample reads to a pre-processed reference database. A general statistical model based on kernel density estimation techniques is introduced for the assignment task and the model solution is obtained using convex optimization tools. Further, we design a greedy algorithm solution for a fast solution. Our approach offers a reasonably fast community composition estimation method which is shown to be more robust to input data variation than a recently introduced related method.

Availability: A platform-independent Matlab implementation of the method is freely available at http://www.ee.kth.se/ctsoftware; source code that does not require access to Matlab is currently being tested and will be made available later through the above website.

## 1 INTRODUCTION

High-throughput sequencing technologies have recently enabled detection of bacterial community composition at an unprecedented level of detail. The high-throughput approach focuses on producing for each sample a large number of reads covering certain variable part of the 16S rRNA gene, which enables an identification and comparison of the relative frequencies of different taxonomic units present across samples. Depending on the characteristics of the samples, the bacteria involved and the quality of the acquired sequences, the taxonomic units may correspond to species, genera or even higher levels of hierarchical classification of the variation existing in the bacterial kingdom. However, at the same time, the rapidly increasing sizes of read sets produced per sample in a typical project call for fast inference methods to assign meaningful labels to the sequence data, a problem which has attracted considerable attention [25, 19, 18, 21].

Many approaches to the bacterial community composition estimation problem use 16S rRNA amplicon sequencing where thousands to hundreds of thousands of moderate length (around 250-500 bp) reads are produced from each sample and then either clustered or classified to obtain estimates of the prevalence of any particular taxonomic unit. In the clustering approach the reads are grouped into taxonomic units by either distance-based or probabilistic methods [8, 13, 12], such that the actual taxonomic labels are assigned to the clusters afterwards by matching their consensus sequences to a reference database. Recently, the Bayesian BeBAC method [12] was shown to provide high biological fidelity in clustering. However, this accuracy comes with a substantial computational cost such that a running time of several days in a computing-cluster environment may be required for large read sets. In contrast to the clustering methods, the classification approach is based on using a reference database directly to assign reads to meaningful units representing biological variations. Methods for the classification of reads have been based either on homology using sequence similarity or on genomic signatures in terms of oligonucleotide composition. Examples of homology-based methods include MEGAN [17, 20] and phylogenetic analysis [24]. A popular approach is the Ribosomal Database Project's (RDP)

*To whom correspondence should be addressed. Email: sach@kth.se

classifier which is based on a naïve Bayesian classifier (NBC) that assigns a label explicitly to each read produced for a particular sample [25]. Despite the computational simplicity of NBC, the RDP classifier may still require several days to process a data set in a desktop environment. Given this challenge, considerably faster methods based on different convex optimization strategies have been recently proposed [19, 18]. In particular, sparsity-based techniques, mainly compressive sensing based algorithms [9], are used for estimation of bacterial community composition in [3, 18, 27]. However, [3] used sparsity-promoting algorithms to analyze mixtures of dye-terminator reads resulting from Sanger sequencing, with the sparsity assumption that each bacterial community is comprised of a small subset of known bacterial species, the scope of the work thus being different from methods intended for high-throughput sequence data. The Quikr method of [18] uses a $k$-mer-based approach on 16S rRNA sequence reads and has a considerable similarity to the method (SEK: Sparsity Exploiting K-mers-based algorithm) introduced here. Explained briefly, the Quikr setup is based on the following core theoretical formulation: given a reference database $D = \{d_1, \ldots, d_M\}$ of sequences and a set $S = \{s_1, \ldots, s_t\}$ of sample sequences (the reads to be classified), it is assumed that there exists a unique $d_j$ for each $s_l$, such that $s_l = d_j$. In general, all reference databases and sample sets consist of sequences with highly variable lengths. In particular the lengths of reference sequences and samples reads are often quite different. Violation of the assumption leads to sensitivity in Quikr performance according to our experiments. Another example of fast estimation is called Taxy [19] that addresses the effect of varying sequence lengths [26]. Taxy uses a mixture model for the system setting and convex optimization for a solution. The method referred to as COMPASS [2] is another convex optimization approach, very similar to the Quikr method, that uses large $k$-mers and a divide-and-conquer technique to handle very large resulting training matrices. The currently available version of the Matlab-based COMPASS software does not allow for training with custom databases, so a direct comparison to SEK is not yet possible.

To enable fast estimation, we adopt an approach where the estimation of the bacterial community composition is performed jointly, in contrast to the read-by-read analysis used in the RDP classifier. Our model is based on kernel density estimators and mixture density models [6], and it leads to solving an under-determined system of linear equations under a particular sparsity assumption. In summary, the SEK approach is implemented in three separate steps: off-line computation of $k$-mers using a reference database of 16S rRNA genes with known taxonomic classification, on-line computation of $k$-mers for a given sample, and then final on-line estimation of the relative frequencies of taxonomic units in the sample by solving an under-determined system of linear equations.

## 2 METHODS

### 2.1 General notation and computational resources used

We denote the non-negative real line by $\mathbb{R}_+$. The $\ell_p$ norm is denoted by $\|.\|_p$, and $\mathbb{E}[.]$ denotes the expectation operator. Transpose of a vector/matrix is denoted by $(.)^t$. We denote cardinality and complement of a set $\mathcal{S}$ by $|\mathcal{S}|$ and $\overline{\mathcal{S}}$, respectively. In the computations reported in the remainder of the paper we used standard Matlab software with some instances of C code. For experiments on mock community data, we used a Dell Latitude E6400 laptop computer with a 3 GHz processor and 8 GB memory. We also used the **cvx** [7] convex optimization toolbox and the Matlab function **lsqnonneg**() for a least-squares solution with non-negativity constraint. For experiments on simulated data, we used standard computers with an Intel Xeon x5650 processor and an Intel i7-4930K processor.

### 2.2 $k$-mer training matrix from reference data

The training step of SEK consists of converting an input labeled database of 16S rRNA sequences into a $k$-mer training matrix. For a fixed $k$, we calculate $k$-mers feature vectors for a window of fixed length, such that the window is shifted (or slid) by a fixed number of positions over a database sequence. This procedure captures variability of localized $k$-mer statistics along 16S rRNA sequences. Using bp as the length unit and denoting the length of a reference database sequence $d$ by $L_d$, and further a fixed window length by $L_w \leq L_d$ and the fixed position shift by $L_p$, the total number of sub-sequences processed to $k$-mers is close to $\lfloor \frac{L_d - L_w}{L_p} \rfloor$. The choice of $L_w$ may be decided by the shortest sample sequence length that is used in the estimation assuming the reads in a sample set are always shorter than the reference training sequences. In practice, for example, we used $L_w = 450$ bp in experiments using mock communities data. The choice of $L_p$ is decided by the trade-off between computational complexity and estimation performance.

Given a database of reference training sequences $D = \{d_1, \ldots, d_M\}$ where $d_m$ is the sequence of the $m$th taxonomic unit, each sequence $d_m$ is treated independently. For $d_m$, the $k$-mer feature vectors are stored column-wise in a matrix $\mathbf{X}_m \in \mathbb{R}_+^{4^k \times N_m}$, where $N_m \approx \lfloor \frac{L_{d_m} - L_w}{L_p} \rfloor$. From the training database $D$, we obtain the full training matrix

$$\begin{aligned} \mathbf{X} = & \quad [\mathbf{X}_1 \, \mathbf{X}_2 \, \ldots, \mathbf{X}_M] \; \in \mathbb{R}_+^{4^k \times N}, \\ \equiv & \quad [\mathbf{x}_1 \, \mathbf{x}_2 \, \ldots \mathbf{x}_N], \end{aligned}$$

where $\sum_{m=1}^{M} N_m = N$, and $\mathbf{x}_n \in \mathbb{R}_+^{4^k \times 1}$ denotes the $n$th $k$-mers feature vector in the full set of training feature vectors $\mathbf{X}$.

### 2.3 SEK model

For the $m$th taxonomic unit, we have the training set

$$\mathbf{X}_m = [\mathbf{x}_{m1} \, \mathbf{x}_{m2} \, \ldots \mathbf{x}_{mN_m}] \; \in \mathbb{R}_+^{4^k \times N_m},$$

where we used an alternative indexing to denote the $l$th $k$-mer feature vector by $\mathbf{x}_{ml}$. Letting $\mathbf{x}$ and $\mathcal{C}_m$ denote random $k$-mer feature vectors and $m$th taxonomic unit respectively, and using $\mathbf{X}_m$, we first model the conditional density $p(\mathbf{x}|\mathcal{C}_m)$ corresponding to $m$th unit by a mixture density as

$$p(\mathbf{x}|\mathcal{C}_m) = \sum_{l=1}^{N_m} \alpha_{ml} \; p_{ml}(\mathbf{x}|\mathbf{x}_{ml}, \Theta_{ml}), \tag{1}$$

where $\alpha_{ml} \geq 0$, $\sum_{l=1}^{N_m} \alpha_{ml} = 1$, $\mathbf{x}_{ml}$ is assumed to be the mean of distribution $p_{ml}$ and $\Theta_{ml}$ denotes the other parameters/properties apart from the mean. In general, $p_{ml}$ could be chosen according to any convenient parametric or non-parametric family of distributions. In biological terms, $\alpha_{ml}$ reflects the amplification of a variable

sequence region and how probable that is in a given dataset with a sufficient level of coverage. The approach of using training data $\mathbf{x}_{ml}$ as the mean of $p_{ml}$ stems from a standard approach of using kernel density estimators (see section 2.5.1 of [6]).

Given a test set of $k$-mers (computed from reads), the distribution of the test set is modeled as

$$p(\mathbf{x}) = \sum_{m=1}^{M} p(\mathcal{C}_m)\ p(\mathbf{x}|\mathcal{C}_m),$$

where we denote probability for taxonomic unit $m$ (or class weight) by $p(\mathcal{C}_m)$, satisfying $\sum_{m=1}^{M} p(\mathcal{C}_m) = 1$. Note that $\{p(\mathcal{C}_m)\}_{m=1}^{M}$ is the composition of taxonomic units. The inference task is to estimate $p(\mathcal{C}_m)$ as accurately and fast as possible, for which a first order moment matching approach is developed. We first evaluate the mean of $\mathbf{x}$ under $p(\mathbf{x})$ as follows

$$\begin{aligned}
&\mathbb{E}[\mathbf{x}] \\
&= \textstyle\int \mathbf{x}\ p(\mathbf{x})\, d\mathbf{x} \in \mathbb{R}_{+}^{4^k \times 1} \\
&= \textstyle\sum_{m=1}^{M} p(\mathcal{C}_m) \int \mathbf{x}\ p(\mathbf{x}|\mathcal{C}_m)\, d\mathbf{x} \\
&= \textstyle\sum_{m=1}^{M} p(\mathcal{C}_m) \int \mathbf{x} \sum_{l=1}^{N_m} \alpha_{ml}\ p_{ml}(\mathbf{x}|\mathbf{x}_{ml}, \Theta_{ml})\, d\mathbf{x} \\
&= \textstyle\sum_{m=1}^{M} p(\mathcal{C}_m) \sum_{l=1}^{N_m} \alpha_{ml} \int \mathbf{x}\ p_{ml}(\mathbf{x}|\mathbf{x}_{ml}, \Theta_{ml})\, d\mathbf{x} \\
&= \textstyle\sum_{m=1}^{M} p(\mathcal{C}_m) \sum_{l=1}^{N_m} \alpha_{ml}\, \mathbf{x}_{ml}.
\end{aligned}$$

Introducing a new indexing $n \triangleq n(m,l) = \sum_{j=1}^{m-1} N_j + l$, we can write

$$\mathbb{E}[\mathbf{x}] = \sum_{n=1}^{N} \gamma_n\, \mathbf{x}_n = \mathbf{X}\boldsymbol{\gamma},$$

where

$$\begin{aligned}
&\boldsymbol{\gamma} = [\gamma_1\, \gamma_2\, \ldots, \gamma_N]^T \in \mathbb{R}_{+}^{N \times 1}, \\
&\gamma_n \triangleq \gamma_{n(m,l)} = p(\mathcal{C}_m)\alpha_{ml},
\end{aligned} \tag{2}$$

with the following properties

$$\begin{aligned}
&\sum_{n(m,1)}^{n(m,N_m)} \gamma_n = p(\mathcal{C}_m) \sum_{l=1}^{N_m} \alpha_{ml} = p(\mathcal{C}_m), \\
&\textstyle\sum_{n=1}^{N} \gamma_n = \|\boldsymbol{\gamma}\|_1 = 1.
\end{aligned}$$

In our approach we use the sample mean of the test set. The test set consists of $k$-mers feature vectors computed from reads. Each read is processed individually to generate $k$-mers in the same manner used for the reference data. We compute sample mean of the $k$-mer feature vectors for test dataset reads. Let us denote the sample mean of the test dataset by $\boldsymbol{\mu} \in \mathbb{R}_{+}^{4^k \times 1}$, and assume that the number of reads is reasonably high such that $\boldsymbol{\mu} \approx \mathbb{E}[\mathbf{x}]$. Then we can write

$$\boldsymbol{\mu} \approx \mathbf{X}\boldsymbol{\gamma}.$$

Considering that model irregularities are absorbed in an additive noise term $\mathbf{n}$, we use the following system model

$$\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\gamma} + \mathbf{n} \in \mathbb{R}_{+}^{4^k \times 1}. \tag{3}$$

Using the sample mean $\boldsymbol{\mu}$ and knowing $\mathbf{X}$, we estimate $\boldsymbol{\gamma}$ from (3) as $\hat{\boldsymbol{\gamma}} \triangleq [\hat{\gamma}_1\, \hat{\gamma}_2\, \ldots, \hat{\gamma}_N]^T \in \mathbb{R}_{+}^{N \times 1}$ followed by estimation of $p(\mathcal{C}_m)$ as

$$\hat{p}(\mathcal{C}_m) = \sum_{n(m,1)}^{n(m,N_m)} \hat{\gamma}_n.$$

Note that the estimation $\hat{\boldsymbol{\gamma}} \in \mathbb{R}_{+}^{N \times 1}$ must satisfy the following constraints

$$\begin{aligned}
&\hat{\boldsymbol{\gamma}} \geq \mathbf{0}, \\
&\|\hat{\boldsymbol{\gamma}}\|_1 = \textstyle\sum_{n=1}^{N} \hat{\gamma}_n = \sum_{m=1}^{M} \hat{p}(\mathcal{C}_m) = 1.
\end{aligned} \tag{4}$$

In (4), $\hat{\boldsymbol{\gamma}} \geq \mathbf{0}$ means $\forall n$, $\hat{\gamma}_n \geq 0$. We note that the linear setup (3) is under-determined as $4^k < N$ (in practice $4^k \ll N$) and hence, in general, solving (3) without any constraint will lead to infinitely many solutions. The constraints (4) result in a feasible set of solutions that is convex and can be used for finding a unique and meaningful solution.

We recall that the main interest is to estimate $p(\mathcal{C}_m)$, which is achieved in our approach by first estimating $\boldsymbol{\gamma}$ and then $p(\mathcal{C}_m)$. Hence $\boldsymbol{\gamma}$ represents an auxiliary variable in our system.

### 2.4 Optimization problem and sparsity aspect

The solution of (3), denoted by $\hat{\boldsymbol{\gamma}}$, must satisfy the constraints in (4). Hence, for SEK, we pose the optimization problem to solve as follows

$$\mathrm{P}_{\mathrm{sek}}^{+,1}: \qquad \hat{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\gamma}} \|\boldsymbol{\mu} - \mathbf{X}\boldsymbol{\gamma}\|_2, \boldsymbol{\gamma} \geq 0, \|\boldsymbol{\gamma}\|_1 = 1, \tag{5}$$

where '+' and '1' notations in $\mathrm{P}_{\mathrm{sek}}^{+,1}$ refer to the constraints $\hat{\boldsymbol{\gamma}} \in \mathbb{R}_{+}^{N}$ and $\|\hat{\boldsymbol{\gamma}}\|_1 = 1$, respectively. The problem $\mathrm{P}_{\mathrm{sek}}^{+,1}$ is a constrained least squares problem and a quadratic program (QP) solvable by convex optimization tools, such as **cvx** [1]. In our assumption $4^k < N$, and hence the required computation complexity is $\mathcal{O}(N^3)$ [7].

The form of $\mathrm{P}_{\mathrm{sek}}^{+,1}$ bears resembance to the widely used LASSO-method from general sparse signal processing, mainly used for solving under-determined problems in compressive sensing [9, 11]. LASSO deals with the following optimization problem (see (1.5) of [14])

$$\mathrm{LASSO}: \qquad \hat{\boldsymbol{\gamma}}_{\mathrm{lasso}} = \arg\min_{\boldsymbol{\gamma}} \|\boldsymbol{\mu} - \mathbf{X}\boldsymbol{\gamma}\|_2, \|\boldsymbol{\gamma}\|_1 \leq \tau,$$

where $\tau \in \mathbb{R}_{+}$ is a user choice that decides the level of sparsity in $\hat{\boldsymbol{\gamma}}_{\mathrm{lasso}}$; for example $\tau = 1$ will lead to a certain level of sparsity. A decreasing $\tau$ leads to an increasing level of sparsity in LASSO solution. LASSO is often presented in an unconstrained Lagrangian form that minimizes $\{\|\boldsymbol{\mu} - \mathbf{X}\boldsymbol{\gamma}\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_1\}$, where $\lambda$ decides the level of sparsity. $\mathrm{P}_{\mathrm{sek}}^{+,1}$ is not theoretically bound to provide a sparse solution with a similar level of sparsity achieved by LASSO when a small $\tau < 1$ is used.

For the community composition estimation problem, the auxiliary variable $\boldsymbol{\gamma}$ defined in (2) is inherently sparse. Two particularly natural motivations concerning the sparsity can be brought forward. Firstly, consider the conditional densities for taxonomic units as shown in (1). Regarding the conditional density model for a single unit, a natural hypothesis for the generating model is that the conditional densities for several other units will induce only few feature vectors, and hence $\alpha_{ml}$ will be negligible or effectively zero for certain patterns in the feature space, leading to sparsity in the auxiliary variable $\boldsymbol{\gamma}$ (unstructured sparsity in $\boldsymbol{\gamma}$). Secondly, in most

samples only a small fraction of the possible taxonomic units is expected to be present, and consequently, many $p(\mathcal{C}_m)$ will turn out to be zero, which again corresponds to sparsity in $\boldsymbol{\gamma}$ (structured block-wise sparsity in $\boldsymbol{\gamma}$) [22]. In practice, for a highly underdetermined system (3) in the community composition estimation problem with the fact that $\boldsymbol{\gamma}$ is inherently sparse, the solution of $\mathrm{P}_{\mathrm{sek}}^{+,1}$ turns out to be effectively sparse due to the constraint $\|\boldsymbol{\gamma}\|_1 = 1$.

### 2.5 A greedy estimation algorithm

For SEK we solve $\mathrm{P}_{\mathrm{sek}}^{+,1}$ using convex optimization tools requiring computational complexity $\mathcal{O}(N^3)$. To reduce the complexity without a significant loss in estimation performance we also develop a new greedy algorithm based on orthogonal matching pursuit (OMP) [23], for a short discussion of OMP with pseudo-code, see also [11]. In the recent literature several algorithms have been designed by extending OMP, such as, for example, the backtracking based OMP [16], and, by a subset of the current authors, the look-ahead OMP [10]. Since the standard OMP uses a least-squares approach and does not provide solutions satisfying constraints in (4), it is necessary to design a new greedy algorithm for the problem addressed here.

The new algorithm introduced here is referred to as $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$, and its pseudo-code is shown in Algorithm 1. In the stopping condition (step 7), the parameter $\nu$ is a positive real number that is used as a threshold and the parameter $I$ is a positive integer that is used to limit the number of iterations. The choice of $\nu$ and $I$ is ad-hoc, depending mainly on user experience.

**Algorithm 1** : $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$

*Input:*

1: $\mathbf{X}$, $\boldsymbol{\mu}$, $\nu$, $I$;

*Initialization:*

1: $\mathbf{r}_0 \leftarrow \boldsymbol{\mu}$, $\mathcal{S}_0 \leftarrow \emptyset$, $i \leftarrow 0$;

*Iterations:*

1: **repeat**
2: $i \leftarrow i + 1$; (Iteration counter)
3: $\tau_i \leftarrow$ index of the highest positive element of $\mathbf{X}^t \mathbf{r}_{i-1}$;
4: $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \tau_i$; ($|\mathcal{S}_i| = i$)
5: $\tilde{\boldsymbol{\gamma}}_i \leftarrow \arg\min_{\boldsymbol{\beta}_i} \|\boldsymbol{\mu} - \mathbf{X}_{\mathcal{S}_i}\boldsymbol{\beta}_i\|_2,\ \boldsymbol{\beta}_i \geq \mathbf{0}$; ($\mathbf{X}_{\mathcal{S}_i} \in \mathbb{R}_+^{4^k \times i}$)
6: $\mathbf{r}_i \leftarrow \boldsymbol{\mu} - \mathbf{X}_{\mathcal{S}_i}\tilde{\boldsymbol{\gamma}}_i$; (Residual)
7: **until** $((|\|\tilde{\boldsymbol{\gamma}}\|_1 - 1| \leq \nu)$ or $(i \geq I))$

*Output:*

1: $\hat{\boldsymbol{\gamma}} \in \mathbb{R}_+^N$, satisfying $\hat{\boldsymbol{\gamma}}_{\mathcal{S}_i} = \tilde{\boldsymbol{\gamma}}_i$ and $\hat{\boldsymbol{\gamma}}_{\overline{\mathcal{S}}_i} = \mathbf{0}$.
2: $\hat{\boldsymbol{\gamma}} \leftarrow \frac{\hat{\boldsymbol{\gamma}}}{\|\hat{\boldsymbol{\gamma}}\|_1}$ (Enforcing $\|\hat{\boldsymbol{\gamma}}\|_1 = 1$)

Compared to the standard OMP, the new aspects in $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$ are as follows:

- In step 3 of *Iterations*, we only search within positive inner product coefficients.
- In step 5 of *Iterations*, a least-squares solution $\tilde{\boldsymbol{\gamma}}_i$ with non-negativity constraint is found for $i$th iteration via the use of intermediate variable $\boldsymbol{\beta}_i \in \mathbb{R}_+^{i \times 1}$. In this step, $\mathbf{X}_{\mathcal{S}_i}$ is the sub-matrix formed by columns of $\mathbf{X}$ indexed in $\mathcal{S}_i$. The concerned optimization problem is convex. We used the Matlab function $\mathbf{lsqnonneg}()$ for this purpose.
- In step 6 of *Iterations*, we find the least squares residual $\mathbf{r}_i$.
- In step 7 of *Iterations*, the stopping condition provides for a solution that has an $\ell_1$ norm close to one, with an error decided by the threshold $\nu$. An unconditional stopping condition is provided by the maximum number of iterations $I$.
- In step 2 of *Output*, the $\ell_1$ norm of the solution is set to one by a rescaling.

The computational complexity of the $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$ algorithm is as follows. The main cost is incurred at step 5 where we need to solve a linearly constrained quadratic program using convex optimization tools; here we assume that the costs of the other steps are negligible. In the $i$th iteration $\mathbf{X}_{\mathcal{S}_i} \in \mathbb{R}_+^{4^k \times i}$ and $i \ll 4^k$, and the complexity required to solve step 5 is $\mathcal{O}(4^k i^2)$ [7]. As we have a stopping condition $i \leq I$, the total complexity of the $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$ algorithm is within $\mathcal{O}(I \times 4^k I^2) = \mathcal{O}(4^k I^3)$. We know that optimal solution of $\mathrm{P}_{\mathrm{sek}}^{+,1}$ using convex optimization tools requires a complexity of $\mathcal{O}(N^3)$. For a setup with $I < 4^k \ll N$, we can have $\mathcal{O}(4^k I^3) \ll \mathcal{O}(N^3)$, and hence the $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$ algorithm is typically much more efficient than using convex optimization tools directly in a high-dimensional setting. It is clear that the $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$ algorithm is not allowed to iterate beyond the limit of $I$; in practice this works as a forced convergence. For both $\mathrm{OMP}_{\mathrm{sek}}^{+,1}$ and $\mathrm{P}_{\mathrm{sek}}^{+,1}$, we do not have a theoretical proof on robust reconstruction of solutions. Further a natural question remains on how to set the input parameters $\nu$ and $I$. The choice of parameters is discussed later in section 3.4.

### 2.6 Overall system flow-chart

Finally we depict the full SEK system by using a flow-chart shown in Figure 1. The flow-chart shows main parts of the overall system, and associated off-line and on-line computations.

### 2.7 Mock communities data

For our experiments on real biological data, we used the mock microbial communities database developed in [15]. The database is called even composition Mock Communities (eMC) for chimeric sequence detection where the involved bacterial species are known in advance. Three regions (V1-V3, V3-V5, V6-V9) of the 16S rRNA gene of the composition eMC were sequenced using 454 sequencing technology in four different sequencing centers. In our experiments we focused on the V3-V5 region datasets, since these have been earlier used for evaluation of the BeBAC method (see Experiment 2 of [12]).

*2.7.1 Test dataset (Reads):* Our basic test dataset used under a variety of different *in silico* experimental conditions is the one used in Experiment 2 of BeBAC [12]. The test dataset consists of 91240 short length reads from 21 different species. The length of reads has a range between 450-550 bp and the bacterial community composition is known at the species level, by the following computation performed in [12]. Each individual sequence of the 91240 read sequences was aligned (local alignment) to all the reference sequences of reference database $D_{\mathrm{known}}^{\mathrm{mock}}$ described in the

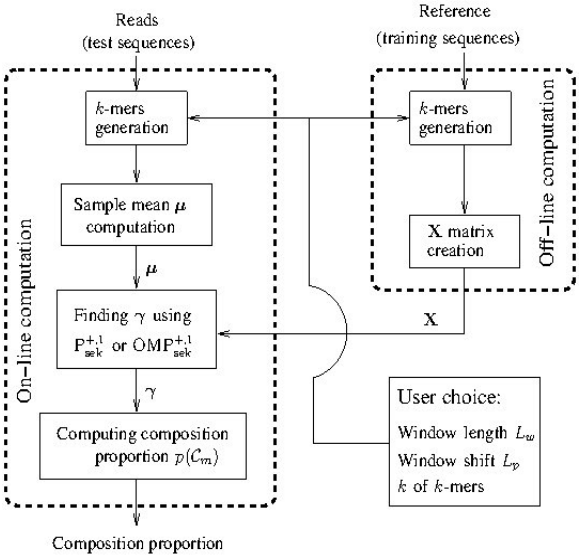


**Fig. 1.** A flow-chart of full SEK system.

section 2.7.2 and then each read sequence is labelled by the species of the highest scoring reference sequence, followed by computation of the community composition referred to as ground truth.

*2.7.2 Training datasets (Reference):* We used two different databases (known and mixed) generated from the mock microbial community database [15]. The first database is denoted by $D^{\text{mock}}_{\text{known}}$ and it consists of the same $M = 21$ species present among the reads described in section 2.7.1. The details of the $D^{\text{mock}}_{\text{known}}$ database can be found in Experiment 2 of [12]. The database consists of 113 reference sequences for a total of 21 bacterial species, such that each reference sequence represents a distinct 16S rRNA gene. Thus there is a varying number of reference sequences for each of the considered species. Each reference sequence has an approximate length of 1500 bp, and for each species, the corresponding reference sequences are concatenated to a single sequence. The final reference database $D^{\text{mock}}_{\text{known}}$ then consists of 21 sequences where each sequence has an approximate length 5000 bp.

To evaluate influence of new species in reference data on the performance of SEK, we created new databases denoted by $D^{\text{mock}}_{\text{mixed}}(E)$. Here $E$ represents the number of additional species included to a partial database created from $D^{\text{mock}}_{\text{known}}$, by downloading additional reference data from the RDP database. Each partial database includes only one randomly chosen reference sequence for each species in $D^{\text{mock}}_{\text{known}}$ and hence consists of 21 reference sequences of approximate length 1500 bp. For example, with $E = 10$, 10 additional species were included in the reference database and consequently $D^{\text{mock}}_{\text{mixed}}(10)$ contains 16S rRNA sequences of $M = 21 + 10 = 31$ species. Several instances of $D^{\text{mock}}_{\text{mixed}}(E)$ were made for each fixed value of $E$ by choosing a varying set of additional species and we also increased $E$ from zero to 100 in steps of 10. Note that, in $D^{\text{mock}}_{\text{mixed}}(E)$, the inclusion of only single reference sequence results in reduction of biological variability for each of the original 21 species compared to $D^{\text{mock}}_{\text{known}}$.

### 2.8 Simulated data

To evaluate how SEK performs for much larger data than the mock communities data, we performed experiments for simulated data described below.

*2.8.1 Test datasets (Reads):* Two sets of simulated data were used to test the performance of the SEK method. First, the 216 different simulated datasets produced in [18] were used for a direct comparison to the Quikr method and the Ribosomal Database Project's (RDP) Naïve Bayesian Classifier (NBC). See [18, §2.5] for the design of these simulations.

The second set of simulated data consists of 486 different pyrosequencing datasets constituting over 179M reads generated using the shotgun/amplicon read simulator Grinder [4]. Read-length distributions were set to be one of the following: fixed at 100bp, normally distributed at $450\text{bp} \pm 50\text{bp}$, or normally distributed at $800\text{bp} \pm 100\text{bp}$. Read depth was fixed to be one of 10K, 100K, or 1M total reads. Primers were chosen to target either only the V1-V3 regions, only the V6-V9 regions, or else the multiple variable regions V1-V9. Three different diversity values were chosen (10, 100, and 500) at the species level, and abundance was modeled by one of the following three distributions: uniform, linear, or power-law with parameter 0.705. Homopolymer errors were modeled using Balzer's model [5], and chimera percentages were set to either $5\%$ or $35\%$. Since only amplicon sequencing is considered, copy bias was employed, but not length bias.

*2.8.2 Training datasets (Reference):* To analyze the simulated data, two different training matrices were used corresponding to the databases $D_{\text{small}}$ and $D_{\text{large}}$ from [18]. The database $D_{\text{small}}$ is identical to RDP's NBC training set 7 and consists of 10,046 sequences covering 1,813 genera. Database $D_{\text{large}}$ consists of a 275,727 sequence subset of RDP's 16S rRNA database covering 2,226 genera. Taxonomic information was obtained from NCBI.

## 3 RESULTS

### 3.1 Performance measure and competing methods

As a quantitative performance measure, we use variational distance (VD) to compare between known proportions of taxonomic units $\mathbf{p} = [p(\mathcal{C}_1), p(\mathcal{C}_2), \ldots, p(\mathcal{C}_K)]^T$ and the estimated proportions $\hat{\mathbf{p}} = [\hat{p}(\mathcal{C}_1), \hat{p}(\mathcal{C}_2), \ldots, \hat{p}(\mathcal{C}_K)]^T$. The VD is defined as

$$\text{VD} = 0.5 \times \|\mathbf{p} - \hat{\mathbf{p}}\|_1 \in [0, 1].$$

A low VD indicates more satisfactory performance.

We compare performances between SEK, Quikr, Taxy and RDP's NBC, for real biological data (mock communities data) and large size simulated data.

### 3.2 Results for Mock Communities data

Using mock communities data, we carried out experiments where the community composition problem is addressed at the species level. Here we investigated how the SEK performs for real biological data, also vis-a-vis relevant competing methods.

*3.2.1 k-mers from test dataset:* In the test dataset, described in section 2.7.1, the shortest read is of length 450 bp. We used a

window length $L_w = 450$ bp and refrained from the sliding-the-window approach in the generation of $k$-mers feature vectors. For $k = 4$ and $k = 6$, the $k$-mers generation took 21 minutes and 48 minutes, respectively.

*3.2.2 Results using small training dataset:* In this experiment, we used SEK for estimation of the proportions of species in the test set described in Section 2.7.1. Here we used the smaller training reference set $D^{\text{mock}}_{\text{known}}$ described in Section 2.2. The experimental setup is the same as shown in Experiment 2 of BeBAC [12]. Therefore we can directly compare with the BeBAC results reported in [12]. SEK estimates were based on 4-mers computed with the setup $L_w = 450$ bp and $L_p = 1$ bp. The choice of $L_p = 1$ bp corresponds to the best case of generating training matrix $\mathbf{X}$, with the highest amount of variability in reference $k$-mers. Using $D^{\text{mock}}_{\text{known}}$, the $k$-mers training matrix $\mathbf{X}$ has the dimension $4^4 \times 121412$. For the use of SEK in such a high dimension, the QP $\text{P}^{+,1}_{\text{sek}}$ using **cvx** suffered of numerical instability, but $\text{OMP}^{+,1}_{\text{sek}}$ provided results in 3.17 seconds, leading to a VD = 0.0305. For $\text{OMP}^{+,1}_{\text{sek}}$, $\nu$ and $I$ in algorithm 1 were set to $10^{-5}$ and 100 respectively; the values of these two parameters remained unchanged for other experiments on mock communities data presented later. The performance of SEK using $\text{OMP}^{+,1}_{\text{sek}}$ is shown in Figure 2, and compared against the estimates from BeBAC, Quikr and Taxy. The Quikr method used 6-mers and provided a VD = 0.4044, whereas the Taxy method used 7-mers and provided a VD = 0.2817. The use of $k = 6$ and $k = 7$ for Quikr and Taxy, respectively, is chosen according to the experiments described in [18] and [19]. Here Quikr is found to provide the least satisfactory performance in terms of VD. BeBAC results are highly accurate with VD = 0.0038, but come with the requirement of a computation time in the order of more than thirty hours. On the other hand $\text{OMP}^{+,1}_{\text{sek}}$ had a total online computation time around 21 minutes that is mainly dominated by $k$-mers computation from sample reads for evaluating $\boldsymbol{\mu}$; given pre-computed $\mathbf{X}$ and $\boldsymbol{\mu}$, the central inferenece (or estimation) task of $\text{OMP}^{+,1}_{\text{sek}}$ took only 3.17 seconds. Considering that Quikr and Taxy also have similar online complexity requirement to compute $k$-mers from sample reads, $\text{OMP}^{+,1}_{\text{sek}}$ can be concluded to provide a good trade-off between performance and computational demands.

*3.2.3 Results for dimension reduction by higher shifts:* The $L_p = 1$ bp leads to a relatively high dimension of $\mathbf{X}$, which is directly related to an increase in computational complexity. Clearly, the $L_p = 1$ bp shift produces highly correlated columns in $\mathbf{X}$ and consequently it might be sufficient to utilize $k$-mers feature vectors with a higher shift without a considerable loss in variability information. To investigate this, we performed an experiment with a gradual increase in $L_p$. We found that selecting $L_p = 15$ bp results in an input $\mathbf{X} \in \mathbb{R}_+^{4^4 \times 8052}$ which the **cvx** based $\text{P}^{+,1}_{\text{sek}}$ was able to process successfully. At $L_p = 15$ bp, the $\text{P}^{+,1}_{\text{sek}}$ provided a performance of VD = 0.033260, while the execution time was 25.25 seconds. The $\text{OMP}^{+,1}_{\text{sek}}$ took 1.86 seconds and provided VD = 0.03355l, indicating almost no performance loss compared to the optimal $\text{P}^{+,1}_{\text{sek}}$. A shift $L_p > 25$ did result in a performance drop, for example, $L_p = 30, 50, 100$ resulted in VD values 0.0527, 0.0879, 0.1197, respectively. Therefore, shifts around $L_p = 15$ bp appear to be sufficient to reduce the dimension of $\mathbf{X}$, while maintaining sufficient biological variability. Hence the next experiment (in section 3.2.4) was conducted using $L_p = 15$ bp.

*3.2.4 Results for mixed training dataset:* In this experiment, we investigated how the performance of SEK varies with an increase in the number of additional species in the reference training database which are not present in the sample test data. We used reference training datasets $D^{\text{mock}}_{\text{mixed}}(E)$ described in Section 2.2, where $E = 0, 10, 20, \ldots, 100$. For each non-zero $E$, we created 10 reference datasets to evaluate variability of the performance. The performance with one-sigma error bars is shown in Figure 3. The trend in the performance curves confirms that the SEK is subjected to gradual decrease in performance with the increase in the number of additional species; the trend holds for both $\text{P}^{+,1}_{\text{sek}}$ and $\text{OMP}^{+,1}_{\text{sek}}$. Also, being optimal the performance of QP $\text{P}^{+,1}_{\text{sek}}$ is found to be more consistent than the greedy $\text{OMP}^{+,1}_{\text{sek}}$.

## 3.3 Results for Simulated Data

The simulated data experiments deal with community composition problem at different taxonomic ranks and also with very large size of $\mathbf{X}$ in (3). Due to the massive size of $\mathbf{X}$, a direct application of QP $\text{P}^{+,1}_{\text{sek}}$ is not feasible, and hence we used only $\text{OMP}^{+,1}_{\text{sek}}$. For all results described, $\nu$ and $I$ in algorithm 1 were set to $10^{-5}$ and 409 respectively.

*3.3.1 Training matrix construction:* In forming the training matrix for $D_{\text{small}}$, the $k$-mer size was fixed at $k = 6$, and the window length and position shifts were set to $L_w = 400$ and $L_p = 100$ respectively. This resulted in a matrix $\mathbf{X}$ with dimensions $4^6 \times 109,773$. For the database $D_{\text{large}}$, a training matrix $\mathbf{X}$ with dimensions $4^6 \times 500,734$ was formed by fixing $k = 6, L_w = 400$, and $L_p = 400$. Calculating the matrices took $\sim 2.5$ and $\sim 11$ minutes respectively using an Intel i7-4930K processor and a custom C program. Slightly varying $L_p$ and $L_w$ did not significantly change the results contained in sections 3.3.2 and 3.3.3 below, but generally decreasing $L_p$ and $L_w$ results in lower reconstruction error at the expense of increased execution time and memory usage. The values of $L_p$ and $L_w$ were chosen to provide an acceptable balance between execution time, memory usage, and reconstruction error.

*3.3.2 Results for first set of simulated data:* For test data, $k$-mers were computed in the same manner as described in section 3.3.1. On average, 4.0 seconds were required to form the 6-mer feature vector for each sample. Figure 4 compares the mean variational distance (VD) error at various taxonomic ranks as well as the algorithm execution time between SEK ($\text{OMP}^{+,1}_{\text{sek}}$), Quikr and RDP's NBC.

As shown in figure 4, using the database $D_{\text{large}}$, SEK outperforms both Quikr and RDP's NBC in terms of reconstruction error and has comparable execution time as Quikr. Both Quikr and SEK have significantly lower execution time than RDP's NBC. Using the database $D_{\text{small}}$ (not shown here), SEK continues to outperform both Quikr and RDP's NBC in terms of reconstruction error, but only RDP's NBC in terms of execution time, as SEK had a median execution time of 15.2 minutes versus Quikr's 25 seconds. All three methods have increasing error for lower taxonomic ranks, but the improvement of SEK over Quikr is emphasized for lower taxonomic ranks.

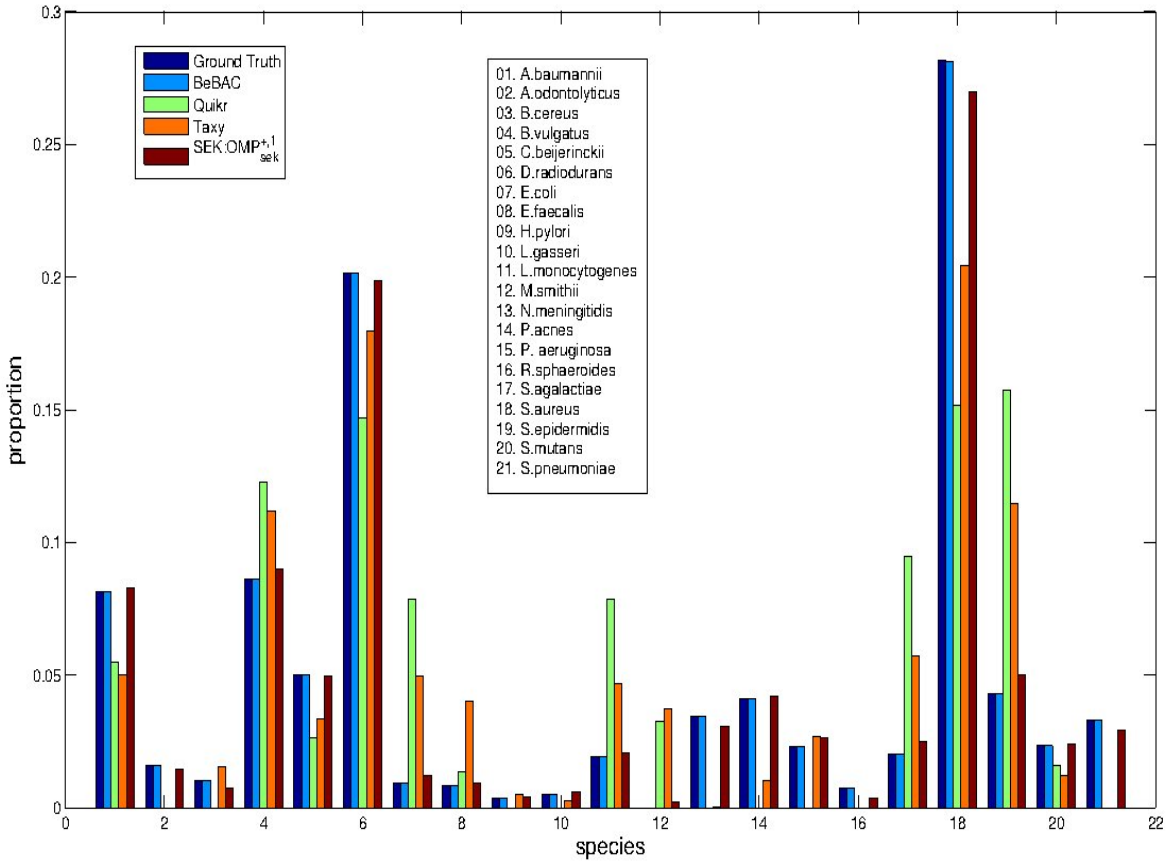


**Fig. 2.** For mock communities data: Performance of $\mathrm{OMP}^{+,1}_{\mathrm{sek}}$ using reference training database $D^{\mathrm{mock}}_{\mathrm{known}}$. Community composition problem is addressed at the species level. The $\mathrm{OMP}^{+,1}_{\mathrm{sek}}$ performance is shown against the ground truth and performances of BeBAC, Quikr and Taxy. The $\mathrm{OMP}^{+,1}_{\mathrm{sek}}$ provides better match to the ground truth than the competing faster methods Quikr and Taxy. The corresponding variational distance (VD) performances of BeBAC, $\mathrm{OMP}^{+,1}_{\mathrm{sek}}$, Taxy and Quikr are 0.0038, 0.0305, 0.2817 and 0.4044, respectively.

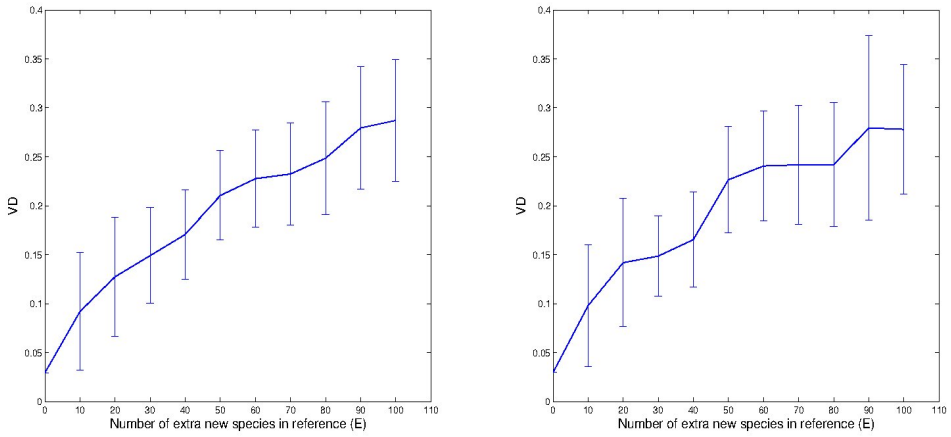


**Fig. 3.** For mock communities data: Variational distance (VD) performance of SEK against increasing reference database $D^{\mathrm{mock}}_{\mathrm{mixed}}(E)$, where $E = 0, 10, 20, \ldots, 100$. The left figure is for $\mathrm{P}^{+,1}_{\mathrm{sek}}$ and the right figure is for $\mathrm{OMP}^{+,1}_{\mathrm{sek}}$. The results show that both SEK implementations are subjected to a gradual decrease in performance with the increase in the number of additional species.

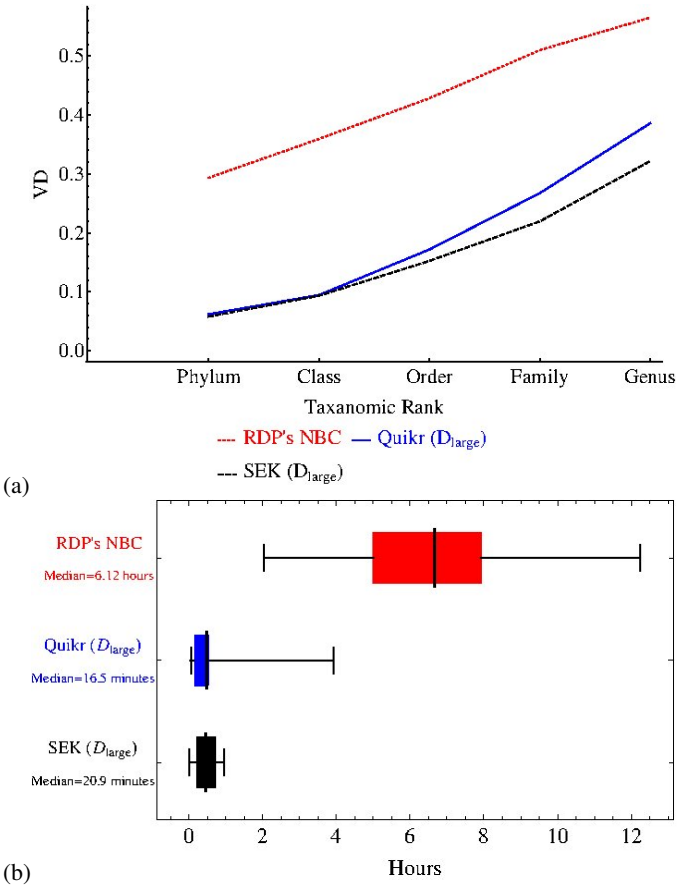


**Fig. 4.** For simulated data: Comparison of SEK ($\text{OMP}^{+,1}_{\text{sek}}$) to Quikr and RDP's NBC on the first set of simulated data. Throughout, RDP's NBC version 10.28 with training set 7 was utilized. (a) Variational distance error averaged over all 216 simulated datasets versus taxonomic rank for RDP's NBC, with SEK and Quikr trained using $D_{\text{large}}$. (b) Algorithm execution time for RDP's NBC, with SEK and Quikr trained using $D_{\text{large}}$. Whiskers denote range of the data, vertical black bars designate the median, and the boxes demarcate quantiles.

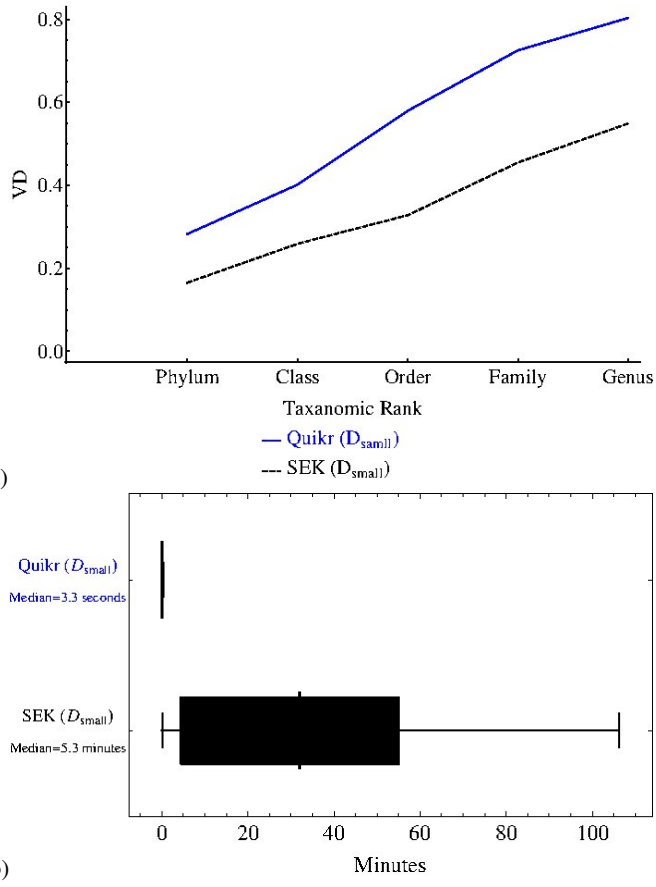


**Fig. 5.** For simulated data: Comparison of SEK ($\text{OMP}^{+,1}_{\text{sek}}$) to Quikr on the second set of simulated data. (a) Variational distance error averaged over all 486 simulated datasets versus taxonomic rank for SEK and Quikr trained using $D_{\text{small}}$. (b) Algorithm execution time for SEK and Quikr trained using $D_{\text{small}}$. Whiskers denote range of the data, vertical black bars designate the median, and the boxes demarcate quantiles.

*3.3.3 Results for second set of simulated data:* Figure 5 summarizes the mean VD and algorithm execution time over the second set of simulated data described in section 2.8 for Quikr and SEK both trained on $D_{\text{small}}$.

Part (a) of Figure 5 demonstrates that SEK shows much lower VD error in comparison to Quikr at every taxonomic rank. However, part (b) of Figure 5 shows that this improvement comes at the expense of moderately increased mean execution time.

When focusing on the simulated datasets of length 100bp, $450\text{bp} \pm 50\text{bp}$, and $800\text{bp} \pm 100\text{bp}$, SEK had a mean VD of 0.803, 0.410, and 0.436 respectively. As $L_w$ was set to 400, this indicates the importance of choosing $L_w$ to roughly match the sequence length of a given sample when forming the $k$-mer training matrix if sequence length is reasonably short (around 400 bp).

SEK somewhat experienced decreasing performance as a function of diversity: at the genus level, SEK gave a mean VD of 0.467, 0.579, and 0.603 for the simulated datasets with diversity 10, 100, and 500 respectively.

### 3.4 Remarks on parameter choice and errors

In SEK, we need to choose several parameters: $k$, $L_w$, $L_p$, $\nu$ and $I$. Typically an increase in $k$ leads to better performance with the fact that a higher $k$ always subsumes a lower $k$ in the process of generating $k$-mers feature vectors. The trend of improvement in performance with increase of $k$ was shown for Quikr [18] and we believe that the same trend will also hold for SEK. For SEK, the increase in $k$ results in exponential increase in row dimension of $\mathbf{X}$ matrix and hence the complexity and memory requirement also increase exponentially. There is no standard approach to fix $k$, except a brute force search. Let us now consider choice of $L_w$ and $L_p$. Our experimental results bring the following heuristic: choose $L_w$ to match the read length of sample data. On the other hand, choose $L_p$ as small as possible to accommodate a high variability of $k$-mers information in $\mathbf{X}$ matrix. A reduction in $L_p$ results to a linear increase in column dimension of $\mathbf{X}$. Overall users should choose $k$, $L_w$ and $L_p$ such that the dimension of $\mathbf{X}$ remains reasonable without considerable loss in estimation performance. Finally we consider $\nu$ and $I$ parameters in Algorithm 1 that enforce sparsity, with the aspect that computational complexity is $\mathcal{O}(4^k I^3)$.

In general there is no standard automatic approach to choose these two parameters, even for any standard algorithm. For example, the unconstrained Lagrangian form of LASSO mentioned in section 2.4 also needs to set the parameter $\lambda$ by user. For Algorithm 1, $0 < \nu < 1$ should be chosen as a small positive number and $I$ can be chosen as a fraction of row dimension of $\mathbf{X}$ that is $4^k$, of-course with the requirement that $I$ is a positive integer. Let us choose $I = \lfloor \eta \times 4^k \rfloor$ where $0 < \eta \leq 1$. In case of a lower $k$, the system is more under-determined and naturally the enforcement of sparsity needs to be slackened to achieve a reasonable estimation performance. Hence for a lower $k$, we need to choose a higher $\eta$ that can provide a good trade-off between complexity and estimation performance. But, for a higher $k$, the system is less under-determined and to keep the complexity reasonable, we should choose a lower $\eta$. Note that, for mock communities date, we used $k = 4$ and $I = 100$, and hence $\eta = \frac{100}{4^4} \approx 0.4$, and for simulated data, we used $k = 6$ and $I = 409$, and hence $\eta = \frac{409}{4^6} \approx 0.1$.

Further, it is interesting to ask what are the types of errors most common in SEK reconstruction. In general, SEK reconstructs the most abundant taxa with remarkable fidelity. The less abundant taxa are typically more difficult to reconstruct and at times each behavior can be observed: low frequency taxa missing, miss-assigned, or their abundances miss-estimated.

## 4 DISCUSSION AND CONCLUSION

In this work we have shown that bacterial compositions of metagenomic samples can be determined quickly and accurately from what initially appears to be very incomplete data. Our method SEK uses only $k$-mer statistics of fixed length (here $k \sim 4, 6$) of reads from high-throughput sequencing data from the bacterial 16S rRNA genes to find which set of tens of bacteria are present out of a library of hundreds of species. For a reasonable size of reference training data, the computational cost is dominated by the pre-computing of the $k$-mer statistics in the data and in the library; the computational cost of the central inference module is negligible, and can be performed in seconds/minutes on a standard laptop computer.

Our approach belongs to the general family of sparse signal processing where data sparsity is exploited to solve under-determined systems. In metagenomics sparsity is present on several levels. We have utilized the fact that $k$-mer statistics computed in windows of intermediate size vary substantially along the 16S rRNA sequences. The number of variables representing the amount of reads assumed to be present in the data from each genome and from each window is thus far greater than the number of observations which are the $k$-mer statistics of all the reads in the data taken together. More generally, while many bacterial communities are rich and diverse, the number of species present in, for example the gut of one patient, will almost always be only a small fraction of the number of species present at the same position across a population, which in turn will only be a very small fraction of all known bacteria for which the genomic sequences are available. We therefore believe that sparsity is a rather common feature of metagenomic data analysis which could have many applications beyond the ones pursued here.

The major technical problem solved in the present paper stems from the fact that the columns of the system matrix $\mathbf{X}$ linking feature vectors are highly correlated. This effect arises both from the construction of the feature vectors i.e. that the windows are overlapping, and from biological similarity of DNA sequences along the 16S rRNA genes across a set of species. An additional technical complication is that the variables (species abundances) are non-negative numbers and naturally normalized to unity, while in most methods of sparse signal processing there are no such constraints. We were able to overcome these problems by constructing a new greedy algorithm based on orthogonal matching pursuit (OMP) modified to handle the positivity constraint. The new algorithm, dubbed $\mathrm{OMP}^{+,1}_{\mathrm{sek}}$, integrates ideas borrowed from kernel density estimators, mixture density models and sparsity-exploiting algebraic solutions.

During the manuscript preparation, we became aware that a similar methodology (Quikr) has been developed by Koslicki et al in [18]. While there is a considerable similarity between Quikr and SEK, we note that Quikr is based only on sparsity-exploiting algebraic solutions while SEK further exploits the additional sparsity assumption of non-uniform amplifications of variable regions in 16S rRNA sequences. Indeed, we hypothesize that the improvement of SEK over Quikr is mainly due to the superior training method of SEK. The comparison between the two methods reported above in Figures 2, 4 and 5 shows that SEK performs generally better than Quikr. The development of two new methodologies independently and roughly simultaneously reflect the timeliness and general interest of sparse processing techniques for bioinformatics applications.

## ACKNOWLEDGEMENT

*Funding*: This work was supported by the Erasmus Mundus scholar program of the European Union (Y.L.), by the Academy of Finland through its Finland Distinguished Professor program grant project 129024/Aurell (E.A.), ERC grant 239784 (J.C.) and the Academy of Finland Center of Excellence COIN (E.A. and J.C.), by the Swedish Research Council Linnaeus Centre ACCESS (E.A., M.S., L.R., S.C. and M.V), and by the Ohio Supercomputer Center and the Mathematical Biosciences Institute at The Ohio State University (D.K.).

*4.0.1 Conflict of interest statement.* None declared.

## REFERENCES


[1]Cvx: A system for disciplined convex programming. http://cvxr.com/cvx/. 2013.

[2]A. Amir, A. Zeisel, O. Zuk, M. Elgart, S. Stern, O. Shamir, J.P. Turnbaugh, Y. Soen, and N. Shental. High-resolution microbial community reconstruction by integrating short reads from multiple 16S rRNA regions. *Nucleic Acids Res.*, 41(22):e205, 2013.

[3]A. Amir and O. Zuk. Bacterial community reconstruction using compressed sensing. *J Comput Biol.*, 18(11):1723–41, 2011.

[4]F. E. Angly, D. Willner, F. Rohwer, P. Hugenholtz, and G. W. Tyson. Grinder: a versatile amplicon and shotgun sequence simulator. *Nucleic acids research*, 40(12):e94, 2012.

[5]S. Balzer, K. Malde, Lanzén A., A. Sharma, and I. Jonassen. Characteristics of 454 pyrosequencing data–enabling realistic simulation with flowsim. *Bioinformatics*, 26(18):i420–5, 2010.

[6]C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[7]S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[8]Y. Cai and Y. Sun. Esprit-tree: hierarchical clustering analysis of millions of 16s rrna pyrosequences in quasilinear computational time. *Nucleic Acids Research*, 39(14):e95, 2011.

[9]E.J. Candes and M.B. Wakin. An introduction to compressive sampling. *IEEE Signal Proc. Magazine*, 25:21–30, march 2008.

[10]S. Chatterjee, D. Sundman, and M. Skoglund. Look ahead orthogonal matching pursuit. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4024 –4027, may 2011.

[11]S. Chatterjee, D. Sundman, M. Vehkaperä, and M. Skoglund. Projection-based and look-ahead strategies for atom selection. *Signal Processing, IEEE Transactions on*, 60(2):634 –647, feb. 2012.

[12]L. Cheng, L.W. Walker, and J. Corander. Bayesian estimation of bacterial community composition from 454 sequencing data. *Nucleic Acids Research*, 2012.

[13]R. C. Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010.

[14]B. Effron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 2004.

[15]B.J. Haas, D. Gevers, A.M. Earl, M. Feldgarden, D.V. Ward, G. Giannoukos, D. Ciulla, D. Tabbaa, S.K. Highlander, E. Sodergren, B. Methe, T.Z. DeSantis, Human Microbiome Consortium, J.F. Petrosino, R. Knight, and B.W. Birren. Chimeric 16s rrna sequence formation and detection in sanger and 454-pyrosequenced pcr amplicons. *Genome Res.*, 21(3):494–504, 2011.

[16]H. Huang and A. Makur. Backtracking-based matching pursuit method for sparse signal reconstruction. *Signal Processing Letters, IEEE*, 18(7):391–394, 2011.

[17]D.H. Huson, A.F. Auch, J. Qi, and S.C. Schuster. Megan analysis of metagenomic data. *Genome Res.*, 17(3):377–386, 2007.

[18]D. Koslicki, S. Foucart, and G. Rosen. Quikr: a method for rapid reconstruction of bacterial communities via compressive sensing. *Bioinformatics*, 29(17):2096–2102, 2013.

[19]P. Meinicke, K.P. Aßhauer, and T. Lingner. Mixture models for analysis of the taxonomic composition of metagenomes. *Bioinformatics*, 27(12):1618–1624, 2011.

[20]S. Mitra, M. Stärk, and D.H. Huson. Analysis of 16s rrna environmental sequences using megan. *BMC Genomics*, 2011.

[21]S.H. Ong, V.U. Kukkillaya, A. Wilm, C. Lay, E.X.P. Ho, L. Low, M.L. Hibberd, and N. Nagarajan. Species identification and profiling of complex microbial communities using shotgun illumina sequencing of 16s rrna amplicon sequences. *PLoS One*, 8(4):e60811, 2013.

[22]M. Stojnic. $l_2/l_1$-optimization in block-sparse compressed sensing and its strong thresholds. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):350–357, 2010.

[23]J.A. Tropp and A.C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on*, 53(12):4655 –4666, dec. 2007.

[24]C. von Mering, P. Hugenholtz, J. Raes, S. G. Tringe, T. Doerks, L. J. Jensen, N. Ward, and P. Bork. Quantitative phylogenetic assessment of microbial communities in diverse environments. *Science*, 315(5815):1126–1130, 2007.

[25]Q. Wang, G.M. Garrity, J.M. Tiedje, and J.R Cole. Naïve bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Appl. Environ. Microbiol.*, 73(16):5261–5267, 2007.

[26]K.E. Wommack, J. Bhavsar, and J. Ravel. Metagenomics: read length matters. *Appl Environ Microbiol.*, 74(5):1453–63, 2008.

[27]O. Zuk, A. Amir, A. Zeisel, O. Shamir, and N. Shental. *Accurate profiling of microbial communities from massively parallel sequencing using convex optimization*, volume LNCS 8214. Springer, Cham, Switzerland, 2013.